# **I.MỤC LỤC**

[**I.MỤC LỤC** 1](#_Toc199723768)

[**II. LỜI CẢM ƠN** 3](#_Toc199723769)

[**III. NỘI DUNG BÀI TẬP LỚN** 4](#_Toc199723770)

[**CHƯƠNG 1. GIỚI THIỆU ĐẦU BÀI** 4](#_Toc199723771)

[**1.1. Thông tin đề tài** 4](#_Toc199723772)

[**1.2. Phân tích bài toán** 4](#_Toc199723773)

[**1.3. Yêu cầu chức năng** 4](#_Toc199723774)

[**1.3.1. Chức Năng Giao Diện và Trải Nghiệm Người Dùng** 4](#_Toc199723775)

[**1.3.2. Chức năng Game play** 5](#_Toc199723776)

[**1.3.3. Chức năng backend và quản lý dữ liệu** 5](#_Toc199723777)

[**CHƯƠNG 2.CƠ SỞ LÝ THUYẾT** 6](#_Toc199723778)

[**2.1.** **Giới thiệu công nghệ nền tảng** 6](#_Toc199723779)

[**2.1.1.** **Web frontend** 6](#_Toc199723780)

[**2.1.2.** **Phaser framework** 6](#_Toc199723781)

[**2.1.3.** **Server backend** 7](#_Toc199723782)

[**2.1.4.** **Quy trình giao tiếp** 7](#_Toc199723783)

[**2.1.5.** **Lưu trữ dữ liệu và xử lý bảng xếp hạng** 7](#_Toc199723784)

[**2.1.6.** **Vấn đề về bảo mật** 7](#_Toc199723785)

[**CHƯƠNG 3.THIẾT KẾ VÀ XÂY DỰNG CHƯƠNG TRÌNH** 9](#_Toc199723786)

[**3.1.Kiến trúc chương trình** 9](#_Toc199723787)

[**3.1.1. Front end** 9](#_Toc199723788)

[**3.1.2. Back end** 9](#_Toc199723789)

[**3.1.3. Cơ sở dữ liệu** 9](#_Toc199723790)

[**3.1.3. Luồng Dữ Liệu và Tương Tác** 9](#_Toc199723791)

[**3.2. Đánh giá kiến trúc chương trình** 10](#_Toc199723792)

[Hình 1. Sơ đồ luồng dữ liệu 10](#_Toc199723793)

[**3.3.Sơ đồ UML** 10](#_Toc199723794)

[**3.3.1.Sơ đồ ca sử dụng** 10](#_Toc199723795)

[Hình 1.Sơ đồ ca sử dụng 11](#_Toc199723796)

[**3.3.2.Sơ đồ lớp** 12](#_Toc199723797)

[Hình 2.Sơ đồ lớp 13](#_Toc199723798)

[**Chương 4.Thực nghiệm và kết luận** 14](#_Toc199723799)

[***4.1.Kết quả thực nghiệm*** 14](#_Toc199723800)

[Hình 3.Giao diện đồ họa 14](#_Toc199723801)

[***4.2.Đánh giá và cải tiến*** 14](#_Toc199723802)

[**III.TÀI LIỆU THAM KHẢO** 15](#_Toc199723803)

# **II. LỜI CẢM ƠN**

Trước hết, em xin gửi lời cảm ơn chân thành đến cô giáo **Nguyễn Thị Hương**, người đã tận tình hướng dẫn và hỗ trợ em trong suốt quá trình thực hiện bài tập lớn này. Những đóng góp quý báu và sự chỉ dẫn tận tâm của thầy đã giúp em hoàn thành đề tài một cách hiệu quả và có định hướng rõ ràng

Em cũng xin cảm ơn các thầy cô trong Khoa Điện Tử, Bộ môn Công Nghệ Thông Tin đã tạo điều kiện thuận lợi để em được học tập, thực hành và phát triển kỹ năng lập trình qua đề tài thực tế

Cuối cùng, em xin cảm ơn gia đình và bạn bè đã luôn cổ vũ, động viên tinh thần em trong suốt quá trình học tập và hoàn thành đồ án

Thái Nguyên, ngày….tháng…..năm 2025

**SINH VIÊN**

**Hà Huy Nam**

**Phạm Thị Hồng Nhung**

*(Ký ghi rõ họ tên)*

# **III. NỘI DUNG BÀI TẬP LỚN**

## **CHƯƠNG 1. GIỚI THIỆU ĐẦU BÀI**

## **1.1. Thông tin đề tài**

- Tên đề tài: “Game vượt chướng ngại vật”

- Mục tiêu: Đề tài này hướng đến việc phát triển một trò chơi chạy trên trình duyệt sử dụng HTML5 và PhaserJS

## **1.2. Phân tích bài toán**

Trò chơi là một tựa game "vượt chướng ngại vật" chạy trên trình duyệt, trong đó nhân vật chính sẽ tự động chạy liên tục cùng với sự xuất hiện ngẫu nhiên của các chướng ngại vật

Người chơi tương tác bằng cách thực hiện thao tác (nhấn phím, thao tác cảm ứng) để khiến nhân vật nhảy và tránh va chạm, từ đó ghi điểm dựa trên khoảng cách vượt qua hoặc số lượng chướng ngại vật đã tránh được

Game hướng đến việc tạo ra một trải nghiệm chơi mượt mà, hấp dẫn và đầy thử thách bằng các hiệu ứng đồ họa sống động nhờ vào công nghệ HTML5 và PhaserJS

## **1.3. Yêu cầu chức năng**

Phần này trình bày chi tiết các chức năng mà game cần triển khai nhằm đảm bảo trải nghiệm người dùng mượt mà, gameplay hấp dẫn đồng thời tích hợp được các tính năng backend cần thiết để quản lý dữ liệu người chơi. Các chức năng có thể được phân loại theo từng nhóm như chức năng giao diện (frontend), chức năng gameplay và chức năng quản lý dữ liệu (backend), hoặc cụ thể hơn như sau:

## **1.3.1. Chức Năng Giao Diện và Trải Nghiệm Người Dùng**

- Màn hình menu chính: hiển thị các lựa chọn cơ bản: “Chơi”, “Bảng Xếp Hạng”, “Hướng Dẫn”,... để người chơi dễ dàng tiếp cận các chế độ chơi. Thiết lập giao diện bắt mắt với các animation chuyển cảnh nhẹ nhàng khi di chuyển giữa các lựa chọn

- Giao diện chơi game: render nhân vật chính, nền chạy tự động và các chướng ngại vật trên trình duyệt bằng Canvas của HTML5 thông qua PhaserJS. Hiển thị thông tin trực quan như điểm số hiện tại, cấp độ (nếu áp dụng) và thời gian chơi. Cung cấp hiệu ứng âm thanh và hình ảnh (sprite animation) nhằm tăng tính thu hút, giảm cảm giác nhàm chán trong quá trình chơi

- Màn hình kết thúc và bảng xếp hạng: sau khi kết thúc game (khi phát hiện va chạm), hiển thị thông báo “Game Over” kèm theo điểm số đạt được. Cho phép người chơi xem bảng xếp hạng với danh sách những người chơi có điểm số cao nhất và tùy chọn chơi lại hoặc trở về menu chính

## **1.3.2. Chức năng Game play**

- Nhân vật và điều khiển: nhân vật chính chạy tự động; người chơi chỉ cần thao tác (nhấn phím) để kích hoạt hành động “nhảy”. Xử lý logic nhảy của nhân vật một cách mượt mà, đảm bảo phản ứng nhanh khi người chơi muốn tránh chướng ngại vật

- Quản lý chướng ngại vật: phát sinh chướng ngại vật ngẫu nhiên với khoảng cách và tốc độ biến đổi theo thời gian, tạo ra độ khó tăng dần. Xử lý cơ chế va chạm giữa nhân vật và các chướng ngại vật: Khi phát hiện va chạm, game sẽ chuyển sang trạng thái kết thúc - “Game Over”

- Tính điểm và nâng cấp độ: tính điểm dựa trên số chướng ngại vật vượt qua hoặc khoảng cách chạy được của nhân vật. Đưa ra các mốc điểm nhất định để nâng cấp cấp độ, từ đó tăng tốc độ hoặc số lượng chướng ngại vật nhằm duy trì tính thử thách của trò chơi. Hiển thị liên tục điểm số và cấp độ hiện tại trên giao diện trong quá trình chơi

## **1.3.3. Chức năng backend và quản lý dữ liệu**

- Quản lý hồ sơ người chơi: Lưu trữ thông tin người chơi như: UID, username, điểm số cao nhất và cấp độ. Xây dựng API, ví dụ để nhận và xử lý dữ liệu người chơi khi game kết thúc

- Bảng xếp hạng: tạo endpoint để truy vấn thứ tự người chơi theo điểm số giảm dần. Cập nhật bảng xếp hạng trên giao diện game theo thời gian thực khi có sự thay đổi dữ liệu

- Bảo mật, kiểm thử dữ liệu: đảm bảo dữ liệu được gửi từ phía client lên server được kiểm chứng, hạn chế gian lận, thao tác chỉnh sửa điểm số không hợp lệ. Có thể sử dụng các biện pháp

## **CHƯƠNG 2.CƠ SỞ LÝ THUYẾT**

## **2.1.** **Giới thiệu công nghệ nền tảng**

Chương cơ sở lý thuyết giúp giới thiệu, trình bày nguyên lý của các công cụ, công nghệ làm game phổ biến hiện nay, dựa vào đó để phân tích, lựa chọn công cụ, công nghệ, hướng làm phù hợp với yêu cầu đề tài

## **2.1.1.** **Web frontend**

- HTML5: HTML5 là nền tảng cơ bản cho các ứng dụng web hiện đại. Đặc biệt, thẻ <canvas> cho phép render đồ họa 2D trực tiếp trên trình duyệt mà không cần đến plugin bên ngoài

- Canvas: sử dụng <canvas> để hiển thị đồ họa, vẽ các sprite, hiệu ứng chuyển động và thiết lập background cho game. Đây là nền tảng giúp game hoạt động mượt mà và có khả năng xử lý các hoạt cảnh động phức tạp

- JavaScript Vanilla: JavaScript thuần là ngôn ngữ cốt lõi để xử lý các logic, tương tác người dùng và xử lý thao tác DOM. Bên cạnh việc dùng để thiết lập và điều khiển game engine, JavaScript được sử dụng để quản lý các sự kiện, giao tiếp với các API backend và điều chỉnh các hiệu ứng đặc thù, giúp game trở nên sống động hơn

## **2.1.2.** **Phaser framework**

PhaserJS là một trong những framework mạnh mẽ cho việc phát triển game HTML5. Nó cung cấp các công cụ và module cần thiết để xử lý animation, vật lý game, quản lý scene và các hiệu ứng chuyển cảnh

Cấu trúc và thành phần chính:

- Các phương thức cơ bản:

+ Preload: Dùng để tải các tài nguyên hình ảnh, âm thanh, sprite sheet trước khi game bắt đầu

+ Create: Khởi tạo các đối tượng trong game như nhân vật, chướng ngại vật, nền, và thiết lập các thông số vật lý hoặc hiệu ứng cần thiết

+ Update: Vòng lặp game chính để cập nhật trạng thái của các đối tượng, xử lý logic va chạm, điều khiển chuyển động của nhân vật và phát sinh các sự kiện

Ưu điểm: tích hợp sẵn các tính năng hỗ trợ vật lý, collision, animation,... giúp quá trình phát triển game trở nên nhanh chóng và hiệu quả

## **2.1.3.** **Server backend**

- NodeJS: cho phép chạy javascript phía server, là nền tảng tốt cho các ứng dụng real time

- ExpressJS: là framework cho NodeJS, giúp xây dựng API rõ ràng, dễ bảo trì và mở rộng. Chức năng chính là tiếp nhận các yêu cầu từ phía client – như cập nhật điểm số, truy vấn dữ liệu trong cơ sở dữ liệu,... một cách nhanh chóng

## **2.1.4.** **Quy trình giao tiếp**

- Endpoint API: xây dựng các API như cập nhật điểm số người chơi sau mỗi phiên chơi, truy xuất thông tin bảng xếp hạng từ cơ sở dữ liệu. Sự phân chia giữa frontend và backend giúp đảm bảo tính bảo mật cho dữ liệu, đồng thời tạo nên một hệ thống mở rộng dễ bảo trì và hiệu quả trong xử lý các yêu cầu đồng thời từ nhiều client

## **2.1.5.** **Lưu trữ dữ liệu và xử lý bảng xếp hạng**

- Cơ sở dữ liệu: Có thể sử dụng các giải pháp NoSQL trong đó có Firebase Firestore nhằm lưu trữ thông tin người chơi một cách linh hoạt. Cấu trúc dữ liệu tổng thể gồm:

+ Thông tin người chơi: chứa các trường như mã người chơi -UID, tên người chơi, điểm số cao nhất và cấp độ

+ Dữ liệu trò chơi: lưu các thông tin về trò chơi như thời gian chơi, log, cấu hình cài đặt, bản cập nhật,... cho các mục đích cần thiết

- Bảng xếp hạng: sử dụng truy vấn dữ liệu, sắp xếp theo thứ tự giảm dần của điểm số, từ đó hiển thị web chức năng bảng xếp hạng cạnh tranh cho người chơi. Các endpoint của expressJS sẽ đảm nhiệm việc cập nhật và cung cấp dữ liệu bảng xếp hạng cho phía client, đảm bảo thông tin luôn được đồng bộ nhanh chóng và chính xác

## **2.1.6.** **Vấn đề về bảo mật**

Do tính chất mở của game chạy trên browser, cần có các biện pháp xác thực và kiểm tra dữ liệu nhằm ngăn chặn gian lận (ví dụ: gửi điểm số không hợp lệ). Có thể làm tích hợp các phương pháp xác thực người dùng như JSON Web Token và thiết lập các quy tắc bảo mật trên API để bảo vệ dữ liệu người chơi

## **CHƯƠNG 3.THIẾT KẾ VÀ XÂY DỰNG CHƯƠNG TRÌNH**

## **3.1.Kiến trúc chương trình**

Chương này đúc kết quá trình nghiên cứu, tìm hiểu về các công cụ, công nghệ đã được gợi ý để đưa đến lựa chọn quyết định để xây dựng kiến trúc chương trình và các chức năng trong trò chơi

Kiến trúc chương trình của game được xây dựng theo mô hình phân tầng có tính module cao, bao gồm các thành phần chính sau:

## **3.1.1. Front end**

- HTML5 Canvas: Sử dụng thẻ <canvas> để render đồ họa 2D của trò chơi

- PhaserJS: Engine chính xử lý animation, vật lý game, quản lý các scene và cập nhật vòng lặp game

- JavaScript Vanilla: Xử lý logic game bổ trợ, sự kiện người dùng và tương tác với DOM khi cần thiết

## **3.1.2. Back end**

- Node.js, ExpressJS: Xây dựng API server để nhận và xử lý các yêu cầu từ phía client

## **3.1.3. Cơ sở dữ liệu**

Sử dụng hệ quản trị dữ liệu NoSQL: lựa chọn Firebase

## **3.1.3. Luồng Dữ Liệu và Tương Tác**

Về tổng thể:

+ Phía Client (Front-end): Người chơi tương tác với game nhấn phím hoặc để điều khiển nhân vật. PhaserJS xử lý game loop, cập nhật các đối tượng nhân vật, chướng ngại vật và thông báo điểm số khi game kết thúc. Client có thể gọi thêm API để lấy dữ liệu bảng xếp hạng và hiển thị cho người chơi

+ Giao tiếp với Server (Back-end): Khi game kết thúc, dữ liệu điểm số cùng thông tin người chơi được gửi từ client đến API của ExpressJS. Server xử lý dữ liệu nhận được, so sánh điểm số, cập nhật thông tin vào cơ sở dữ liệu và sau đó trả về thông báo hoặc dữ liệu bảng xếp hạng

## **3.2. Đánh giá kiến trúc chương trình**

- Phân tách rõ ràng: Giúp mỗi module (front-end, back-end, cơ sở dữ liệu) thực hiện nhiệm vụ của mình mà không gây ảnh hưởng lẫn nhau

- Tính mở rộng cao: Khi game phát triển, người ta có thể dễ dàng bổ sung thêm các tính năng (chế độ multiplayer, phân tích dữ liệu, …) mà không cần thay đổi quá trình xử lý dữ liệu hiện hành

- Hiệu suất và bảo mật: Tối ưu hoá vòng lặp game ở phía client, kết hợp với kiểm chứng dữ liệu và bảo mật API ở phía server, đảm bảo trải nghiệm người dùng mượt mà và an toàn

A diagram of a express api server

AI-generated content may be incorrect.

### Hình 1. Sơ đồ luồng dữ liệu

## **3.3.Sơ đồ UML**

Sơ đồ UML (Unified Modeling Language) là một công cụ trực quan giúp mô hình hóa hệ thống phần mềm, giúp các nhà phát triển hiểu rõ cấu trúc, chức năng và luồng dữ liệu của ứng dụng

## **3.3.1.Sơ đồ ca sử dụng**

- Các tác nhân:

+ Người chơi: trực tiếp tương tác với game

+ Hệ thống game: xử lý logic và phản hồi theo thao tác của người chơi

+ Server: quản lý dữ liệu điểm số và bảng xếp hạng

- Các trường hợp sử dụng:

+ Đăng nhập: đăng nhập nhanh có thể qua tài khoản google

+ Bắt đầu trò chơi: người chơi nhấn nút bắt đầu để vào màn chơi

+ Điều khiển nhân vật: người chơi nhấn phím để nhảy vượt chướng ngại vật

+ Xử lý va chạm: kiểm tra xem nhân vật có va chạm chướng ngại vật hay không

+ Tính điểm: khi người chơi vượt qua chướng ngại vật hoặc chạy được một quãng đường nhất định

+ Kết thúc trò chơi: khi nhân vật va vào chướng ngại vật hoặc hết thời gian

+ Hiển thị bảng xếp hạng: người chơi có thể xem danh sách điểm số cao

+ Cập nhật điểm số (lên server): sau mỗi lần chơi, điểm số sẽ được gửi lên server để lưu trữ

+ Tải lại game: Sau khi kết thúc, người chơi có thể chọn để chơi lại

A diagram of a game

AI-generated content may be incorrect.

### Hình 1.Sơ đồ ca sử dụng

## **3.3.2.Sơ đồ lớp**

Sơ đồ lớp giúp mô hình hóa các thành phần chính của trò chơi, các thuộc tính và phương thức của chúng cũng như mối quan hệ giữa các lớp

- Chi tiết các lớp:

+ GameScene: đại diện cho “scene” chính của Phaser, chịu trách nhiệm khởi tạo (preload, create) các đối tượng và cập nhật trạng thái của game trong vòng lặp (update). Các phương thức: phương thức preload() – tải tài nguyên hình ảnh, âm thanh, nhân vật. Phương thức create() - khởi tạo các đối tượng, cài đặt physics và UI. Phương thức update() - cập nhật trạng thái của các đối tượng trong game

+ Player: lớp đại diện cho nhân vật chính của trò chơi. Thuộc tính: sprite – nhân vật. Speed - tốc độ chạy của nhân vật. isJumping - Trạng thái nhảy hiện tại của nhân vật. Các phương thức: jump() - phương thức kích hoạt hành động nhảy, update() - cập nhật vị trí và xử lý trạng thái của nhân vật

+ ObstacleManager: lớp chịu trách nhiệm tạo và quản lý tập hợp các chướng ngại vật trong quá trình chơi. Các thuộc tính: obstacles - danh sách chướng ngại vật hiện có. spawnRate - tốc độ (hoặc khoảng cách) sinh ra chướng ngại vật mới. Các phương thức: spawnObstacle() - sinh ra chướng ngại vật mới, updateObstacles() - cập nhật và xử lý xoá các chướng ngại vật đã qua màn hình

+ ScoreManager: quản lý điểm số của người chơi. Các thuộc tính: currentScore - điểm số hiện tại của phiên chơi. bestScore - điểm số cao nhất (có thể lưu trữ trong cơ sở dữ liệu). Các phương thức: updateScore() - cập nhật điểm số dựa trên hành động của người chơi, resetScore() - đặt lại điểm số khi game khởi động lại

+ UIManager: lớp quản lý các thành phần giao diện hiển thị cho người chơi. Các thuộc tính: scoreText - đối tượng hiển thị điểm số. gameOverScreen - giao diện hiển thị sau khi game kết thúc. menuUI - giao diện menu chính. Các phương thức: updateUI() - cập nhật thông tin UI theo trạng thái game, showGameOver() – h, showMenu() hiển thị màn hình báo hiệu kết thúc trò chơi, showMenu() - hiển thị menu chính

+ APIClient: quản lý các giao tiếp với server qua API. Các thuộc tính: endpointURL - địa chỉ endpoint của server. Các phương thức: sendScore() - gửi điểm số lên server để lưu trữ, getRanking() - lấy bảng xếp hạng từ server

A screenshot of a computer

AI-generated content may be incorrect.

### Hình 2.Sơ đồ lớp

## **Chương 4.Thực nghiệm và kết luận**

## ***4.1.Kết quả thực nghiệm***

-Hình ảnh giao diện chạy game

### Hình 3.Giao diện đồ họa

## ***4.2.Đánh giá và cải tiến***

- Đánh giá chung

+ Ưu điểm: game chạy mượt và tải tài nguyên nhanh. Gameplay ổn định, cơ chế nhảy và va chạm chính xác. Giao diện trực quan, dễ sử dụng. Tích hợp backend hiệu quả cho việc lưu trữ và cập nhật điểm số

+ Hạn chế: độ khó không cân bằng khi game tăng tốc độ và tần số chướng ngại vật. Đôi chỗ đồng bộ dữ liệu và bảo mật API chưa tối ưu. Hiệu ứng giao diện và chuyển cảnh cần được cải tiến để tăng tính hấp dẫn

- Hướng cải tiến: cập nhật hiệu ứng chuyển cảnh, thông báo và UI để nâng cao trải nghiệm người dung. Tối ưu hóa giao tiếp client-server, giảm độ trễ và sử dụng các biện pháp bảo mật như JWT

# **III.TÀI LIỆU THAM KHẢO**

1. [butadpj/platformer-game-PhaserJS: A simple platformer game using Phaser JS](https://github.com/butadpj/platformer-game-PhaserJS) *https://github.com/butadpj/platformer-game-PhaserJS*