KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN KẾT THÚC MÔN
CÔNG NGHỆ PHẦN MỀM
HỌC KỲ 2, NĂM HỌC: 2023 – 2024

XÂY DỰNG HỆ THỐNG QUẢN LÝ CÂU LẠC BỘ HÀNH TRÌNH SINH VIÊN

Giáo viên hướng dẫn:
TS. Nguyễn Bảo Ân

Sinh viên thực hiện:
Họ tên: Nguyễn Đại Hoàng Phúc
MSSV: 110121087
Họ tên: Phạm Thúy Hằng
MSSV: 110121182
Họ tên: Huỳnh Nguyễn Bích Trâm
MSSV: 110121118

Trà Vinh, ngày 12 tháng 6 năm 2024

NHẬN XÉT CỦA GIẢNG VIÊN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Giảng viên

MỤC LỤC

CHƯƠNG 1: GIỚI THIỆU

1.1 Giới thiệu về Câu Lạc Bộ Hành Trình Sinh Viên:

CLB Hành trình sinh viên là nơi tạo ra những hành trình đầy ý nghĩa và trải nghiệm đáng nhớ cho sinh viên Đại học Trà Vinh. Với sứ mệnh xây dựng cộng đồng sinh viên tích cực và sáng tạo, câu lạc bộ sẽ mang lại những hoạt động hấp dẫn và ý nghĩa cho tất cả các thành viên.

Câu lạc bộ thường xuyên tổ chức các chương trình thực tế như tham quan, tình nguyện và hội thảo, giúp sinh viên mở rộng hiểu biết, kết nối và thể hiện tinh thần trách nhiệm xã hội.

Đồng thời, câu lạc bộ cũng không quên phát triển kỹ năng mềm và chuyên môn thông qua các khóa học và buổi tập huấn chuyên sâu.

Câu Lạc Bộ Hành Trình Sinh Viên lập ra nhằm mục đích:

- Tạo điều kiện cho sinh viên giao tiếp, ứng xử, vui chơi giải trí lành mạnh, bày tỏ quan điểm, tâm tư nguyện vọng, đồng thời hỗ trợ giải quyết các vấn đề khó khăn, vướng mắc trong học tập, công tác và trong cuộc sống.

- Thông qua các hoạt động để tuyên truyền, nâng cao nhận thức về bản lĩnh chính trị, đạo đức, lối sống, giáo dục lý tưởng cách mạng, truyền thống của dân tộc cho sinh viên.

1.2 Đặc tả ứng dụng:

Câu Lạc Bộ Hành Trình Sinh Viên cần xây dựng một hệ thống quản lý các hoạt động để tối ưu hóa công tác quản lý.

Hệ thống sẽ lưu trữ thông tin chi tiết về sinh viên, bao gồm mã sinh viên, họ tên, ngày sinh, giới tính, số điện thoại, email và hình ảnh. Mỗi sinh viên sẽ thuộc một lớp học cụ thể và có một chức vụ trong câu lạc bộ.

Thông tin về lớp học sẽ được quản lý với mã lớp, tên lớp và mã khoa, trong khi thông tin về khoa sẽ bao gồm mã khoa và tên khoa.

Hệ thống cũng sẽ quản lý tài khoản cho sinh viên, bao gồm mã tài khoản, tên đăng nhập, mật khẩu và quyền hạn.

Chức vụ của từng thành viên trong câu lạc bộ sẽ được ghi nhận thông qua mã chức vụ và tên chức vụ. Mối quan hệ giữa sinh viên và chức vụ sẽ được quản lý trong bảng chức vụ sinh viên, lưu trữ mã chức vụ, mã sinh viên và nhiệm kỳ của chức vụ đó.

Thông tin về các hoạt động của câu lạc bộ sẽ được lưu trữ chi tiết, bao gồm mã hoạt động, tên hoạt động, mô tả, thời gian, địa điểm tổ chức, học kỳ, năm học và trạng thái hoạt động. Sinh viên sẽ đăng ký tham gia các hoạt động này, và quá trình đăng ký sẽ được lưu lại với mã sinh viên, mã hoạt động, ngày đăng ký, trạng thái đăng ký và trạng thái tham gia. Chỉ những sinh viên đã đăng ký tham gia mới được điểm danh có tham gia hoạt động.

Cuối cùng, hệ thống sẽ lưu trữ thông tin về tin tức để thông báo các sự kiện và tin tức liên quan đến câu lạc bộ. Thông tin này bao gồm mã tin tức, mã sinh viên đăng tin, tiêu đề, nội dung, ngày đăng và hình ảnh tin tức. Điều này giúp câu lạc bộ duy trì kênh thông tin hiệu quả và đồng bộ, đảm bảo mọi thông tin đều được cập nhật kịp thời đến tất cả các thành viên.

1.2.1 Đối tượng người dùng:

- Ban Chủ Nhiệm Câu Lạc Bộ (Admin)
- Sinh viên là thành viên Câu Lạc Bộ (Users)

1.2.2 Yêu Cầu Chức Năng:

- Quản Lý Sinh Viên
    - Thêm/Chỉnh sửa/Xóa Sinh Viên: Quản lý thông tin cá nhân của sinh viên.
    - Phân Lớp Sinh Viên: Gán sinh viên vào các lớp học cụ thể.
    - Quản Lý Chức Vụ: Quản lý chức vụ và nhiệm kỳ của sinh viên trong câu lạc bộ.
- Quản Lý Lớp Học và Khoa
    - Thêm/Chỉnh sửa/Xóa Lớp Học: Quản lý thông tin lớp học với các trường
    - Thêm/Chỉnh sửa/Xóa Khoa: Quản lý thông tin về khoa với các trường

- Quản Lý Tài Khoản Sinh Viên: Thêm/Chỉnh sửa/Xóa Tài Khoản: Quản lý tài khoản sinh viên với các trường
- Quản Lý Chức Vụ
  - Thêm/Chỉnh sửa/Xóa Chức Vụ: Quản lý các chức vụ trong câu lạc bộ
  - Gán Chức Vụ: Liên kết sinh viên với chức vụ cụ thể và nhiệm kỳ của chức vụ đó.
- Quản Lý Hoạt Động
  - Thêm/Chỉnh sửa/Xóa Hoạt Động: Quản lý thông tin các hoạt động câu lạc bộ
  - Đăng Ký Tham Gia: Cho phép sinh viên đăng ký tham gia hoạt động.
  - Quản Lý Đăng Ký: Lưu trữ thông tin về đăng ký tham gia
- Quản Lý Tin Tức: Thêm/Chỉnh sửa/Xóa Tin Tức: Quản lý tin tức và thông báo liên quan đến câu lạc bộ
- Tìm Kiếm và Lọc Thông Tin
  - Tìm Kiếm Sinh Viên: Tìm kiếm thông tin sinh viên theo mã sinh viên, tên, lớp học, khoa.
  - Tìm Kiếm Hoạt Động: Tìm kiếm các hoạt động theo tên, thời gian, địa điểm, trạng thái.
  - Lọc Tin Tức: Lọc tin tức theo tiêu đề, ngày đăng, sinh viên đăng tin.
- Quản Lý Quyền Truy Cập
  - Phân Quyền Người Dùng: Quản lý quyền truy cập cho từng loại người dùng (Admin, Thành Viên).
  - Đăng Nhập/Đăng Xuất: Hỗ trợ chức năng đăng nhập/đăng xuất cho người dùng.
- Báo Cáo và Thống Kê:
  - Thống Kê Sinh Viên: Số lượng sinh viên theo lớp, khoa.
  - Thống Kê Hoạt Động: Số lượng hoạt động, số lượng sinh viên tham gia.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 Lý thuyết về Agile và SCRUM

2.1.1 Giới thiệu về Agile

Agile (viết tắt của Agile Software Development) là một phương pháp phát triển phần mềm linh hoạt, được ứng dụng trong quy trình phát triển phần mềm với mục tiêu là đưa sản phẩm đến tay người dùng càng nhanh càng tốt.

Agile nhấn mạnh sự tương tác giữa các thành viên trong nhóm, sự cộng tác chặt chẽ với khách hàng, và việc phát hành phần mềm nhanh chóng với các phiên bản cải tiến liên tục. Triết lý của Agile được thể hiện rõ trong Tuyên ngôn Agile (Agile Manifesto), được công bố vào năm 2001, với bốn giá trị cốt lõi:

- Con người và sự tương tác hơn là quy trình và công cụ.

  Phần mềm chạy tốt hơn là tài liệu đầy đủ.

  Cộng tác với khách hàng hơn là thương thảo hợp đồng.

  Phản hồi với thay đổi hơn là tuân theo kế hoạch.

2.1.2 Nguồn gốc của Alige

Xuất phát từ The Manifesto for Agile Software Development được 17 tác giả đưa ra vào năm 2001. Từ "Agile" được các tác giả chọn vì nó thể hiện được khả năng thích ứng và phản ứng với sự thay đổi, điều mà họ thấy cực kỳ quan trọng với những định hướng của họ.

2.1.3 Ưu, nhược điểm của Agile

Ưu điểm:

- Việc triển khai phần mềm nhanh hơn và do đó giúp tăng sự tin tưởng của khách hàng

- Có thể thích ứng tốt hơn với các yêu cầu thay đổi nhanh chóng và đáp ứng hơn

- Giúp nhận được phản hồi ngay lập tức có thể được sử dụng để cải thiện phần mềm trong những bước tiếp theo

- Con người và sự tương tác được ưu tiên hơn là quy trình và công cụ

- Liên tục chú ý đến sự xuất sắc về kỹ thuật và thiết kế tốt

Nhược điểm

- Trong các dự án phần mềm lớn, rất khó để đánh giá nỗ lực cần thiết ở giai đoạn đầu của vòng đời phát triển phần mềm

- Phát triển Agile phụ thuộc rất nhiều vào đầu vào của khách hàng. Nếu khách hàng cảm thấy mơ hồ trong tầm nhìn của mình về kết quả cuối cùng, thì rất có thể dự án sẽ đi chệch hướng

- Chỉ có những lập trình viên cấp cao mới có khả năng đưa ra quyết định cần thiết trong quá trình phát triển. Do đó, đây là một tình huống khó khăn đối với các lập trình viên mới để thích nghi với môi trường.

2.1.4 Giới thiệu về Scrum

Scrum là một mô hình phát triển phần mềm linh hoạt (Agile) vận hành dựa trên cơ chế lặp và tăng trưởng. Scrum được tạp lập nhằm mục đích hỗ trợ việc phát triển, cung cấp và cải tiến các sản phẩm phức tạp.

Scrum là một framework (khung tổ chức công việc) được áp dụng với các dự án phát triển phần mềm. Với mục đích là tạo điều kiện cho sự hợp tác hiệu quả giữa các thành viên trong nhóm khi phát triển các dự án phức tạp.

2.1.5 Quy trình Scrum:

1. Lập kế hoạch Sprint (Sprint Planning):

- Xác định mục tiêu Sprint (Sprint Goal).

- Lựa chọn các mục từ Product Backlog vào Sprint Backlog.

- Đội phát triển phân chia các mục vào các nhiệm vụ cụ thể và ước lượng công việc.

2. Thực hiện Sprint:

- Đội phát triển làm việc trên các nhiệm vụ trong Sprint Backlog.

- Tổ chức các cuộc họp Daily Scrum để đảm bảo tiến độ và giải quyết các vấn đề nảy sinh.

3. Đánh giá Sprint (Sprint Review):

- Trình bày và kiểm tra công việc đã hoàn thành.

- Nhận phản hồi từ các bên liên quan để cải thiện sản phẩm.

4. Hồi tưởng Sprint (Sprint Retrospective):

- Thảo luận về những gì đã diễn ra tốt và những gì cần cải thiện.

- Đề xuất các hành động cụ thể để cải thiện trong Sprint tiếp theo.

2.1.6 Giá trị cốt lõi của mô hình Scrum

Tính minh bạch (Transparency)

Mô hình Scrum được triển khai thành công khi thông tin quy trình phải công khai minh bạch. Nó bao gồm tầm nhìn sản phẩm, nhu cầu người dùng, tiến độ công việc,… Để tăng sức mạnh cho dự án thì mọi người cần hiểu rõ vai trò của mình trong nhóm và phối hợp với nhau để hoàn thành mục tiêu đề ra. Các công cụ và cuộc họp trong Scrum luôn đảm bảo thông tin cho các bên được minh bạch.

Thanh tra (Inspection)

Để đảm bảo chất lượng của sản phẩm đồng thời tránh được sự chênh lệch quá lớn giữ sản phẩm được sản xuất thực tế và sản phẩm mong muốn. Chúng ta cần kiểm tra liên tục các hoạt động trong Scrum để phát hiện vấn đề cùng giải pháp để thông tin hữu ích đến với các bên tham gia quá trình phát triển. Sự thanh tra nên diễn ra ở một thời điểm nhất định và thực hiện ở cá nhân đủ năng lực. Tại những điểm quan trọng ở công việc thì việc kiểm tra giúp thúc đẩy sự phát triển liên tục ở mô hình Scrum.

Sự thích nghi (Adaptation)

Thông qua các thông tin minh bạch hóa từ quá trình tham gia và làm việc. Scrum có thể phản hồi lại các thay đổi một cách tích cực để đem lại sự thành công cho sản phẩm được tạo ra. Hai giá trị ở trên gồm minh bạch và thanh tra đều hướng tới sự thích nghi nhanh chóng và hiệu quả.

2.1.7 Các vai trò trong Scrum: (Scrum Team)

Khác với các mô hình phát triển phần mềm khác thì ở Scrum Team sẽ có 3 vai trò gồm 3 thành tố sau:

Product Owner: nhằm đảm bảo việc quản lý những cồng việc tồn đọng của việ phát triển phần mềm. Thành phần này phải liên tục cập nhật thông tin cho các thành viên trong team để họ hiểu về các yêu cầu và tính năng cần có của sản phẩm khi họ không trực tiếp phát triển tính năng đó.

Development Team: Các lập trình viên sẽ tham dự vào việc phát triển từng tính năng cụ thể. Mỗi lập trình viên sẽ có thể có nhiều kỹ năng và giỏi về các tính năng nhất định. Tuy vậy khi dùng Scrum thì các thành viên trong Development Team cần có khả năng làm việc ở các vị trí nhất định. Không ai có thể chịu trách nhiệm hoàn toàn về việc xây dựng tính năng cụ thể của sản phẩm.

Scrum Master: Chịu trách nhiệm cho việc lên kế hoạch để phân công công việc. Sắp xếp thứ tự ưu tiên giải quyết những vấn đề tồn đọng có trong Backlog trước. Đồng thời tổ chức các buổi họp với với Product Owner để theo dõi tình hình và cập nhật thông tin cần thiết liên tục.

2.1.8 Lợi ích của Scrum:

- Tăng cường sự minh bạch và giao tiếp: Các cuộc họp thường xuyên và các tạo tác rõ ràng giúp đội ngũ và các bên liên quan hiểu rõ hơn về tiến trình và tình trạng công việc.
- Tăng tính linh hoạt: Khả năng điều chỉnh kế hoạch theo phản hồi từ các Sprint trước giúp sản phẩm phát triển theo hướng đúng đắn.
- Phát hành nhanh và liên tục: Mỗi Sprint kết thúc với một sản phẩm hoàn chỉnh và có thể sử dụng hoặc phát hành, giúp tăng giá trị cho khách hàng.
- Tăng cường tinh thần làm việc nhóm: Đội ngũ tự tổ chức và tự quản lý, thúc đẩy sự hợp tác và sáng tạo trong nhóm.

2.1.9 Những thách thức khi áp dụng Scrum:

- Khả năng thích ứng: Đòi hỏi sự thay đổi trong văn hóa và cấu trúc tổ chức, điều này có thể gặp khó khăn đối với một số doanh nghiệp.

- Yêu cầu sự cam kết: Các thành viên trong nhóm và các bên liên quan cần phải cam kết tham gia đầy đủ vào các quy trình Scrum.
- Khả năng điều phối và quản lý: Scrum đòi hỏi sự phối hợp chặt chẽ và liên tục, điều này có thể là thách thức đối với các đội ngũ lớn hoặc phân tán.

2.2 Lý thuyết về công nghệ, kiến trúc được sử dụng trong ứng dụng:

Trong quá trình phát triển ứng dụng quản lý câu lạc bộ "Hành trình sinh viên", việc lựa chọn các công nghệ và kiến trúc phù hợp là cực kỳ quan trọng. Điều này không chỉ giúp đảm bảo hiệu suất và khả năng mở rộng của ứng dụng mà còn ảnh hưởng đến sự dễ dàng trong việc bảo trì và phát triển sau này. Dưới đây, chúng tôi sẽ trình bày về các công nghệ và kiến trúc được sử dụng trong ứng dụng này.

2.2.1 Asp.Net MVC:

Asp.Net MVC (Model-View-Controller) là một framework phổ biến của Microsoft được sử dụng để phát triển các ứng dụng web. Đây là một phần của .NET framework và cung cấp một cách tiếp cận hiện đại để xây dựng các ứng dụng web với sự tách biệt rõ ràng giữa các phần của ứng dụng.

MVC là một mô hình kiến trúc phân tách ứng dụng thành ba phần chính:

- Model: Đại diện cho dữ liệu và logic nghiệp vụ của ứng dụng. Trong ứng dụng "Hành trình sinh viên", Model bao gồm các đối tượng như Sinh viên, Sự kiện, và Thông tin câu lạc bộ. Entity Framework được sử dụng để quản lý việc tương tác với cơ sở dữ liệu cho các model này.
- View: Là phần giao diện người dùng của ứng dụng. Views chịu trách nhiệm hiển thị dữ liệu từ Model và cung cấp giao diện tương tác với người dùng. Trong ứng dụng của chúng tôi, các Views sẽ hiển thị thông tin như danh sách sự kiện, chi tiết thành viên, và các bảng điều khiển quản trị.
- Controller: Đóng vai trò trung gian giữa Model và View. Controllers xử lý các yêu cầu từ người dùng, tương tác với Model để lấy dữ liệu và quyết định View nào sẽ hiển thị dữ liệu đó. Trong ứng dụng, các Controllers sẽ quản lý các hành động như đăng nhập, đăng ký sự kiện, và quản lý thông tin thành viên.

Lợi ích của Kiến trúc MVC:

- Tách biệt rõ ràng các mối quan tâm: Giúp dễ dàng duy trì và mở rộng ứng dụng vì mỗi thành phần có một nhiệm vụ rõ ràng và độc lập.
- Tái sử dụng mã nguồn: Cho phép tái sử dụng các thành phần Model và Controller với các View khác nhau hoặc trong các ngữ cảnh khác nhau.
- Tăng tính linh hoạt: Dễ dàng thay đổi giao diện người dùng (View) mà không ảnh hưởng đến logic nghiệp vụ (Model).
- Hỗ trợ làm việc nhóm: Cho phép các nhà phát triển làm việc song song trên các thành phần khác nhau mà không gây xung đột.

2.2.2 Entity Framework:

- Entity Framework (EF) là một ORM (Object-Relational Mapping) framework của Microsoft, giúp đơn giản hóa việc làm việc với cơ sở dữ liệu trong các ứng dụng .NET. EF cho phép các nhà phát triển làm việc với dữ liệu dưới dạng các đối tượng .NET mà không cần phải viết nhiều mã SQL thủ công.

- Lợi ịch của Entity Framework:

Trừu tượng hóa cơ sở dữ liệu: EF cung cấp một lớp trừu tượng hóa, cho phép làm việc với dữ liệu dưới dạng các đối tượng mà không cần quan tâm đến các chi tiết cụ thể của cơ sở dữ liệu.

Hỗ trợ LINQ: LINQ (Language Integrated Query) là một ngôn ngữ truy vấn mạnh mẽ tích hợp trong C#. EF hỗ trợ LINQ, giúp viết các truy vấn dữ liệu một cách dễ dàng và trực quan.

Tự động hóa việc tạo và quản lý cơ sở dữ liệu: Với EF, bạn có thể tạo cơ sở dữ liệu từ các model dữ liệu hoặc ngược lại, tạo các model từ một cơ sở dữ liệu hiện có. Điều này giúp tiết kiệm thời gian và giảm thiểu các lỗi phát sinh khi làm việc với cơ sở dữ liệu.

2.2.3 Docker và CI/CD:

- **Docker:**

+ Docker là một nền tảng phần mềm cho phép bạn đóng gói ứng dụng cùng với các phụ thuộc của nó thành các container. Container cung cấp một môi trường độc lập, đảm bảo rằng ứng dụng chạy nhất quán trên các môi trường khác nhau.

+ Các thành phần của Docker:

Docker Engine: Là nền tảng chính của Docker, bao gồm một daemon (quản lý Docker containers), một REST API (giao diện lập trình ứng dụng) để tương tác với daemon, và một giao diện dòng lệnh (CLI) để thực hiện các lệnh Docker.

Docker Image: Là các tập tin chỉ đọc chứa mã nguồn, thư viện và các phần phụ thuộc khác của ứng dụng. Các hình ảnh Docker (images) là mẫu để tạo ra các container.

Docker Container: Là các thể hiện (instances) của Docker image. Mỗi container là một môi trường chạy độc lập với phần còn lại của hệ thống.

Docker Registry: Là nơi lưu trữ và phân phối các Docker image. Docker Hub là một Docker Registry công khai phổ biến nhất.

- CI/CD (Continuous Integration/Continuous Deployment):

CI CD là một phương pháp phát triển phần mềm mà các thay đổi mã được kiểm tra và triển khai tự động. Kết hợp với Docker, CI/CD giúp tự động hóa quá trình kiểm tra và triển khai, đảm bảo rằng ứng dụng được phát hành liên tục và nhanh chóng mà không gặp phải sự cố. Cụ thể:

+ Continuous Integration (CI): Là quá trình tự động hóa việc tích hợp mã từ nhiều lập trình viên vào một nhánh chính của kho mã. CI thường đi kèm với việc chạy các bộ kiểm thử tự động để đảm bảo rằng mã mới không gây ra lỗi.

+ Continuous Deployment (CD): Là quá trình tự động hóa việc triển khai phần mềm đến các môi trường sản xuất. Mỗi thay đổi mã đã qua kiểm thử có thể được triển khai ngay lập tức mà không cần can thiệp thủ công.

Quy trình CI/CD:

+ Commit Code: Các lập trình viên đẩy mã nguồn mới hoặc đã chỉnh sửa lên kho mã (repository).

+ Build: Hệ thống CI/CD tự động lấy mã từ kho, biên dịch và xây dựng phần mềm.

+ Test: Hệ thống chạy các bộ kiểm thử tự động để xác minh tính chính xác của mã mới.

+ Deploy: Nếu tất cả kiểm thử đều thành công, mã mới được triển khai đến môi trường staging hoặc production.

+ Monitor: Sau khi triển khai, hệ thống giám sát để đảm bảo rằng phần mềm hoạt động như mong đợi và không có lỗi phát sinh.

2.2.4 Kiến Trúc Web:

Website "CLB Hành trình sinh viên" được xây dựng theo kiến trúc tầng, một trong những kiến trúc phổ biến trong phát triển phần mềm. Kiến trúc này giúp tách biệt các phần khác nhau của ứng dụng thành các tầng riêng biệt, mỗi tầng chịu trách nhiệm cho một chức năng cụ thể.

Tầng Giao diện (Presentation Layer): Đây là tầng hiển thị giao diện người dùng, sử dụng ASP.NET MVC và Bootstrap để tạo ra các trang web và giao diện tương tác với người dùng.

Tầng Nghiệp vụ (Business Logic Layer): Tầng này quản lý logic nghiệp vụ của ứng dụng. Nó chịu trách nhiệm xử lý các quy tắc nghiệp vụ và các quy trình, tương tác với Model để lấy hoặc cập nhật dữ liệu.

Tầng Dữ liệu (Data Access Layer): Sử dụng Entity Framework để quản lý tương tác với cơ sở dữ liệu. Tầng này xử lý các yêu cầu truy cập dữ liệu từ tầng nghiệp vụ và gửi dữ liệu tương ứng.

CHƯƠNG 3: XÁC ĐỊNH NHU CẦU

3.1 Đặc tả mục tiêu dự án:

Mục tiêu của dự án là phát triển một ứng dụng web ASP.NET MVC để quản lý câu lạc bộ "Hành trình sinh viên", bao gồm các chức năng quản lý thông tin cá nhân, các hoạt động của câu lạc bộ, xem tin tức, đăng ký hoạt động, và phân quyền cho người dùng,...

3.2 Xác định personas:

Personas là các mô hình đại diện cho các nhóm người dùng khác nhau mà trang web sẽ phục vụ.

Sinh viên: người sử dụng chính của ứng dụng, họ có nhu cầu quản lý thông tin cá nhân, tham gia các hoạt động của câu lạc bộ và đọc tin tức.

Quản trị viên câu lạc bộ: người quản lý hoạt động, sự kiện, tin tức và quản lý thành viên.

3.3 Xác định các user stories:

- Admin:

  Story 1: Là một Admin, tôi muốn thêm mới, chỉnh sửa và xóa thành viên để quản lý danh sách thành viên câu lạc bộ.

  Story 2: Là một Admin, tôi muốn tạo và quản lý các sự kiện để tổ chức các hoạt động câu lạc bộ.

  Story 3: Là một Admin, tôi muốn xem báo cáo tham gia của các thành viên để đánh giá mức độ hoạt động của câu lạc bộ.

- Thành viên câu lạc bộ:

  Story 1: Là một thành viên, tôi muốn đăng ký và chỉnh sửa hồ sơ cá nhân để thông tin của tôi luôn chính xác.

  Story 2: Là một thành viên, tôi muốn xem lịch trình các sự kiện để lên kế hoạch tham gia.

  Story 3: Là một thành viên, tôi muốn đăng ký tham gia các sự kiện để tham gia các hoạt động của câu lạc bộ

3.4 .Xác định các product backlog:

- Thiết kế cơ sở dữ liệu và cấu hình Entity Framework.
- Thiết kế giao diện người dùng cơ bản.
- Tạo cơ sở dữ liệu sử dụng Entity Framework.
- Xây dựng chức năng Đăng nhập và Đăng ký.
- Quản lý thông tin Thành viên.
- Tạo, cập nhật và xóa Hoạt động.
- Đăng ký và quản lý Tham gia Hoạt động.
- Tạo, cập nhật, xóa chức năng Tin tức.
- Phân quyền và quản lý vai trò.
- Bổ sung một số tính năng.
- Kiểm tra các lỗi có thể phát sinh.
- Cấu hình CI/CD và triển khai với Docker.

3.5 Xác định các nhu cầu phi tính năng:

- Hiệu năng: Trang web phải có khả năng tải nhanh, phản hồi tốt với số lượng lớn người dùng và dữ liệu.
- Tính sẵn sàng: Đảm bảo trang web luôn sẵn sàng sử dụng với thời gian tối thiểu.
- Khả năng mở rộng: Hệ thống phải dễ dàng mở rộng để thêm các tính năng mới hoặc hỗ trợ nhiều người dùng hơn trong tương lai.
- Thân thiện với người dùng: Giao diện trang web cần trực quan, dễ sử dụng và phù hợp với nhiều loại thiết bị.

3.6 Các yêu cầu bảo mật

- Quản lý Quyền Truy cập:

Xác thực người dùng (Authentication): Đảm bảo chỉ những người dùng đã xác minh danh tính mới có thể truy cập vào hệ thống.

Phân quyền người dùng (Authorization): Xác định và kiểm soát quyền truy cập của người dùng đối với các phần khác nhau của hệ thống.

- Mã hóa Dữ liệu:

Mã hóa dữ liệu khi truyền tải (Data-in-Transit Encryption): Sử dụng các giao thức bảo mật như SSL/TLS để mã hóa dữ liệu trong quá trình truyền tải giữa máy khách và máy chủ, giúp bảo vệ dữ liệu khỏi bị đánh cắp trong quá trình giao tiếp trên mạng.

CHƯƠNG 4: LẬP KẾ HOẠCH SCRUM

Trong quá trình phát triển phần mềm, việc lập kế hoạch là một phần quan trọng để đảm bảo rằng dự án được thực hiện một cách hiệu quả và đáp ứng được yêu cầu của khách hàng. Phương pháp Scrum, một trong những phương pháp phổ biến trong quản lý dự án phần mềm, cung cấp một kỳ hạn linh hoạt và phản hồi liên tục từ khách hàng và nhóm phát triển. Trong chương này, chúng tôi sẽ trình bày về quá trình lập kế hoạch Scrum cho dự án quản lý câu lạc bộ "Hành trình sinh viên" như sau:

4.1 Xác định Sprints:

Đầu tiên, nhóm đã xác định ba sprint cho dự án, mỗi sprint với một thời gian nhất định như sau:

- Sprint 1: Từ ngày 29/04 đến ngày 06/05.
- Sprint 2: Từ ngày 06/05 đến ngày 20/05.
- Sprint 3: Từ ngày 20/05 đến ngày 29/05.

Mỗi sprint được thiết kế để tập trung vào một tập hợp cụ thể các nhiệm vụ và mục tiêu để đảm bảo tiến độ của dự án.

4.1.1 Sprint 1: Thiết kế và Xây dựng nên tảng (1 tuần)

- **Thời gian**: Từ 29/04 đến 06/05 (1 tuần)
- **Mục tiêu**: Thiết kế và xây dựng nền tảng cho ứng dụng.
- **User Stories**:

  Thiết kế cơ sở dữ liệu và cấu hình Entity Framework.

  Thiết kế giao diện người dùng cơ bản.

  Tạo cơ sở dữ liệu sử dụng Entity Framework.

4.1.2 Sprint 2: Phát triển các chức năng (2 tuần)

- **Thời gian**: Từ 06/05 đến 20/05 (2 tuần)
- **Mục tiêu**: Phát triển các chức năng cơ bản của ứng dụng.
- **User Stories**:

  Xây dựng chức năng Đăng nhập và Đăng ký.

Quản lý thông tin Thành viên.

Tạo, cập nhật và xóa Hoạt động.

Đăng ký và quản lý Tham gia Hoạt động.

Tạo, cập nhật, xóa chức năng Tin tức.

Phân quyền và quản lý vai trò.

4.1.3 Sprint 3: Kiểm tra và triển khai (9 ngày)

- **Thời gian**: Từ 20/05 đến 29/05 (1 tuần và 2 ngày)
- **Mục tiêu**: Kiểm tra, bổ sung tính năng và triển khai ứng dụng.
- **User Stories**:

  Bổ sung một số tính năng.

  Kiểm tra các lỗi có thể phát sinh.

  Cấu hình CI/CD và triển khai với Docker.

4.2 Phân công chi tiết:

Sprint	Task	Start	End	Point	Assignee
Thiết kế và Xây dựng nên tảng	Thiết kế cơ sở dữ liệu (DB) và Cấu hình Entity Framework	30/04/2024	01/05/2024	5	Phúc
	Thiết kế giao diện người dùng (UI) cơ bản	01/05/2024	04/05/2024	5	Hằng
	Tạo cơ sở dữ liệu dùng Entity Framework	04/05/2024	05/05/2024	3	Trâm

Sprint	Task	Start	End	Point	Assignee
Phát triển các chức năng	Xây dựng chức năng Đăng nhập và Đăng ký	06/05/2024	09/05/2024	8	Phúc
	Quản lý thông tin Thành viên	09/05/2024	11/05/2024	5	Hằng
	Tạo, Cập nhật và Xóa Hoạt động	11/05/2024	13/05/2024	3	Trâm
	Đăng ký và Quản lý Tham gia Hoạt động	13/05/2024	16/05/2024	8	Phúc
	Tạo, Cập nhật, xóa Chức năng tin tức	15/05/2024	17/05/2024	3	Hằng
	Phân quyền và Quản lý vai trò	18/05/2024	20/05/2024	3	Trâm
Kiểm tra và triển khai	Bổ sung một số tính năng	21/05/2024	25/05/2024	3	Phúc
	Kiểm tra các lỗi có thể phát sinh	24/05/2024	26/05/2024	3	Hằng
	Cấu hình CI CD và Triển khai với Docker	27/05/2024	29/05/2024	5	Phúc

CHƯƠNG 5: HIỆN THỰC HÓA KẾ HOẠCH

5.1 Thiết kế kiến trúc ứng dụng: (3 tầng)

Tầng Giao diện (View): Đây là tầng hiển thị giao diện người dùng, sử dụng ASP.NET MVC và Bootstrap để tạo ra các trang web và giao diện tương tác với người dùng.

- **Chức năng:**

  + Chịu trách nhiệm về hiển thị dữ liệu cho người dùng và nhận đầu vào từ người dùng.

  + Thể hiện dữ liệu từ Model dưới dạng thân thiện với người dùng, sử dụng các thành phần giao diện như văn bản, bảng, biểu mẫu, v.v.

  + Không chứa logic nghiệp vụ mà chỉ tập trung vào việc trình bày dữ liệu.

Tầng Nghiệp vụ (Controller): Tầng này quản lý logic nghiệp vụ của ứng dụng. Nó chịu trách nhiệm xử lý các quy tắc nghiệp vụ và các quy trình, tương tác với Model để lấy hoặc cập nhật dữ liệu.

- **Chức năng:**

  + Điều khiển luồng dữ liệu giữa Model và View.

  + Xử lý các yêu cầu từ người dùng, thao tác dữ liệu qua Model và trả về View tương ứng.

  + Thường chứa các phương thức hành động (action methods) tương ứng với các yêu cầu HTTP.

Tầng Dữ liệu (Model): Sử dụng Entity Framework để quản lý tương tác với cơ sở dữ liệu. Tầng này xử lý các yêu cầu truy cập dữ liệu từ tầng nghiệp vụ và gửi dữ liệu tương ứng.

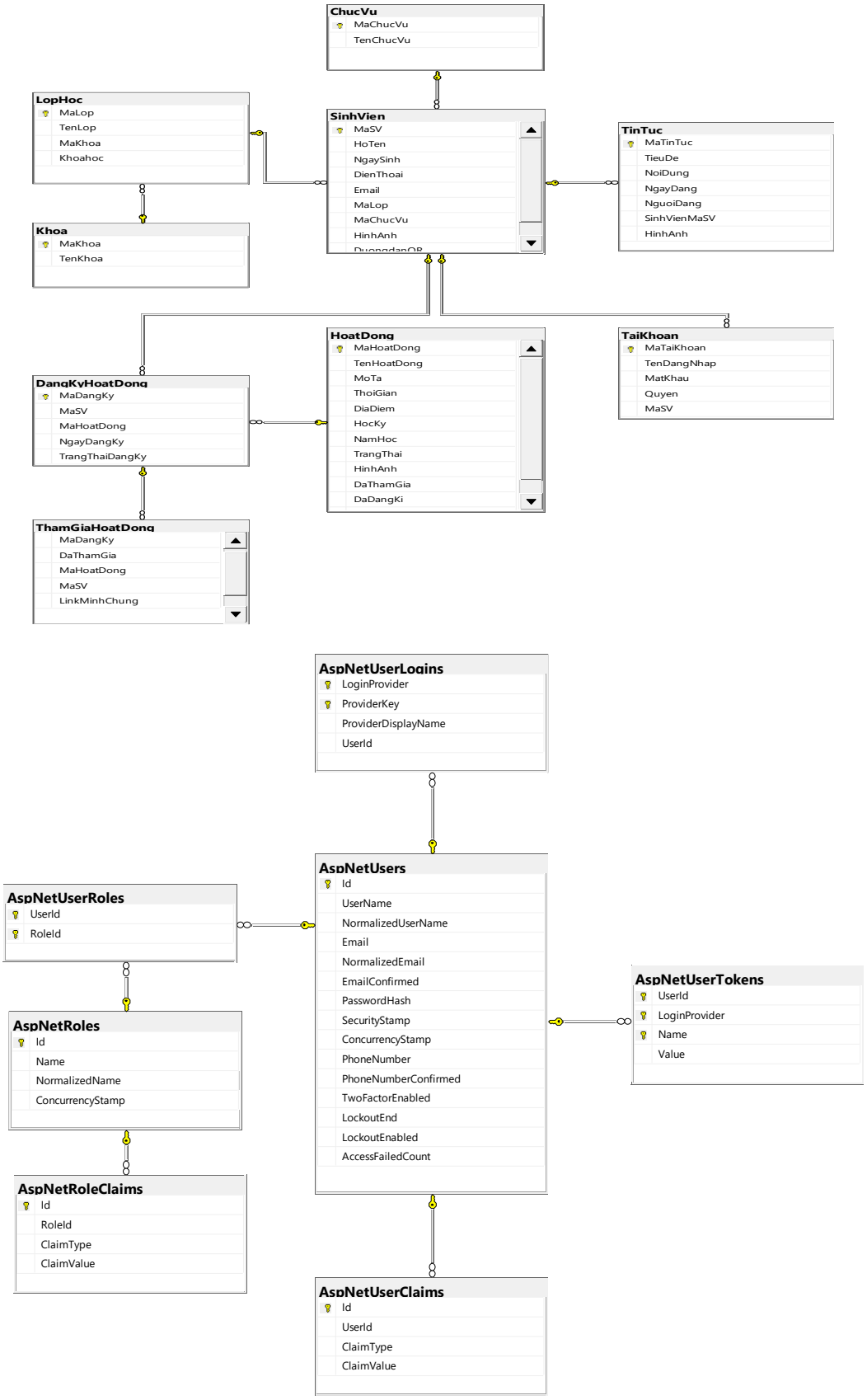
- **Chức năng:**

  + Đại diện cho các dữ liệu và logic nghiệp vụ của ứng dụng.

  + Quản lý trạng thái dữ liệu và tương tác với cơ sở dữ liệu hoặc các dịch vụ khác để xử lý dữ liệu.

  + Chứa các lớp, phương thức để truy xuất, lưu trữ và thao tác dữ liệu.

5.2 Thiết kế database:



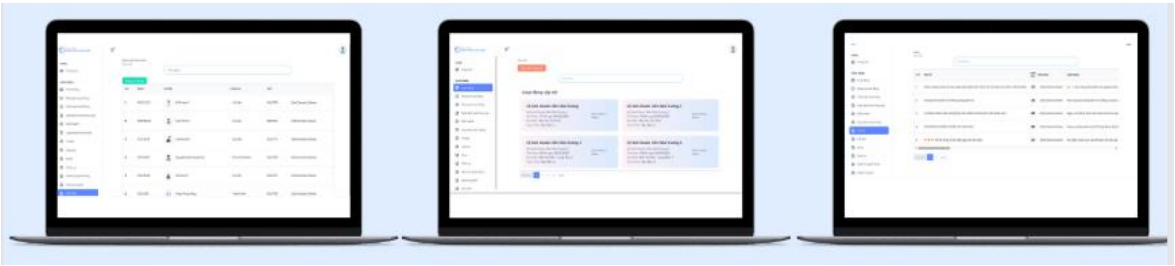
5.3 Kết quả các Sprints:

- Sprint 1: Thiết kế và Xây dựng nên tảng:

+Thiết kế giao diện cơ bản:

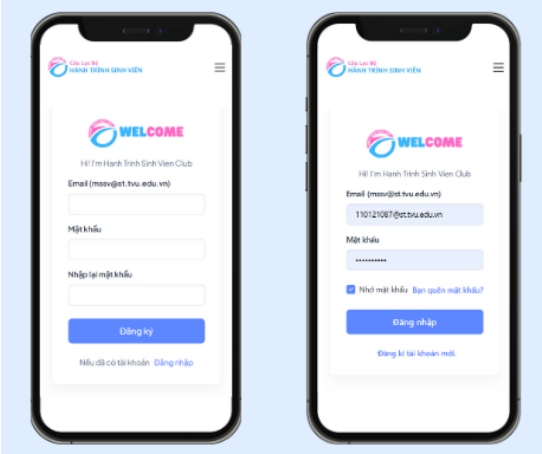
- Sprint 2: Phát triển các chức năng:
- **Một số chức năng của Admin:**

  + Gồm các giao diện, xử lý: thêm sửa xóa sinh viên, hoạt động, tin tức, ...

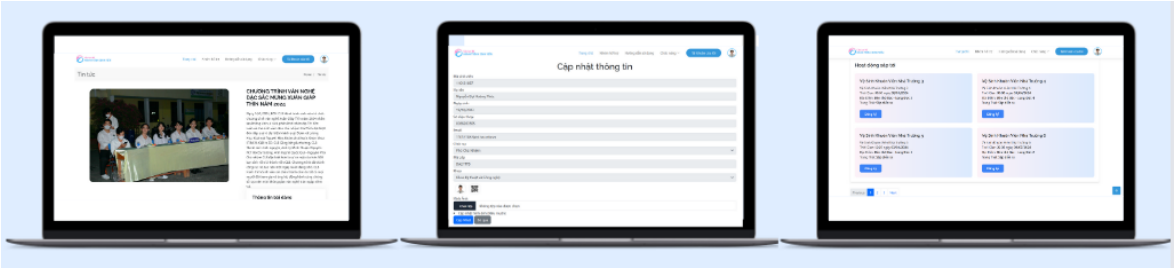


- **Một số chức năng của User:**

  + Giao diện, xử lý đăng nhập, đăng ký:



  + Gồm các giao diện, xử lý:



- Sprint 3: Triển khai với Docker kết hợp CI/CD:

  + Dockerfile:

```
# Bước 1: Sử dụng image aspnet 6.0
FROM mcr.microsoft.com/dotnet/aspnet:6.0 AS base
WORKDIR /app
EXPOSE 80
EXPOSE 443

# Bước 2: Thêm các tệp certificate và private key vào container
COPY Certificates/certificate.crt /app
COPY Certificates/private.key /app
COPY Certificates/your_certificate.pfx /app

# Bước 3: Thiết lập HTTPS cho Kestrel
ENV ASPNETCORE_URLS=http://+:80;https://+:443
RUN sed -i 's/TLSv1.2/TLSv1.0 TLSv1.1 TLSv1.2/g' /etc/ssl/openssl.cnf

# Bước 4: Tạo stage mới để thực thi các lệnh dotnet dev-certs
FROM mcr.microsoft.com/dotnet/sdk:6.0 AS certs
WORKDIR /app

# Khai báo ARG để truyền biến từ build command
ARG PFX_PASSWORD

# Sử dụng biến ARG với lệnh dotnet dev-certs
RUN dotnet dev-certs https -ep /https/aspnetapp.pfx -p $PFX_PASSWORD
RUN openssl pkcs12 -in /https/aspnetapp.pfx -out /https/aspnetapp.pem -nodes -password pass:$PFX_PASSWORD

# Bước 5: Cài đặt ứng dụng
FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build
WORKDIR /src
COPY . .

# Bước 6: Thiết lập biến môi trường trong runtime
ARG DB_PASSWORD
ARG SMTP_PASSWORD
ARG PFX_PASSWORD
ENV DB_PASSWORD=$DB_PASSWORD
ENV SMTP_PASSWORD=$SMTP_PASSWORD
ENV PFX_PASSWORD=$PFX_PASSWORD

# Thay đổi nội dung của tệp appsettings.json
RUN sed -i "s|\${secrets.DB_PASSWORD}|$DB_PASSWORD|g" appsettings.json
RUN sed -i "s|\${secrets.SMTP_PASSWORD}|$SMTP_PASSWORD|g" appsettings.json
RUN sed -i "s|\${secrets.PFX_PASSWORD}|$PFX_PASSWORD|g" appsettings.json

RUN dotnet restore
RUN dotnet build -c Release -o /app/build

# Bước 7: Publish ứng dụng
FROM build AS publish
RUN dotnet publish -c Release -o /app/publish

# Bước 8: Build ứng dụng cuối cùng
FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
COPY --from=certs /https/aspnetapp.pem /https/aspnetapp.pem

ENTRYPOINT ["dotnet", "Manage_CLB_HTSV.dll"]
```

- Giải thích:

- Bước 1: Sử dụng image aspnet 6.0:

  Trong Docker, mỗi image là một bản sao của một hệ điều hành và một ứng dụng. Ở đây, chúng ta đang sử dụng một image có sẵn từ Microsoft chứa ASP.NET Core Runtime phiên bản 6.0.

AS base đặt tên cho stage này, giúp ta xác định các stage sau có thể sử dụng lại được.

- Bước 2: Thêm các tệp certificate và private key vào container:

Các tệp certificate và private key cần thiết để thiết lập HTTPS cho ứng dụng của bạn được sao chép vào thư mục /app trong container.

Điều này là bước chuẩn bị cho việc cấu hình HTTPS sau này.

- Bước 3: Thiết lập HTTPS cho Kestrel:

Kestrel là máy chủ web được tích hợp sẵn trong ASP.NET Core.

ASPNETCORE_URLS được sử dụng để chỉ định các URL mà ứng dụng sẽ lắng nghe, bao gồm cả HTTP (port 80) và HTTPS (port 443).

sed được sử dụng để chỉnh sửa cấu hình SSL, chuyển từ TLS v1.2 sang TLS v1.0, TLS v1.1 và TLS v1.2.

- Bước 4: Tạo stage mới để thực thi các lệnh dotnet dev-certs:

Tạo một stage mới với tên certs để thực hiện việc tạo certificate HTTPS cho ứng dụng.

Sử dụng dotnet dev-certs để tạo ra certificate, sử dụng password được truyền từ biến môi trường.

Sử dụng openssl để chuyển đổi định dạng từ .pfx sang .pem.

- Bước 5: Cài đặt ứng dụng:

Định nghĩa stage build để cài đặt và cấu hình ứng dụng.

Sao chép toàn bộ nội dung của thư mục project vào thư mục làm việc trong container.

Các biến ARG được sử dụng để truyền giá trị từ secrets của GitHub vào quá trình xây dựng Docker images.

- Bước 6: Thiết lập biến môi trường trong runtime:

Thiết lập các biến môi trường để sử dụng trong quá trình chạy của container, bao gồm các thông tin về cơ sở dữ liệu và SMTP.

Bước 7: Thay thế chuỗi ${secrets} trong tệp appsettings.json:

Sử dụng sed để thay thế các giá trị mật khẩu trong tệp appsettings.json bằng các giá trị của các biến môi trường đã được thiết lập.

- Bước 8: Publish ứng dụng:

Định nghĩa stage publish để build ứng dụng và publish vào thư mục /app/publish.

- Bước 9: Build ứng dụng cuối cùng:

Các ứng dụng đã được build và publish sẽ được sử dụng trong Docker container cuối cùng để triển khai.

+ Cấu hình CI/CD:

```yaml
name: CI/CD

on:
  push:
    branches:
      - main
jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout repository
        uses: actions/checkout@v2

      - name: Set up .NET
        uses: actions/setup-dotnet@v1
        with:
          dotnet-version: '6.0.x'

      - name: Restore dependencies
        run: dotnet restore

      - name: Build project
        run: dotnet build --configuration Release

      - name: Test project
        run: dotnet test

      - name: Build Docker image
        run: |
          docker build --build-arg DB_PASSWORD=${{ secrets.DB_PASSWORD }} \
                       --build-arg SMTP_PASSWORD=${{ secrets.SMTP_PASSWORD }} \
                       --build-arg PFX_PASSWORD=${{ secrets.PFX_PASSWORD }} \
                       -t phuchoang1910/webapp_htsv:latest .

      - name: Login to Docker Hub
        run: |
          docker login -u ${{ secrets.DOCKER_USERNAME }} -p ${{ secrets.DOCKER_PASSWORD }}

      - name: Push Docker image
        run: |
          docker push phuchoang1910/webapp_htsv:latest

      - name: Deploy to Production
        uses: appleboy/ssh-action@master
        with:
          host: ${{ secrets.VPS_HOST }}
          username: ${{ secrets.VPS_USERNAME }}
          key: ${{ secrets.SSH_PRIVATE_KEY }}
          port: ${{ secrets.VPS_SSH_PORT }}
          script: |
            docker pull phuchoang1910/webapp_htsv:latest
            docker stop webapp_htsv || true
            docker rm webapp_htsv || true
            docker run -d --name webapp_htsv -p 80:80 -p 443:443 phuchoang1910/webapp_htsv:latest

```

- Giải thích:

name: CI/CD: Định danh cho workflow, trong trường hợp này là "CI/CD".

on: Xác định các sự kiện sẽ kích hoạt workflow này. Trong trường hợp này, workflow sẽ được kích hoạt khi có sự kiện push (đẩy code lên) vào nhánh main của repository.

jobs: Định nghĩa các công việc cần thực hiện trong workflow.

build: Tên của công việc, bạn có thể đặt tên tùy ý.

runs-on: Xác định hệ điều hành mà công việc này sẽ chạy trên. Trong trường hợp này, công việc sẽ chạy trên Ubuntu.

steps: Danh sách các bước cần thực hiện trong công việc.

Checkout repository: Sử dụng action checkout để checkout mã nguồn từ repository.

Set up .NET: Sử dụng action setup-dotnet để cài đặt phiên bản .NET Core 6.0.

Restore dependencies: Sử dụng lệnh dotnet restore để khôi phục các gói phụ thuộc của dự án.

Build project: Sử dụng lệnh dotnet build để build dự án với cấu hình Release.

Test project: Sử dụng lệnh dotnet test để chạy các bài kiểm tra dự án.

Build Docker image: Sử dụng lệnh docker build để build Docker image của ứng dụng. Các biến secret từ GitHub (DB_PASSWORD, SMTP_PASSWORD, PFX_PASSWORD) được truyền vào trong quá trình build.

Login to Docker Hub: Đăng nhập vào Docker Hub bằng cách sử dụng tên người dùng và mật khẩu được lưu trữ trong secrets của GitHub.

Push Docker image: Đẩy Docker image lên Docker Hub.

Deploy to Production: Sử dụng action ssh-action để triển khai ứng dụng lên môi trường production. Công việc này bao gồm việc pull Docker image từ Docker Hub, dừng và xóa container cũ (nếu có), và sau đó khởi chạy một container mới với Docker image vừa được cập nhật.

CHƯƠNG 6: KẾT LUẬN

6.1 Kết Quả Đạt Được:

Phát triển một hệ thống quản lý hiệu quả: Ứng dụng đã được xây dựng để đáp ứng nhu cầu quản lý của câu lạc bộ, từ việc quản lý thông tin thành viên, tổ chức và quản lý các hoạt động, đến việc cung cấp các chức năng phân quyền và quản lý vai trò.

6.2 Giá Trị Mang Lại:

Nâng cao trải nghiệm người dùng: Với giao diện người dùng trực quan và dễ sử dụng, ứng dụng đã tạo điều kiện thuận lợi cho các thành viên câu lạc bộ trong việc tương tác và tham gia vào các hoạt động của câu lạc bộ.

Tăng cường khả năng mở rộng và bảo trì: Nhờ vào kiến trúc rõ ràng và công nghệ hiện đại, ứng dụng có khả năng mở rộng dễ dàng trong tương lai và dễ dàng bảo trì bởi các nhà phát triển khác.

BẢNG PHÂN CÔNG CÔNG VIỆC BÁO CÁO

STT	Nội dung	Người thực hiện
1	Viết mã nguồn, thuyết trình tìm kiếm nội dung.	Nguyễn Đại Hoàng Phúc
2	Tìm kiếm nội dung, thuyết trình, làm báo cáo.	Huỳnh Nguyễn Bích Trâm
3	Tìm kiếm nội dung, thuyết trình, làm slide trình chiếu	Phạm Thúy Hằng