TRƯỜNG ĐẠI HỌC TRÀ VINH

KHOA KỸ THUẬT VÀ CÔNG NGHỆ

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO MÔN HỌC**

**ỨNG DỤNG QUẢN LÝ LỊCH TRÌNH CÁ NHÂN**

|  |  |
| --- | --- |
| **Giáo viên giảng dạy:**  TS. Nguyễn Bảo Ân | **Sinh viên thực hiện:**  110122193 Thạch Thị Huệ Trinh DA22TTC  110122196 Lê Xuân Trường DA22TTC |

***Trà Vinh, tháng 5 năm 2025***

TRƯỜNG ĐẠI HỌC TRÀ VINH

KHOA KỸ THUẬT VÀ CÔNG NGHỆ

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO MÔN HỌC**

**ỨNG DỤNG QUẢN LÝ LỊCH TRÌNH CÁ NHÂN**

|  |  |
| --- | --- |
| **Giáo viên giảng dạy:**  TS. Nguyễn Bảo Ân | **Sinh viên thực hiện:**  110122193 Thạch Thị Huệ Trinh DA22TTC  110122196 Lê Xuân Trường DA22TTC |

***Trà Vinh, tháng 5 năm 2025***

**NHẬN XÉT CỦA GIÁO VIÊN**

*Trà Vinh, ngày ….. tháng …… năm ……*

**Giáo viên giảng dạy**

*(Ký tên và ghi rõ họ tên)*

# MỤC LỤC

[MỤC LỤC 1](#_Toc203586541)

[DANH MỤC HÌNH ẢNH 3](#_Toc203586542)

[CHƯƠNG 1. GIỚI THIỆU 4](#_Toc203586543)

[1.1. Tên đề tài và chủ đề 4](#_Toc203586544)

[1.2. Mục tiêu của ứng dụng 5](#_Toc203586545)

[1.3. Lí do chọn đề tài 5](#_Toc203586546)

[CHƯƠNG 2. PHÂN TÍCH YÊU CẦU 6](#_Toc203586547)

[2.1. Các chức năng chính của hệ thống (Functional Requirements). 6](#_Toc203586548)

[2.1.1. Đăng ký tài khoản người dùng 6](#_Toc203586549)

[2.1.2. Đăng nhập và xác thực người dùng 6](#_Toc203586550)

[2.1.3. Khôi phục mật khẩu 6](#_Toc203586551)

[2.1.4. Quản lý lịch trình cá nhân 6](#_Toc203586552)

[2.1.5. Quản lý nhiều sự kiện trong cùng một ngày 7](#_Toc203586553)

[2.1.6. Cài đặt tài khoản 7](#_Toc203586554)

[2.1.7. Giao diện người dùng thân thiện 7](#_Toc203586555)

[2.1.8. Đồng bộ hóa dữ liệu với backend 7](#_Toc203586556)

[2.2. Các yêu cầu phi chức năng (Non-functional Requirements). 8](#_Toc203586557)

[CHƯƠNG 3. KẾT QUẢ THỰC HIỆN 9](#_Toc203586558)

[3.1. Một số hình ảnh trong trò chơi 9](#_Toc203586559)

[3.1.1. Giao diện ban đầu 9](#_Toc203586560)

[3.1.2. Giao diện sau khi xáo trộn 10](#_Toc203586561)

[3.1.3. Giao diện kết thúc 11](#_Toc203586562)

[3.1.4. Giao diện giải bằng A Star 11](#_Toc203586563)

[3.2. Trạng thái kết thúc của trò chơi 12](#_Toc203586564)

[3.3. Một số kinh nghiệm trong trò chơi 14](#_Toc203586565)

[3.3.1. Ưu tiên giải quyết theo từng hàng hoặc cột 14](#_Toc203586566)

[3.3.2. Hạn chế làm rối phần đã hoàn thành 14](#_Toc203586567)

[3.3.3. Sử dụng ô trống thông minh 15](#_Toc203586568)

[3.3.4. Kiên nhẫn và đánh giá kỹ bước đi 15](#_Toc203586569)

[3.3.5. Tận dụng thuật toán A Star 15](#_Toc203586570)

[CHƯƠNG 4. ĐÁNH GIÁ VÀ HƯỚNG PHÁT TRIỂN 17](#_Toc203586571)

[4.1. Đánh giá 17](#_Toc203586572)

[4.1.1. Ưu điểm của trò chơi 17](#_Toc203586573)

[4.1.2. Hạn chế của trò chơi 17](#_Toc203586574)

[4.2. Hướng phát triển 18](#_Toc203586575)

[CHƯƠNG 5. KẾT LUẬN 19](#_Toc203586576)

[DANH MỤC TÀI LIỆU THAM KHẢO 21](#_Toc203586577)

# DANH MỤC HÌNH ẢNH

[Hình 2.1 Trí tuệ nhân tạo (Artificial Intelligence – AI) là ngành khoa học [1] 7](#_Toc198749270)

[Hình 2.2 Unity là một nền tảng phát triển game đa nền tảng [2] 8](#_Toc198749271)

[Hình 2.3 Thuật toán A\* là một thuật toán tìm kiếm [3] 10](#_Toc198749272)

[Hình 3.1 Hình ảnh giao diện ban đầu của trò chơi 13](#_Toc198749273)

[Hình 3.2 Hình ảnh giao diện sau khi xáo trộn 14](#_Toc198749274)

[Hình 3.3 Hình ảnh giao kết thúc của trò chơi 15](#_Toc198749275)

[Hình 3.4 Hình ảnh giao diện giải bằng A Star 16](#_Toc198749276)

[Hình 3.5 Hình ảnh giao diện chiến thắng trong trò chơi 17](#_Toc198749277)

# GIỚI THIỆU

## Tên đề tài và chủ đề

Tên đề tài: Ứng dụng quản lý lịch trình cá nhân

Chủ đề nghiên cứu: Phát triển một ứng dụng web nhằm hỗ trợ người dùng trong việc quản lý và theo dõi lịch trình cá nhân với các chế độ hiển thị linh hoạt theo ngày, tuần và tháng. Hệ thống cho phép thực hiện các thao tác nghiệp vụ cơ bản như tạo mới, chỉnh sửa, xoá sự kiện, đồng thời đảm bảo tính nhất quán và liên tục của dữ liệu thông qua cơ chế đồng bộ với máy chủ thông qua giao tiếp API.

Trong quá trình xây dựng hệ thống, đề tài ứng dụng các công nghệ và công cụ tiêu biểu trong lĩnh vực công nghệ phần mềm hiện nay, cụ thể như sau:

* Thiết kế giao diện người dùng (Frontend): Xây dựng giao diện trực quan, tối ưu trải nghiệm người dùng, hỗ trợ tương tác lịch trình thông qua trình duyệt web, sử dụng các công nghệ như HTML, CSS và JavaScript.
* Phát triển hệ thống API theo kiến trúc RESTful (Backend): Thiết kế và triển khai các điểm cuối (endpoint) phục vụ cho việc lưu trữ, truy xuất và xử lý dữ liệu lịch trình một cách hiệu quả.
* Quản lý mã nguồn với GitHub: Đảm bảo kiểm soát phiên bản, đồng bộ hóa tiến độ phát triển và hỗ trợ làm việc nhóm trong môi trường phân tán.

Ứng dụng các công cụ hỗ trợ phát triển phần mềm, bao gồm:

* Jira: Quản lý tiến độ và công việc theo mô hình Agile, hỗ trợ phân chia nhiệm vụ và theo dõi tiến trình phát triển.
* Docker: Tạo lập môi trường phát triển và triển khai nhất quán, đảm bảo tính đồng bộ giữa các môi trường hệ thống.
* Postman: Hỗ trợ kiểm thử và mô phỏng các truy vấn đến API một cách trực quan và hiệu quả.
* Swagger: Tài liệu hóa API và cung cấp giao diện kiểm thử tự động, nâng cao khả năng kiểm tra và bảo trì hệ thống.

## Mục tiêu của ứng dụng

Mục tiêu chính của đề tài là xây dựng một hệ thống ứng dụng web cho phép người dùng quản lý lịch trình cá nhân một cách hiệu quả, trực quan và có khả năng mở rộng. Ứng dụng đáp ứng các chức năng cốt lõi bao gồm:

* Hiển thị lịch trình theo chế độ ngày, tuần, tháng, hỗ trợ quan sát tổng thể các sự kiện đã được thiết lập.
* Cung cấp khả năng tạo mới, chỉnh sửa, xoá sự kiện tương ứng với từng ngày cụ thể trên giao diện lịch.
* Hiển thị chi tiết nội dung của các sự kiện khi người dùng tương tác với từng ô ngày.
* Tự động cập nhật và hiển thị lại giao diện khi có sự thay đổi dữ liệu.
* Đồng bộ hoá dữ liệu với máy chủ thông qua hệ thống API, đảm bảo tính nhất quán, an toàn và khả năng lưu trữ lâu dài.

## Lí do chọn đề tài

Quản lý lịch trình là một nhu cầu phổ biến trong nhiều lĩnh vực như giáo dục, hành chính, doanh nghiệp và đời sống cá nhân. Việc ghi nhớ và sắp xếp thời gian một cách hợp lý đóng vai trò quan trọng trong việc nâng cao hiệu suất làm việc và tổ chức công việc khoa học. Tuy nhiên, phần lớn các giải pháp hiện tại thường tích hợp trong các hệ sinh thái phức tạp, yêu cầu tài khoản đồng bộ, hoặc không phù hợp với các tình huống sử dụng độc lập, đơn giản.

Với mục tiêu xây dựng một ứng dụng có giao diện trực quan, chức năng tập trung, dễ sử dụng và dễ triển khai, đề tài hướng đến việc giải quyết một cách cụ thể bài toán quản lý lịch trình cá nhân theo cách tiếp cận nhẹ, hiệu quả và khả thi.

Đồng thời, việc lựa chọn đề tài còn nhằm hiện thực hóa một hệ thống phần mềm hoàn chỉnh theo đúng các giai đoạn của quy trình phát triển phần mềm hiện đại, từ đặc tả yêu cầu, thiết kế, lập trình đến kiểm thử và triển khai. Đây là một trong những hướng tiếp cận phù hợp để đánh giá khả năng áp dụng lý thuyết vào thực tiễn cũng như kiểm chứng các nguyên tắc thiết kế phần mềm thông qua sản phẩm cụ thể.

# PHÂN TÍCH YÊU CẦU

## Các chức năng chính của hệ thống (Functional Requirements).

Các yêu cầu chức năng mô tả các hành vi, tác vụ mà hệ thống cần thực hiện nhằm đáp ứng các mục tiêu sử dụng của người dùng. Đối với ứng dụng quản lý lịch trình cá nhân, các chức năng chính bao gồm:

### Đăng ký tài khoản người dùng

* Cho phép người dùng tạo tài khoản bằng tên, số điện thoại, mật khẩu và các thông tin cơ bản.
* Kiểm tra tính hợp lệ của thông tin đầu vào (số điện thoại hợp lệ).
* Thông báo cho người dùng khi đăng ký thành công hoặc khi xảy ra lỗi.

### Đăng nhập và xác thực người dùng

* Cho phép người dùng đăng nhập bằng tên đăng nhập và mật khẩu đã đăng ký.
* Xác thực thông tin đăng nhập qua hệ thống backend.
* Cung cấp chức năng lưu phiên đăng nhập và đăng xuất an toàn.

### Khôi phục mật khẩu

* Cung cấp chức năng “Quên mật khẩu” cho người dùng nhập số điện thoại để khôi phục tài khoản.
* Gửi mã xác nhận hoặc liên kết đặt lại mật khẩu qua số điện thoại.

### Quản lý lịch trình cá nhân

* **Thêm sự kiện:** Cho phép người dùng thêm mới sự kiện với các thông tin như tiêu đề, mô tả, ngày, giờ.
* **Hiển thị lịch:** Giao diện lịch tháng trực quan với khả năng hiển thị sự kiện tương ứng với từng ngày.
* **Sửa sự kiện:** Cho phép người dùng chỉnh sửa lại các sự kiện phù hợp với mục đính cá nhân.
* **Xóa sự kiện:** Hỗ trợ người dùng xóa sự kiện không còn cần thiết.

### Quản lý nhiều sự kiện trong cùng một ngày

* Cho phép người dùng thêm nhiều sự kiện vào cùng một ngày.
* Giao diện popup hiện danh sách sự kiện của ngày được chọn.

### Cài đặt tài khoản

* **Hiển thị tên tài khoản:** Cho phép người dùng kiểm tra nhanh thông tin tên tài khoản( username) hiện tại đang được dùng trong ứng dụng.
* **Đổi mật khẩu:** Cho phép người dùng chủ động thay đổi mật khẩu để đảm bảo an toàn bảo mật tài khoản. Việc thay đổi mật khẩu yêu cầu người dùng cung cấp đầy đủ thông tin như mật khẩu hiện tại, mật khẩu mới. Hệ thống thực hiện kiểm tra hợp lệ trước khi cập nhật.
* **Đăng xuất**: Cho phép người dùng kết thúc phiên làm việc hiện tại và quay trở lại giao diện đăng nhập. Việc đăng xuất giúp đảm bảo an toàn thông tin, đặc biệt trong trường hợp người dùng sử dụng thiết bị công cộng hoặc không thuộc sở hữu cá nhân.

### Giao diện người dùng thân thiện

* Hệ thống cung cấp giao diện dạng lịch.
* Các sự kiện hiển thị trực tiếp trong ô ngày tương ứng.
* Giao diện phản hồi nhanh với các thao tác như click vào ngày, chọn sự kiện,...

### Đồng bộ hóa dữ liệu với backend

* Tất cả các thao tác tạo, sửa, xóa, lấy lịch trình đều được xử lý thông qua các API RESTful kết nối với backend.
* Đảm bảo dữ liệu được lưu trữ an toàn và đồng bộ giữa các thiết bị.

## Các yêu cầu phi chức năng (Non-functional Requirements).

Hệ thống Quản lý lịch trình cá nhân cần đáp ứng một số yêu cầu phi chức năng nhằm đảm bảo chất lượng và hiệu quả hoạt động như sau:

* **Hiệu năng (Performance):** Hệ thống đảm bảo khả năng phản hồi nhanh chóng. Thời gian xử lý các thao tác chính như đăng nhập, hiển thị lịch, thêm, sửa, xoá sự kiện không vượt quá 2-3 giây để đảm bảo trải nghiệm người dùng mượt mà.
* **Bảo mật (Security):** Hệ thống đảm bảo an toàn cho dữ liệu người dùng. Thông tin tài khoản như tên đăng nhập, mật khẩu được mã hoá khi lưu trữ. Đồng thời, hệ thống phải có cơ chế xác thực, đảm bảo mỗi người chỉ truy cập được dữ liệu lịch trình cá nhân của chính họ.
* **Khả năng mở rộng (Scalability):** Hệ thống được xây dựng theo mô hình RESTful API tách biệt frontend và backend, thuận lợi cho việc mở rộng chức năng hoặc tích hợp với các hệ thống khác trong tương lai.
* **Tính dễ sử dụng (Usability):** Giao diện người dùng thân thiện, trực quan, dễ sử dụng với người không am hiểu công nghệ. Các thao tác như thêm sự kiện, chỉnh sửa, xoá hay chuyển đổi các tháng rõ ràng, dễ thực hiện. Hệ thống cung cấp thông báo, cảnh báo hoặc xác nhận khi người dùng thao tác.
* **Khả năng hoạt động ổn định (Reliability):** Hệ thống đảm bảo hoạt động liên tục, hạn chế lỗi phát sinh trong quá trình sử dụng. Các chức năng quan trọng như lưu dữ liệu, hiển thị lịch phải luôn sẵn sàng và chính xác để người dùng có thể sử dụng mọi lúc, mọi nơi.
* **Khả năng bảo trì (Maintainability):** Mã nguồn cần được tổ chức rõ ràng, dễ đọc, dễ bảo trì. Các thành phần như giao diện người dùng, API được thiết kế riêng biệt giúp thuận lợi trong việc sửa lỗi, nâng cấp hoặc thay đổi công nghệ trong tương lai.

# THIẾT KẾ HỆ THỐNG

## Kiến trúc tổng thể

### Mô hình kiến trúc hệ thống

Hệ thống được xây dựng dựa trên mô hình Client - Server kết hợp RESTful API nhằm đảm bảo sự phân tách rõ ràng giữa phần giao diện người dùng (Frontend) và phần xử lý nghiệp vụ, truy xuất dữ liệu (Backend). Hệ thống sử dụng cơ sở dữ liệu MongoDB để lưu trữ các thông tin liên quan đến tài khoản người dùng và lịch trình sự kiện.

Mô hình tổng thể bao gồm các thành phần chính sau:

* Frontend (Client) - Giao diện người dùng:

+ Được xây dựng bằng các công nghệ HTML, CSS (Tailwind CSS), JavaScript.

+ Giao diện cho phép người dùng thao tác trực tiếp với hệ thống: đăng ký, đăng nhập, tạo mới, chỉnh sửa, xoá sự kiện, và quản lý lịch trình cá nhân.

+ Kết nối tới Backend thông qua API HTTP.

* Backend (Server API) - Xử lý nghiệp vụ:

+ Sử dụng Node.js và Express.js làm nền tảng chính.

+ Áp dụng kiến trúc MVC (Model - View - Controller) giúp quản lý mã nguồn rõ ràng, dễ bảo trì.

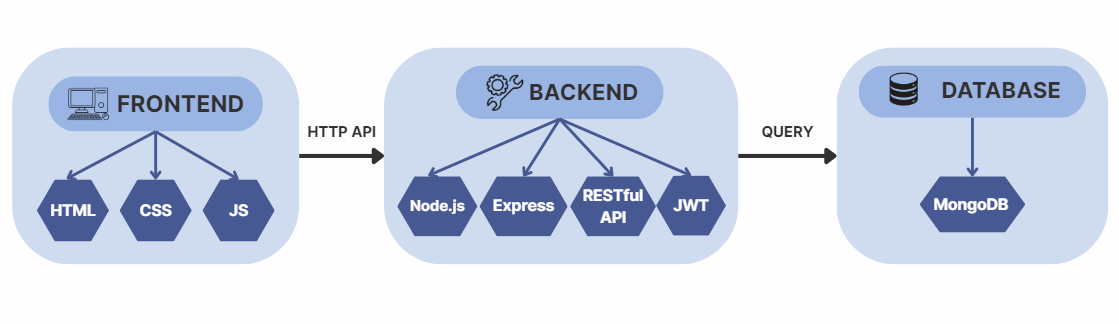
+ Xây dựng các API dạng RESTful cho phép frontend gửi request và nhận response dưới dạng JSON.

+ Bảo mật xác thực người dùng bằng JWT (JSON Web Token).

* Database (Cơ sở dữ liệu):

Lưu trữ dữ liệu thông qua MongoDB, một hệ quản trị cơ sở dữ liệu NoSQL linh hoạt, phù hợp cho việc lưu trữ lịch trình và tài khoản người dùng.

### Sơ đồ kiến trúc hệ thống

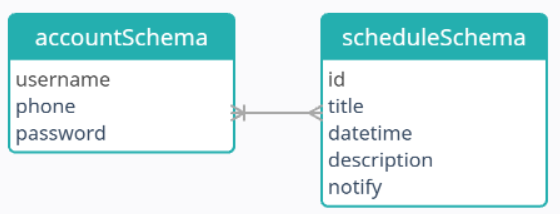


Hình 3.1 Sơ đồ kiến trúc hệ thống

Mô tả hoạt động:

* Người dùng thao tác trên trình duyệt thông qua các giao diện như đăng nhập, đăng ký, quản lý lịch.
* Frontend gửi các yêu cầu HTTP tới Backend.
* Backend tiếp nhận request, xử lý, truy xuất dữ liệu từ Database và trả về kết quả JSON.
* Database quản lý dữ liệu lịch trình, tài khoản, sự kiện.

## Thiết kế cơ sở dữ liệu



Hình 3.2 Mô hình ERD

## Thiết kế API

### Các endpoint chính

Bảng 3.1 Các endpoint chính

|  |  |  |
| --- | --- | --- |
| **Method** | **Endpoint** | **Chức năng** |
| POST | /auth/register | Đăng ký tài khoản |
| POST | /auth/login | Đăng nhập |
| POST | /accounts | Tạo tài khoản |
| GET | /accounts/:username | Lấy thông tin tài khoản |
| DELETE | /accounts/:username | Xóa tài khoản |
| PUT | /accounts/change-password | Đổi mật khẩu người dùng |
| POST | /accounts/:username/schedules | Thêm lịch trình mới |
| DELETE | /accounts/:username/schedules/:id | Xóa lịch trình |
| PATCH | /accounts/:username/schedules/:id | Cập nhật lịch trình |

### Cấu trúc request/response

|  |  |  |
| --- | --- | --- |
| **API** | **Request** | **Response** |
| POST /auth/register | {  "username": "string",  "phone": "string",  "password": "string"  } | 201: User đăng ký thành công  400: Lỗi đăng ký |
| POST /auth/login | {  "username": "string",  "password": "string"  } | Token  401: Lỗi đăng nhập |
| POST /accounts | {  "username": "string",  "phone": "string",  "password": "string"  } | 201: Tạo tài khoản thành công  400: Lỗi tạo tài khoản |
| GET /accounts/:username |  | 200: Thông tin tài khoản  404: Không tìm thấy tài khoản |
| DELETE /accounts/:username |  | 204: Xóa thành công  404: Không tìm thấy tài khoản |
| PUT /accounts/change-password | {  "currentPassword": "string",  "newPassword": "string"  } | 200: Đổi mật khẩu thành công  400: Lỗi đổi mật khẩu  401: Chưa đăng nhập hoặc token không hợp lệ |
| POST /accounts/:username/schedules | {  "title": "string",  "datetime": "YYYY-MM-DDThh:mm:ss.xxxx",  "description": "string"  } | 201: Thêm lịch trình thành công  400: Lỗi thêm lịch trình |
| DELETE /accounts/:username/schedules/:id |  | 204: Xóa lịch trình thành công  404 Không tìm thấy lịch trình |
| PATCH /accounts/:username/schedules/:id | {  "title": "string",  "datetime": "YYYY-MM-DDThh:mm:ss.xxxx",  "description": "string"  } | 200: Cập nhật thành công  400: Lỗi cập nhật lịch trình |

## Thiết kế giao diện (UI/UX)

# TRIỂN KHAI VÀ cÔNG NGHỆ SỬ DỤNG

## Các công nghệ sử dụng

Bảng 4.1 Danh sách các công nghệ sử dụng

|  |  |  |
| --- | --- | --- |
| **Thành phần** | **Công nghệ** | **Đặc điểm** |
| **Backend** | Node.js | Môi trường chạy JavaScript phía server. |
| express | Framework chính để tạo API RESTful nhanh gọn và linh hoạt. |
| dotenv | Thư viện giúp tải các biến môi trường từ file .env vào process.env trong Node.js. |
| cors | Middleware trong Node.js cho phép xử lý CORS (Cross-Origin Resource Sharing) cho frontend gọi API |
| body-parser | Middleware trong Node.js dùng với Express để phân tích dữ liệu body của request, giúp dễ dàng truy cập nội dung gửi từ client. |
| jsonwebtoken | Còn gọi là JWT, một thư viện trong Node.js dùng để tạo và xác thực token, hash mật khẩu người dùng trước khi lưu vào CSDL, đảm bảo bảo mật. |
| swagger-ui-express | Tích hợp Swagger UI để mô tả và thử nghiệm API thông qua giao diện web. |
| swagger-jsdoc | Công cụ giúp tạo file Swagger spec (OpenAPI) từ các comment trong mã nguồn JavaScript. |
| **Fontend** | HTML/CSS/JavaScript | Giao diện nhẹ, đơn giản, dễ tùy biến. |
| **Cơ sở dữ liệu** | MongoDB | Hệ quản trị cơ sở dữ liệu NoSQL. |
| Mongoose | Thư viện ODM (Object Data Modeling) cho Node.js dùng để giao tiếp với MongoDB, hỗ trợ schema và mô hình hóa dữ liệu. |
| **Kiểm thử** | Jest | Framework kiểm thử phổ biến được phát triển bởi Meta, chủ yếu dùng để kiểm thử JavaScript và Node.js. |
| Supertest | Thư viện trong Node.js dùng để kiểm thử các API HTTP (RESTful API) của ứng dụng Express hoặc các ứng dụng web khác một cách dễ dàng và chính xác. |
| Postman | Công cụ hỗ trợ phát triển và kiểm thử API rất phổ biến hiện nay. Nó giúp bạn dễ dàng gửi yêu cầu HTTP đến máy chủ và xem phản hồi mà API trả về. |

## Quy trình KẾT LUẬN

Trong quá trình thực hiện đề tài “Xây dựng trò chơi xếp hình”, nhóm đã tiến hành đầy đủ các bước từ phân tích bài toán, thiết kế giao diện người dùng, lập trình logic trò chơi, đến việc tích hợp thuật toán tìm kiếm A\* nhằm giải quyết bài toán tìm đường tối ưu trong không gian trạng thái hữu hạn. Kết quả thu được cho thấy mô hình đã hoạt động ổn định, đáp ứng các yêu cầu đặt ra cả về mặt chức năng lẫn khả năng minh họa cho ứng dụng trí tuệ nhân tạo trong thực tiễn.

Cụ thể, sản phẩm hoàn chỉnh gồm các thành phần chính: giao diện trực quan, cơ chế xáo trộn hình ảnh đảm bảo tính khả giải, chức năng điều khiển bằng thao tác kéo thả các mảnh ghép, xác định trạng thái mục tiêu, cùng khả năng tự động giải bài toán sắp xếp bằng thuật toán A\*. Việc triển khai thuật toán A\* đã được thực hiện hiệu quả với hàm đánh giá f(n) = g(n) + h(n), trong đó g(n) là chi phí từ trạng thái ban đầu đến trạng thái hiện tại, còn h(n) là chi phí ước lượng từ trạng thái hiện tại đến trạng thái đích. Kết quả mô phỏng cho thấy thuật toán tìm được lời giải tối ưu với số bước di chuyển hợp lý.

Thông qua quá trình xây dựng và thử nghiệm trò chơi, nhóm đã rút ra nhiều bài học giá trị. Về mặt kỹ thuật, đề tài giúp củng cố kiến thức về cấu trúc dữ liệu, giải thuật tìm kiếm, thiết kế hệ thống tương tác và lập trình hướng đối tượng. Về mặt lý thuyết, đề tài giúp làm rõ cách thức hoạt động của các thuật toán heuristic và vai trò của chúng trong việc tối ưu hóa quá trình tìm kiếm lời giải trong các không gian trạng thái lớn.

Đáng chú ý, đề tài đã chứng minh được tính khả thi và giá trị ứng dụng của AI trong các hệ thống đơn giản như trò chơi giải đố. Sự kết hợp giữa yếu tố giải trí và yếu tố trí tuệ nhân tạo không chỉ tăng tính hấp dẫn của sản phẩm, mà còn mang lại tiềm năng lớn trong việc giáo dục tư duy thuật toán cho người học. Trò chơi không chỉ dừng lại ở mục đích giải trí, mà còn có thể được sử dụng như một công cụ trực quan hỗ trợ giảng dạy các môn học như tin học, trí tuệ nhân tạo cơ bản, hoặc lý thuyết tìm kiếm.

Trong tương lai, nhóm đề xuất một số hướng mở rộng như: tăng kích thước lưới trò chơi (4x4, 5x5), thêm các chế độ chơi nâng cao như đua thời gian, tích hợp hệ thống xếp hạng, và triển khai trên nền tảng web hoặc thiết bị di động để nâng cao khả năng tiếp cận của sản phẩm. Những phát triển này sẽ tiếp tục khai thác hiệu quả việc ứng dụng AI vào các mô hình trò chơi thông minh, góp phần làm phong phú hơn lĩnh vực giao thoa giữa giáo dục, giải trí và công nghệ.

# DANH MỤC TÀI LIỆU THAM KHẢO

[1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Pearson, 2010.

[2] Unity Technologies, *Unity - Real-time Development Platform*, 2024. [Online]. Available: [https://unity.com/](https://unity.com/" \t "_new).

[3] “A\* search algorithm,” *Wikipedia*, [Online]. Available: [https://en.wikipedia.org/wiki/A\*\_search\_algorithm](https://en.wikipedia.org/wiki/A*_search_algorithm" \t "_new). [Accessed: 18-May-2025].

[4] Microsoft Docs, “C# Programming Guide,” Microsoft, 2024. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/csharp/