TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI TP. HCM

**VIỆN CÔNG NGHỆ THÔNG TIN VÀ ĐIỆN, ĐIỆN TỬ**

**ĐỒ ÁN MÔN HỌC**

**LẬP TRÌNH MẠNG**

**XÂY DỰNG WEBSITE QUIZ GAME**

GVHD: Bùi Dương Thế

SVTH:

MSSV\_Nguyễn Hà Phúc

MSSV\_Đàm Ngọc Khoa

035205000214\_Lê Nguyễn Thành Phát

051205010011\_Phạm Xuân Hòa

**MỤC LỤC**

**1. Giới thiệu 2**

**2. Giới hạn 3**

**3. Mục tiêu 4**

**4. Mục đích 5**

**5. Lý thuyết 6**

**6. Thực hành/Thực nghiệm 8**

**7. Kinh nghiệm 11**

**8. Kết quả đạt được/chưa đạt được 12**

**9. Hướng phát triển trong tương lai 13**

**10. Kết luận 15**

**Tài liệu tham khảo 16**

**PHẦN 1: GIỚI THIỆU**

Trong lĩnh vực công nghệ thông tin, lập trình mạng đóng vai trò quan trọng trong việc xây dựng các ứng dụng web hiện đại, nơi dữ liệu cần được trao đổi liên tục giữa nhiều thành phần và nhiều người dùng. Một trong những ví dụ tiêu biểu cho ứng dụng lập trình mạng trong thực tế là các trò chơi trực tuyến, đặc biệt là quiz game – hình thức vừa mang tính giải trí, vừa mang tính học tập, giúp người chơi củng cố kiến thức qua các câu hỏi trắc nghiệm.

Hiện nay, các nền tảng nổi tiếng như Kahoot! hay Quizizz đã chứng minh sức hút mạnh mẽ của hình thức quiz game trực tuyến, nơi hàng trăm người chơi có thể tham gia cùng lúc, nhận câu hỏi, gửi đáp án, và xem kết quả ngay trong thời gian thực. Điểm mấu chốt tạo nên thành công của các hệ thống này chính là khả năng giao tiếp dữ liệu qua mạng một cách nhanh chóng, ổn định và chính xác.

Xuất phát từ ý tưởng đó, nhóm chúng em lựa chọn đề tài “Xây dựng Website Quiz Game” với định hướng không chỉ dừng lại ở việc phát triển giao diện quiz cơ bản, mà còn chú trọng đến khía cạnh lập trình mạng. Cụ thể, hệ thống được thiết kế với:

* **Frontend:** Sử dụng ReactJS, HTML, CSS, JavaScript để xây dựng giao diện trực quan, hỗ trợ hiển thị câu hỏi, lựa chọn đáp án và hiển thị kết quả.
* **Backend:** Phát triển bằng Node.js/Express để quản lý dữ liệu, xử lý logic và cung cấp REST API cho phía client.
* **Mạng thời gian thực:** Tích hợp WebSocket để hỗ trợ truyền tải dữ liệu nhanh chóng giữa client và server, phục vụ các chức năng như hiển thị kết quả ngay lập tức, hoặc cập nhật tình trạng người chơi.

Thông qua việc thực hiện đề tài, nhóm kỳ vọng đạt được hai mục tiêu:

1. Xây dựng một sản phẩm web quiz game có thể sử dụng trong học tập và giải trí.
2. Rèn luyện kiến thức và kỹ năng lập trình mạng, bao gồm giao tiếp client–server, xử lý API, truyền dữ liệu thời gian thực, và quản lý đồng bộ nhiều người chơi.

Đây sẽ là cơ hội để nhóm áp dụng những kiến thức đã học vào một dự án thực tế, đồng thời rèn luyện kỹ năng làm việc nhóm, sử dụng GitHub trong quản lý mã nguồn, và triển khai một hệ thống phần mềm từ phân tích đến hiện thực.

**PHẦN 2: GIỚI HẠN**

Để đảm bảo đề tài có thể triển khai hiệu quả trong phạm vi thời gian và kiến thức của nhóm, dự án **“Website Quiz Game”** được giới hạn như sau:

1. **Phạm vi chức năng**

* Hệ thống hỗ trợ người chơi tham gia quiz trực tuyến, trả lời câu hỏi trắc nghiệm với 4 lựa chọn.
* Người dùng nhận kết quả ngay sau khi hoàn thành quiz.
* Backend cung cấp REST API để quản lý câu hỏi, đáp án và điểm số.
* Hệ thống sử dụng WebSocket để truyền dữ liệu thời gian thực, ví dụ: cập nhật điểm hoặc tình trạng quiz cho nhiều người chơi.
* Không triển khai các chức năng nâng cao như đa người chơi với bảng xếp hạng toàn cầu, hoặc tạo quiz tùy chỉnh trực tuyến (có thể phát triển trong tương lai).

1. **Phạm vi công nghệ**

* **Frontend**: ReactJS, HTML, CSS, JavaScript.
* **Backend**: Node.js + Express.
* **Giao tiếp mạng**: REST API cho các thao tác CRUD (Create, Read, Update, Delete) câu hỏi; WebSocket cho truyền dữ liệu thời gian thực.
* **Cơ sở dữ liệu**: dùng cơ sở dữ liệu nhỏ (có thể là MongoDB hoặc JSON tĩnh) để lưu trữ câu hỏi, đáp án, và kết quả người chơi.

1. **Phạm vi người dùng**

* Hệ thống hướng đến người chơi cá nhân hoặc nhóm nhỏ, chủ yếu phục vụ mục đích học tập hoặc giải trí tại lớp/hội nhóm.
* Không phát triển các chức năng quản lý tài khoản người dùng nâng cao (login đa lớp, phân quyền…).

1. **Hạn chế khác**

* Giao diện tập trung trên desktop, chưa tối ưu hoàn toàn cho thiết bị di động, vẫn chưa thể mang đến sự thoải mái cho người dùng
* Không hỗ trợ các hình thức truyền dữ liệu phức tạp như video trực tiếp, livestream.
* Quy mô dự án phù hợp với môn học, nhằm rèn luyện kỹ năng lập trình mạng cơ bản, không hướng tới sản phẩm thương mại.

**PHẦN 3: MỤC TIÊU**

Đề tài “Xây dựng Website Quiz Game” được thực hiện với các mục tiêu chính sau:

**3.1. Mục tiêu tổng quát**

* Xây dựng một hệ thống web quiz game có khả năng hoạt động trực tuyến, hỗ trợ nhiều người chơi tham gia trả lời câu hỏi và nhận kết quả ngay lập tức.
* Áp dụng kiến thức lập trình mạng vào một sản phẩm thực tế, kết hợp giữa frontend (ReactJS) và backend (Node.js/Express, REST API, WebSocket).
* Rèn luyện kỹ năng làm việc nhóm, quản lý mã nguồn và triển khai hệ thống phần mềm trong môi trường học thuật.

**3.2. Mục tiêu cụ thể**

1. **Về chức năng hệ thống**

Phát triển giao diện quiz trực quan bằng ReactJS, cho phép người dùng:

* Chọn quiz.
* Trả lời câu hỏi với 4 lựa chọn.
* Nhận kết quả và điểm số ngay sau khi hoàn thành.

Xây dựng backend bằng Node.js/Express:

* Quản lý dữ liệu câu hỏi và đáp án.
* Cung cấp REST API để giao tiếp với frontend.

Ứng dụng WebSocket để hỗ trợ tương tác thời gian thực, ví dụ: hiển thị tiến độ hoặc điểm của người chơi ngay trong lúc quiz diễn ra.

1. **Về kiến thức và kỹ năng**

Củng cố kiến thức về lập trình mạng, bao gồm:

* Xây dựng REST API.
* Quản lý giao tiếp client–server.
* Ứng dụng WebSocket cho dữ liệu thời gian thực.

Thành thạo hơn trong việc sử dụng ReactJS để phát triển giao diện web động.

Nâng cao kỹ năng làm việc nhóm, phân chia nhiệm vụ, quản lý tiến độ bằng Git/GitHub.

Rèn luyện khả năng phân tích, thiết kế và triển khai một dự án web hoàn chỉnh từ lý thuyết đến thực tế.

1. **Về sản phẩm báo cáo**

Hoàn thiện một sản phẩm demo có thể chạy ổn định trên trình duyệt.

Viết báo cáo tổng hợp (Word) trình bày quá trình phát triển, kết quả đạt được, những hạn chế và hướng phát triển trong tương lai.

**PHẦN 4: MỤC ĐÍCH**

Đề tài “Xây dựng Website Quiz Game” không chỉ nhằm tạo ra một sản phẩm phần mềm đơn thuần, mà còn mang nhiều ý nghĩa thực tiễn và học thuật, cụ thể như sau:

1. **Về học tập và nghiên cứu**

* Giúp nhóm áp dụng kiến thức đã học trong môn Lập trình mạng vào một dự án cụ thể, từ đó hiểu rõ hơn về cách hoạt động của các giao thức truyền dữ liệu trong môi trường web.
* Tạo cơ hội thực hành với các công nghệ hiện đại như ReactJS (xây dựng giao diện người dùng), Node.js/Express (xây dựng backend) và WebSocket (truyền dữ liệu thời gian thực).
* Rèn luyện kỹ năng lập trình theo mô hình client–server, một trong những kiến thức trọng tâm của lập trình mạng.

1. **Về kỹ năng thực tiễn**

* Giúp các thành viên làm quen với quy trình phát triển phần mềm, từ khâu phân tích yêu cầu, thiết kế, cài đặt cho đến kiểm thử và báo cáo.
* Nâng cao kỹ năng làm việc nhóm: phân công công việc, phối hợp phát triển frontend – backend, quản lý mã nguồn chung qua GitHub.
* Cải thiện kỹ năng giải quyết vấn đề và tư duy logic khi xử lý các lỗi phát sinh trong quá trình lập trình mạng và tích hợp hệ thống.

1. **Về ứng dụng thực tế**

* Tạo ra một công cụ giải trí và học tập trực tuyến có thể sử dụng trong các buổi sinh hoạt, học nhóm hoặc lớp học.
* Hỗ trợ người dùng vừa học vừa chơi, giúp củng cố kiến thức theo cách thú vị và dễ tiếp thu hơn.
* Đặt nền tảng để phát triển thêm các tính năng nâng cao như: tạo quiz tùy chỉnh, chơi nhiều người cùng lúc, hoặc kết nối với cơ sở dữ liệu lớn.

Tóm lại, mục đích chính của đề tài là kết hợp việc học kiến thức chuyên môn với việc tạo ra một sản phẩm có tính ứng dụng thực tiễn, qua đó giúp nhóm vừa rèn luyện kỹ năng lập trình mạng, vừa có trải nghiệm phát triển một ứng dụng web hoàn chỉnh.

**PHẦN 5: LÝ THUYẾT**

Để xây dựng và triển khai hệ thống Website Quiz Game, nhóm đã tham khảo và vận dụng nhiều cơ sở lý thuyết liên quan đến lập trình mạng và công nghệ web. Cụ thể:

1. **Kiến thức về lập trình mạng**

* **Mô hình Client–Server:**
  + Đây là mô hình phổ biến trong lập trình mạng, trong đó client (ứng dụng phía người dùng) gửi yêu cầu (request) và server xử lý rồi trả về phản hồi (response).
  + Ứng dụng Quiz Game được thiết kế theo mô hình này, với ReactJS làm client và Node.js/Express đóng vai trò server.
* **Giao thức HTTP/HTTPS:**
  + HTTP là giao thức truyền tải siêu văn bản, nền tảng cho việc giao tiếp giữa trình duyệt và server.
  + Trong hệ thống quiz game, HTTP/HTTPS được sử dụng để gọi REST API từ client tới server nhằm lấy dữ liệu câu hỏi, gửi đáp án và nhận kết quả.
* **REST API (Representational State Transfer – Application Programming Interface):**
  + Là phương pháp xây dựng API sử dụng các phương thức HTTP như GET, POST, PUT, DELETE.
  + REST API trong hệ thống cho phép frontend (ReactJS) trao đổi dữ liệu với backend (Node.js/Express). Ví dụ:
    - GET /questions → Lấy danh sách câu hỏi.
    - POST /answers → Gửi đáp án của người chơi.
    - GET /results → Nhận điểm số.
* **WebSocket:**
  + Là giao thức truyền dữ liệu hai chiều theo thời gian thực giữa client và server.
  + Khác với HTTP chỉ cho phép client chủ động gửi yêu cầu, WebSocket cho phép server chủ động đẩy dữ liệu về client.
  + Trong quiz game, WebSocket được dùng để hiển thị kết quả hoặc tiến độ của người chơi ngay lập tức, thay vì phải tải lại trang.

1. **Kiến thức về phát triển web**

* **HTML (HyperText Markup Language):** Ngôn ngữ đánh dấu để xây dựng cấu trúc giao diện người dùng.
* **CSS (Cascading Style Sheets):** Ngôn ngữ định kiểu, giúp thiết kế giao diện đẹp mắt, dễ sử dụng.
* **JavaScript:** Ngôn ngữ lập trình chạy phía client, tạo sự tương tác động cho website.
* **ReactJS:**
  + Thư viện JavaScript hiện đại dùng để xây dựng giao diện người dùng.
  + ReactJS hoạt động dựa trên khái niệm component (thành phần độc lập) và state (trạng thái ứng dụng).
  + Trong hệ thống quiz game, mỗi câu hỏi được xem là một component, có thể tái sử dụng và quản lý riêng.
* **Node.js và Express:**
  + Node.js là môi trường chạy JavaScript phía server, cho phép xây dựng ứng dụng backend hiệu quả.
  + Express là framework của Node.js, hỗ trợ xây dựng REST API đơn giản và nhanh chóng.
  + Trong dự án này, Express được sử dụng để viết các API phục vụ cho frontend ReactJS.

1. **Lý thuyết về quiz game**

* Quiz game là trò chơi trắc nghiệm, trong đó người chơi trả lời câu hỏi trong một khoảng thời gian nhất định.
* Các yếu tố cơ bản của quiz game:
  1. Ngân hàng câu hỏi: danh sách các câu hỏi và đáp án được lưu trữ trong cơ sở dữ liệu hoặc file JSON.
  2. Cơ chế tính điểm: điểm được cộng khi trả lời đúng, có thể tính thêm bonus khi trả lời nhanh.
  3. Kết quả: hiển thị số điểm, số câu đúng/sai sau khi kết thúc quiz.
  4. Tương tác nhiều người chơi (nếu có): người chơi có thể tham gia cùng lúc, so sánh kết quả qua mạng.

**PHẦN 6: THỰC HÀNH**

Trong quá trình triển khai đề tài Xây dựng Website Quiz Game, nhóm tiến hành thực hành và thử nghiệm trên cả hai phần chính: Frontend (giao diện người dùng) và Backend (xử lý logic và dữ liệu). Ngoài ra, nhóm cũng thực hiện tích hợp WebSocket để đảm bảo khả năng giao tiếp thời gian thực giữa client và server.

1. Môi trường phát triển

* Frontend: ReactJS, HTML, CSS, JavaScript
* Backend: Node.js với Express Framework
* Cơ sở dữ liệu: MongoDB (dùng để lưu thông tin người dùng, câu hỏi và kết quả thi)
* Realtime Communication: Socket.IO (xây dựng trên WebSocket)
* Công cụ hỗ trợ: Visual Studio Code, Postman (test API), GitHub (quản lý mã nguồn)

1. Các bước triển khai
2. Khởi tạo dự án

* Tạo dự án React bằng create-react-app.
* Tạo server backend Node.js với Express.

1. Xây dựng giao diện Quiz Game

* Sử dụng ReactJS để tạo giao diện trực quan, gồm các thành phần chính:
  + Trang đăng nhập/đăng ký cho người dùng.
  + Trang Home: chọn quiz, tạo phòng và tham gia trò chơi.
  + Trang tạo câu hỏi và chỉnh sửa câu hỏi.
  + Trang lịch sử làm bài
  + Trang sảnh phòng quiz game
  + Giao diện làm bài trắc nghiệm (hiển thị câu hỏi, hình ảnh, đáp án lựa chọn, đồng hồ đếm ngược).
  + Trang kết quả, hiển thị điểm số và so sánh với những người chơi khác.

1. Xây dựng Backend cơ sở dữ liệu

* Cấu hình server bằng Express.js để tạo REST API phục vụ cho client.
* Bộ sưu tập questions gồm: question, options, correctAnswer, image.
* Bộ sưu tập users để lưu điểm của người chơi.
* Tạo các API chính:
  + /api/questions: trả về danh sách câu hỏi.
  + /api/users: đăng ký, đăng nhập và quản lý thông tin người chơi.
  + /api/results: lưu kết quả sau khi người chơi hoàn thành quiz.
* Tích hợp MongoDB để lưu trữ dữ liệu.

1. Kết nối thời gian thực

* Sử dụng Socket.IO để hỗ trợ tính năng thi quiz trực tuyến nhiều người chơi.
* Khi một người chơi tham gia, server sẽ thông báo cho những người khác trong phòng.
* Đồng bộ thời gian làm bài và hiển thị kết quả cuối cùng theo real-time.

1. Quá trình thử nghiệm

* Nhóm tiến hành chạy server backend và client frontend trên môi trường cục bộ (localhost).
* Kiểm tra API bằng Postman để đảm bảo dữ liệu trả về đúng định dạng.
* Thử nghiệm giao diện trên nhiều trình duyệt khác nhau (Chrome, Edge).
* Tổ chức một buổi kiểm thử nội bộ với nhiều người cùng đăng nhập để kiểm tra chức năng thời gian thực.

1. Chức năng chính được cài đặt

* Lấy và hiển thị câu hỏi từ cơ sở dữ liệu.
* Tính điểm dựa trên đáp án đúng/sai.
* Lấy và hiển thị tên người dùng khi nhập tên của mình.
* Tạo và chỉnh sửa câu hỏi.
* Hiển thị đáp án đúng sau khi trả lời.
* Lưu kết quả vào MongoDB.
* Hiển thị bảng xếp hạng.

1. Đánh giá sơ bộ

* Hệ thống hoạt động ổn định trên môi trường localhost.
* Người chơi có thể:
* Bắt đầu quiz, trả lời lần lượt các câu hỏi.
* Xem kết quả điểm số sau khi hoàn thành.
* So sánh kết quả với người chơi khác qua bảng xếp hạng.
* Ưu điểm:
  + Hệ thống chạy ổn định trên môi trường thử nghiệm.
  + Các API hoạt động chính xác, dữ liệu lưu trữ đầy đủ.
  + Giao diện đơn giản, dễ sử dụng.
  + Tính năng realtime hoạt động, cho phép nhiều người chơi tham gia cùng lúc.
* Hạn chế:
  + Giao diện chưa thật sự bắt mắt, mới dừng ở mức cơ bản.
  + Chưa tối ưu giao hiện hiển thị trên thiết bị di động.
  + Chưa có tính năng chống gian lận khi làm quiz (ví dụ đổi tab, refresh).
  + Hiệu năng còn hạn chế khi số lượng người chơi tăng cao.

**PHẦN 7: KINH NGHIỆM**

Trong quá trình phát triển website Quiz Game, nhóm đã rút ra nhiều kinh nghiệm quý báu:

1. Quản lý công việc nhóm:

* Việc phân chia công việc theo từng nhánh Git (ví dụ: dev-backend, dev-frontend, dev-backend-quiz) giúp các thành viên làm việc song song và hạn chế xung đột code.
* Thường xuyên kiểm tra pull request, review code trước khi merge để đảm bảo chất lượng.

1. Kỹ năng sử dụng Git & GitHub:

* Làm quen với việc tạo branch, commit message rõ ràng, và xử lý conflict.
* Biết cách tổ chức repository một cách khoa học để dễ bảo trì.

1. Kiến thức kỹ thuật:

* Nắm vững hơn về React trong việc quản lý state, props, và tái sử dụng component.
* Biết cách kết hợp Node.js + Express để xây dựng API đơn giản phục vụ cho hệ thống quiz.
* Làm quen với kết nối cơ sở dữ liệu MongoDB, lưu trữ và truy vấn dữ liệu một cách hiệu quả.

1. Thiết kế giao diện:

* Hiểu rõ hơn về CSS và cách áp dụng vào dự án để tạo giao diện gọn gàng, dễ tùy biến.
* Học được cách tối ưu trải nghiệm người dùng (UX), như thêm tính năng hiển thị tiến độ, thông báo kết quả.

1. Kỹ năng mềm:

* Rèn luyện kỹ năng thuyết trình, báo cáo tiến độ trong nhóm.
* Học cách giải quyết vấn đề khi gặp lỗi: tra cứu tài liệu, thử nghiệm nhiều hướng xử lý.
* Từ những kinh nghiệm trên, nhóm tự tin có thể triển khai các dự án web phức tạp hơn trong tương lai.

PHẦN 8: KẾT QUẢ ĐẠT ĐƯỢC/CHƯA ĐẠT ĐƯỢC

1. Kết quả đạt được

* **Giao diện hoàn thiện cơ bản**: Website đã xây dựng được giao diện chính theo yêu cầu, bao gồm trang hiển thị câu hỏi, lựa chọn đáp án và hiển thị kết quả ngay sau khi làm bài.
* **Hoạt động ổn định**: Hệ thống hoạt động ổn định trong môi trường phát triển, các chức năng chính như chọn đáp án, chấm điểm, và hiển thị kết quả được thực thi đúng.
* **Trải nghiệm người dùng trực quan**: Người dùng dễ dàng thao tác nhờ thiết kế giao diện gọn gàng, có hình ảnh minh họa trong câu hỏi, và bố cục hợp lý.
* **Ứng dụng kiến thức React**: Nhóm đã áp dụng thành công các kiến thức về React (quản lý state, props, component, sự kiện) để xây dựng hệ thống.
* **Khả năng mở rộng**: Cấu trúc mã nguồn được thiết kế theo component, thuận tiện để mở rộng thêm chức năng mới trong tương lai (ví dụ: nhiều dạng câu hỏi, hệ thống tính thời gian, bảng xếp hạng).

1. **Kết quả chưa đạt được**

* **Thiếu kiểm thử trên nhiều môi trường:** Hệ thống mới được chạy thử ở môi trường phát triển (localhost), chưa kiểm thử đầy đủ trên nhiều trình duyệt và thiết bị khác nhau.
* **Chưa tối ưu giao diện**: Dù đã có giao diện cơ bản nhưng vẫn cần cải thiện về mặt responsive (chưa tối ưu cho điện thoại), màu sắc và hiệu ứng còn đơn giản, một vài chỗ như khu vực chơi quiz mẫu vẫn còn chưa cải thiện màu sắc.
* **Chưa triển khai thực tế**: Website chưa được đưa lên hosting hoặc deploy trên nền tảng online (VD: Vercel, Netlify) để người dùng trải nghiệm trực tiếp.

PHẦN 9: HƯỚNG PHÁT TRIỂN TRONG TƯƠNG LAI

Mặc dù hệ thống Quiz Game đã đạt được những mục tiêu cơ bản, nhóm vẫn nhận thấy còn nhiều cơ hội để mở rộng và nâng cấp nhằm đáp ứng tốt hơn nhu cầu của người dùng. Một số định hướng phát triển trong thời gian tới bao gồm:

1. **Bổ sung ngân hàng câu hỏi đa dạng và phong phú hơn**

* Tích hợp hệ thống quản trị (Admin Panel) cho phép giảng viên hoặc người quản trị dễ dàng thêm, chỉnh sửa, và phân loại câu hỏi theo từng chủ đề, cấp độ.
* Mở rộng sang nhiều lĩnh vực khác ngoài toán học, lý thuyết lập trình, và kiến thức phổ thông, từ đó giúp Quiz Game có thể phục vụ nhiều đối tượng khác nhau.

1. **Cải thiện trải nghiệm người dùng (UX/UI)**

* Thiết kế giao diện trực quan, sinh động hơn với hiệu ứng hình ảnh, âm thanh để tăng tính hấp dẫn.
* Phát triển chế độ chơi theo nhóm (multiplayer) để người chơi có thể thi đấu trực tiếp với bạn bè.
* Tích hợp xếp hạng (leaderboard) để tạo sự cạnh tranh lành mạnh.

1. **Ứng dụng công nghệ trí tuệ nhân tạo (AI)**

* Tích hợp hệ thống gợi ý câu hỏi dựa trên năng lực của người học (adaptive learning).
* Đưa ra phân tích chi tiết sau mỗi bài kiểm tra, giúp người chơi nhận biết điểm mạnh – điểm yếu để cải thiện.

1. **Phát triển đa nền tảng**

* Tối ưu hóa Quiz Game để có thể chạy mượt mà trên thiết bị di động (smartphone, tablet).
* Xây dựng ứng dụng di động riêng (Android/iOS) để mở rộng phạm vi tiếp cận.

1. **Kết nối cộng đồng học tập**

* Cho phép người dùng tự tạo bộ câu hỏi riêng và chia sẻ với cộng đồng.
* Tích hợp chức năng bình luận, trao đổi và thảo luận để tăng tính tương tác.

1. **Bảo mật và hiệu năng**

* Cải thiện tốc độ tải trang, tối ưu truy vấn cơ sở dữ liệu khi số lượng câu hỏi và người chơi tăng lên.
* Tích hợp các biện pháp bảo mật, ngăn chặn gian lận trong khi thi.
* Như vậy, với các hướng phát triển trên, Quiz Game sẽ không chỉ dừng lại ở một công cụ kiểm tra kiến thức đơn giản, mà còn trở thành một nền tảng học tập – giải trí toàn diện, linh hoạt và bền vững.

PHẦN 10: KẾT LUẬN

Qua quá trình nghiên cứu và thực hành môn Lập trình mạng, nhóm đã có cơ hội tiếp cận và vận dụng các kiến thức quan trọng như mô hình client-server, giao thức TCP/UDP, kỹ thuật lập trình socket, và xử lý bất đồng bộ. Từ đó, nhóm không chỉ củng cố được lý thuyết đã học mà còn rèn luyện kỹ năng triển khai các ứng dụng mạng thực tế như chương trình chat, truyền file và các dịch vụ trực tuyến cơ bản.

Bên cạnh những kết quả đạt được, nhóm cũng nhận thấy còn tồn tại những hạn chế, đặc biệt là về khả năng tối ưu hóa hiệu năng, xử lý đồng thời nhiều kết nối và đảm bảo an toàn bảo mật trong truyền dữ liệu. Những hạn chế này là cơ sở để nhóm tiếp tục cải thiện trong các nghiên cứu và dự án sau.

Tổng thể, môn học Lập trình mạng đã giúp nhóm hình thành tư duy hệ thống, hiểu rõ vai trò quan trọng của lập trình mạng trong các ứng dụng hiện đại, đồng thời tạo nền tảng vững chắc để phát triển và mở rộng kiến thức trong tương lai.

**TÀI LIỆU THAM KHẢO**

Slide giảng dạy môn Lập trình mạng – Trường Đại học Giao Thông Vận Tải TPHCM

<https://courses.ut.edu.vn/pluginfile.php/844715/mod_resource/content/3/Chuong_3_Mang_may_tinh.pdf>