

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO TIỂU LUẬN MÔN
CHUYÊN ĐỀ CÔNG NGHỆ PHẦN MỀM

XÂY DỰNG WEB CHAT
THEO THỜI GIAN THỰC SỬ DỤNG
SIGNALR KẾT HỢP ASP.NET

Giảng viên hướng dẫn: **ThS. DOÃN XUÂN THANH**

Sinh viên thực hiện: **TÔ VĨNH KHANG - 51800408**

BÙI QUANG KHẢI - 51800785

DU THUẬN LONG - 51800429

Lớp: **18050203**

Khóa: **22**

Nhóm Lớp: **04**

Tổ: **02**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO TIỂU LUẬN MÔN
CHUYÊN ĐỀ CÔNG NGHỆ PHẦN MỀM

XÂY DỰNG WEB CHAT
THEO THỜI GIAN THỰC SỬ DỤNG
SIGNALR KẾT HỢP ASP.NET

Giảng viên hướng dẫn: **ThS. DOÃN XUÂN THANH**

Sinh viên thực hiện: **TÔ VĨNH KHANG - 51800408**

BÙI QUANG KHẢI - 51800785

DU THUẬN LONG - 51800429

Lớp: **18050203**

Khóa: **22**

Nhóm Lớp: **04**

Tổ: **02**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021

LỜI CẢM ƠN

Chuyên Đề Công Nghệ Phần Mềm là một môn học rất hay và bổ ích, chúng em xin chân thành cảm ơn Thầy Doãn Xuân Thanh đã hướng dẫn và tạo dựng những điều kiện tốt nhất để chúng em hoàn thành đồ án cuối kỳ với đề tài “Xây dựng Web chat theo thời gian thực sử dụng SignalR kết hợp ASP.NET”. Giúp chúng em có thêm kiến thức, hiểu biết khái quát hơn về cách sử dụng nền tảng ASP.NET.

Trong quá trình thực hiện, nhóm vẫn khó tránh khỏi những sai sót không mong muốn, kính mong thầy có thể góp ý và giúp đỡ chúng em. Nhóm xin chân thành cảm ơn thầy.

BÀI TIỂU LUẬN NÀY ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Chúng tôi xin cam đoan đây là sản phẩm của riêng chúng tôi và được sự hướng dẫn của thầy Doãn Xuân Thanh. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong bài tập lớn này còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào chúng tôi xin hoàn toàn chịu trách nhiệm về nội dung bài tập lớn của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do chúng tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 29 tháng 11 năm 2021

Tác giả

(ký tên và ghi rõ họ tên)

Tô Vĩnh Khang

(Trưởng nhóm)

Bùi Quang Khải

Du Thuận Long

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

Tp. Hồ Chí Minh, ngày tháng năm
(ký và ghi họ tên)

Phần đánh giá của GV chấm bài

Tp. Hồ Chí Minh, ngày tháng năm
(ký và ghi họ tên)

TÓM TẮT

Chúng em chọn đề tài này là vì muốn có thêm kiến thức về xây dựng website trên nền tảng ASP.NET. Loại hình nhắn tin trực tuyến này đang ngày càng được ưa chuộng. Với đề tài “Xây dựng Web chat theo thời gian thực sử dụng SignalR kết hợp ASP.NET”. Nhóm đã thảo luận, thống nhất ý kiến, phân tích yêu cầu về việc phải triển khai thiết kế xây dựng một website mang tên MiniZalo với phần front-end sử dụng React, Redux, TypeScript, MaterialUI và phần back-end sử dụng thư viện SignalR, nền tảng ASP.NET và lưu trữ trên cơ sở dữ liệu PostgreSQL. Website cho phép người dùng kết bạn với nhau để chat trực tuyến cũng như trao đổi qua lại các tập tin tài liệu, hình ảnh,.. Để giúp cho việc sử dụng được dễ dàng, website minh họa này phải đáp ứng được phần giao diện thân thiện, dễ sử dụng, chức năng chat phải theo thời gian thực (real-time). Thông qua sơ đồ usecase và ERD đã minh họa được các chức năng cần thiết và cốt lõi của website, cách thức hoạt động của từng chức năng và một số ràng buộc quan trọng. Qua đó, làm rõ được cách thức hoạt động của thư viện SignalR trong nền tảng ASP.NET.

Với đề tài hấp dẫn và thực tiễn này, nhóm mong muốn áp dụng những kiến thức đã học trong trường cùng với việc tìm hiểu nghiên cứu, phân tích hệ thống, những kiến thức từ bên ngoài cũng những trải nghiệm của cá nhân để xây dựng ứng dụng hỗ trợ được cho mọi người trong công việc lẫn học tập đạt được hiệu quả cao nhất.

MỤC LỤC

LỜI CẢM ƠN.....	3
PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN.....	5
TÓM TẮT.....	6
MỤC LỤC.....	7
DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT.....	9
DANH MỤC CÁC BẢNG BIỂU VÀ HÌNH.....	10
Danh mục Hình.....	10
Danh mục Bảng.....	10
CHƯƠNG I: CƠ SỞ LÝ THUYẾT.....	11
1.1 ASP.NET.....	11
1.2 SignalR.....	11
1.3 PostgreSQL.....	12
1.4 React.....	12
1.5 Redux.....	13
1.6 TypeScript.....	13
1.7 MaterialUI.....	14
CHƯƠNG II: PHÂN TÍCH YÊU CẦU.....	15
2.1 Yêu cầu chức năng.....	15
2.1.1 Mô tả.....	15
2.1.2 Xác định các tác nhân.....	15
2.1.3 Xác định các use case.....	15
2.1.4 Xác định phiên bản công nghệ sử dụng.....	16
2.2 Yêu cầu phi chức năng.....	17

2.2.1 Yêu cầu về Hiệu suất.....	17
2.2.2 Yêu cầu về Bảo mật.....	17
2.2.3. Thuộc tính chất lượng phần mềm.....	17
CHƯƠNG III: THIẾT KẾ HỆ THỐNG.....	18
3.1 Sơ đồ use case tổng quát.....	18
3.2 Sơ đồ ERD.....	18
CHƯƠNG IV: HIỆN THỰC HỆ THỐNG VÀ KẾT QUẢ.....	19
4.1 Phần code khởi tạo.....	19
4.1.1 DataContext.....	19
4.1.2 Entities.....	19
4.1.3 Repositories.....	21
4.1.4 Hubs.....	26
4.1.5 Helpers.....	26
4.1.6 Startup.....	27
4.2 Phần code xử lý.....	27
4.2.1 AuthController.....	27
4.2.2 FriendController.....	29
4.2.3 InboxController.....	30
4.2.4 MessageController.....	34
4.3 Kết quả đạt được.....	37
CHƯƠNG IV: TỔNG KẾT.....	41
PHÂN CÔNG CÔNG VIỆC VÀ ĐÁNH GIÁ.....	42
Phân công công việc.....	42
Đánh giá thành viên.....	43
TÀI LIỆU THAM KHẢO.....	44

DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT

CSDL	Cơ Sở Dữ Liệu
ASP	Active Server Pages
NET	Network Enabled Technologies
API	Application Programming Interface
SQL	Structured Query Language
ERD	Entity Relationship Diagram
KKL	Khang Khải Long

DANH MỤC CÁC BẢNG BIỂU VÀ HÌNH

Danh mục Hình

Hình 1. Hình ảnh về logo ASP.NET.....	11
Hình 2. Hình ảnh về logo SignalR.....	11
Hình 3. Hình ảnh về logo PostgreSQL.....	12
Hình 4. Hình ảnh về logo React.....	12
Hình 5. Hình ảnh về logo React.....	13
Hình 6. Hình ảnh về logo TypeScript.....	13
Hình 7. Hình ảnh về logo MaterialUI.....	14
Hình 8. Sơ đồ use case Tổng quát.....	18
Hình 9. Sơ đồ ERD.....	18
Hình 10. Hình ảnh về Đăng ký.....	37
Hình 11. Hình ảnh về Đăng nhập.....	37
Hình 12. Hình ảnh về Đăng xuất.....	37
Hình 13. Hình ảnh về Tìm kiếm người dùng để kết bạn.....	38
Hình 14. Hình ảnh về Danh sách sau khi kết bạn.....	38
Hình 15. Hình ảnh về Tạo nhóm chat.....	38
Hình 16. Hình ảnh về Trò chuyện nhóm.....	39
Hình 17. Hình ảnh về Trò chuyện cá nhân.....	39
Hình 18. Hình ảnh về Gửi file hình.....	39
Hình 19. Hình ảnh về Gửi file tài liệu và video.....	40
Hình 20. Hình ảnh về Các emoji.....	40

Danh mục Bảng

Bảng 1. Các use case.....	16
---------------------------	----

CHƯƠNG I: CƠ SỞ LÝ THUYẾT

1.1 ASP.NET

ASP.NET là một nền tảng dành cho phát triển web, cho phép những người lập trình tạo ra những trang web động, những ứng dụng web và những dịch vụ web. Nó được thiết kế để tương thích với giao thức HTTP. Thông thường, ngôn ngữ C# được sử dụng nhiều nhất trong ASP.NET.



Hình 1. Hình ảnh về logo ASP.NET

1.2 SignalR

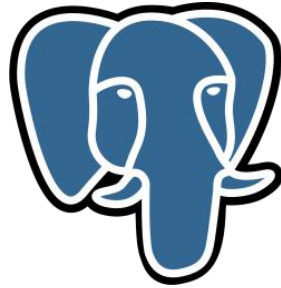
SignalR là thư viện phần mềm mã nguồn mở và miễn phí dành cho ASP.NET, cho phép mã máy chủ gửi thông báo không đồng bộ đến các ứng dụng web phía máy khách. Nó tận dụng WebSocket, một API HTML5 cho phép giao tiếp hai chiều giữa trình duyệt và máy chủ. SignalR sẽ sử dụng WebSockets dưới vỏ bọc khi có sẵn và quay trở lại các kỹ thuật và công nghệ khác khi không có, trong khi mã ứng dụng vẫn giữ nguyên.



Hình 2. Hình ảnh về logo SignalR

1.3 PostgreSQL

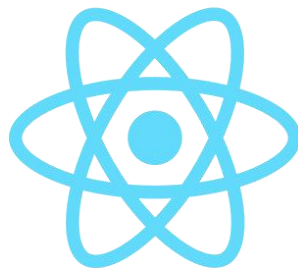
PostgreSQL là một hệ quản trị cơ sở dữ liệu quan hệ, hỗ trợ mạnh trong việc lưu trữ dữ liệu không gian. Nó mở đường cho nhiều khái niệm quan trọng mà các hệ quản trị dữ liệu thương mại rất lâu sau mới có.



Hình 3. Hình ảnh về logo PostgreSQL

1.4 React

React là một thư viện mã nguồn mở và miễn phí để xây dựng giao diện người dùng hoặc các thành phần UI. React chỉ quan tâm đến việc quản lý trạng thái và hiển thị trạng thái đó cho DOM, vì vậy việc tạo các ứng dụng React thường yêu cầu sử dụng các thư viện bổ sung để định tuyến, cũng như một số chức năng phía máy khách nhất định. Điều này cho phép lựa chọn bất kỳ thư viện nào mà nhà phát triển thích để hoàn thành các tác vụ như thực hiện truy cập mạng hoặc lưu trữ dữ liệu cục bộ.



Hình 4. Hình ảnh về logo React

1.5 Redux

Redux là một thư viện nhỏ với một API giới hạn, mã nguồn mở dùng để quản lý và được thiết kế đơn giản để trở thành một vùng chứa có thể đoán trước cho trạng thái ứng dụng. Hiểu đơn giản chính là quản lý các trạng thái chính trong ứng dụng và quản lý các API tương ứng cho từng trạng thái bởi ko phải thông qua từng component hay page, một giá trị trạng thái mà khi muốn lấy chỉ việc gọi ra. Nó được sử dụng phổ biến nhất trong thư viện React để xây dựng giao diện người dùng.



Hình 5. Hình ảnh về logo React

1.6 TypeScript

TypeScript là một ngôn ngữ lập trình, một tập hợp siêu cú pháp nghiêm ngặt của JavaScript và thêm tính năng kiểu tĩnh tùy chọn vào ngôn ngữ. Nó được thiết kế để phát triển các ứng dụng lớn. Vì TypeScript là một siêu tập hợp JavaScript, các chương trình JavaScript hiện có cũng là các chương trình TypeScript hợp lệ. Nó có thể được sử dụng để phát triển các ứng dụng cho cả thực thi phía máy khách và phía máy chủ. Những thách thức đối với việc xử lý mã JavaScript phức tạp đã dẫn đến nhu cầu về công cụ tùy chỉnh để dễ dàng phát triển các thành phần trong ngôn ngữ này.



Hình 6. Hình ảnh về logo TypeScript

1.7 MaterialUI

MaterialUI là một thư viện cho phép nhập và sử dụng các thành phần khác nhau để tạo giao diện người dùng trong các ứng dụng React. Điều này giúp tiết kiệm đáng kể thời gian vì các nhà phát triển không cần phải viết mọi thứ từ đầu.



Hình 7. Hình ảnh về logo MaterialUI

CHƯƠNG II: PHÂN TÍCH YÊU CẦU

2.1 Yêu cầu chức năng

2.1.1 Mô tả

Người dùng truy cập trang website mang tên MiniZalo, đăng ký tài khoản sau đó đăng nhập vào hệ thống. Tìm kiếm tài khoản người khác thông qua thanh tìm kiếm. Sau khi kết bạn, họ có thể chat và gửi file trực tuyến cho nhau. Bên cạnh đó, người dùng có thể tạo nhóm và mỗi nhóm được tạo ra là riêng biệt. Người dùng có thể mời bạn bè vào nhóm. Người dùng khi ở trong nhóm có thể thoát nhóm để tự rời khỏi nhóm hoặc người tạo nhóm (chủ nhóm) có thể xóa nhóm đã tạo. Hệ thống sẽ điều hướng người dùng về trang đăng nhập nếu chưa thực hiện bước đăng nhập.

2.1.2 Xác định các tác nhân

Qua tìm hiểu từ các nguồn, những kiến thức sẵn có cũng như tính toán nhân lực của nhóm, thời gian triển khai dự án. Nhóm chúng em xác định hệ thống này có 2 tác nhân là người dùng và chủ nhóm.

2.1.3 Xác định các use case

Các usecase được xem là cần thiết cho hệ thống đã được nhóm đúc kết như sau:

ID	Use case	Mô tả
SISO	Đăng nhập/Đăng xuất	Người dùng đăng nhập hoặc đăng xuất hệ thống
SU	Đăng ký	Người dùng đăng kí tài khoản lên hệ thống
SE	Tìm kiếm	Người dùng tìm kiếm các người dùng khác
AF	Kết bạn	Người dùng kết bạn với người dùng khác

CT	Chat trực tuyến	Người dùng chat trực tuyến với bạn bè
SF	Gửi file trực tuyến	Người dùng gửi file trực tuyến với bạn bè
GC	Trò chuyện nhóm	Người dùng chat và gửi file trực tuyến với nhóm
CR	Tạo nhóm	Người dùng tạo nhóm
JR	Tham gia nhóm	Người dùng vào nhóm đã tạo
SR	Mời vào nhóm	Người dùng mời bạn bè vào nhóm đã tạo
DR	Xóa nhóm	Người dùng là chủ nhóm có thể xóa nhóm
ER	Thoát nhóm	Người dùng rời khỏi nhóm

Bảng 1. Các use case

2.1.4 Xác định phiên bản công nghệ sử dụng

- ASP.NET (Phiên bản 4.8.0)
- SignalR (Phiên bản 2.4.2)
- PostgreSQL (Phiên bản 9.5.25)
- React (Phiên bản 17.0.0)
- Redux (Phiên bản 7.2.6)
- TypeScript (Phiên bản 4.3.2)
- MaterialUI (Phiên bản 5.0.5)

2.2 Yêu cầu phi chức năng

2.2.1 Yêu cầu về Hiệu suất

Dữ liệu được cập nhật theo thời gian thực, nhất quán toàn vẹn.

Thông tin tiếp nhận được nhanh chóng hiệu quả.

Giao diện thân thiện, dễ sử dụng, tốc độ phản hồi nhanh.

Có khả năng duy trì thời gian truy cập cao và ổn định.

2.2.2 Yêu cầu về Bảo mật

Đảm bảo dữ liệu nhóm nhóm truy cập hiển thị đúng với các tài khoản người dùng khác đã truy cập.

2.2.3. Thuộc tính chất lượng phần mềm

Sử dụng mọi lúc mọi nơi.

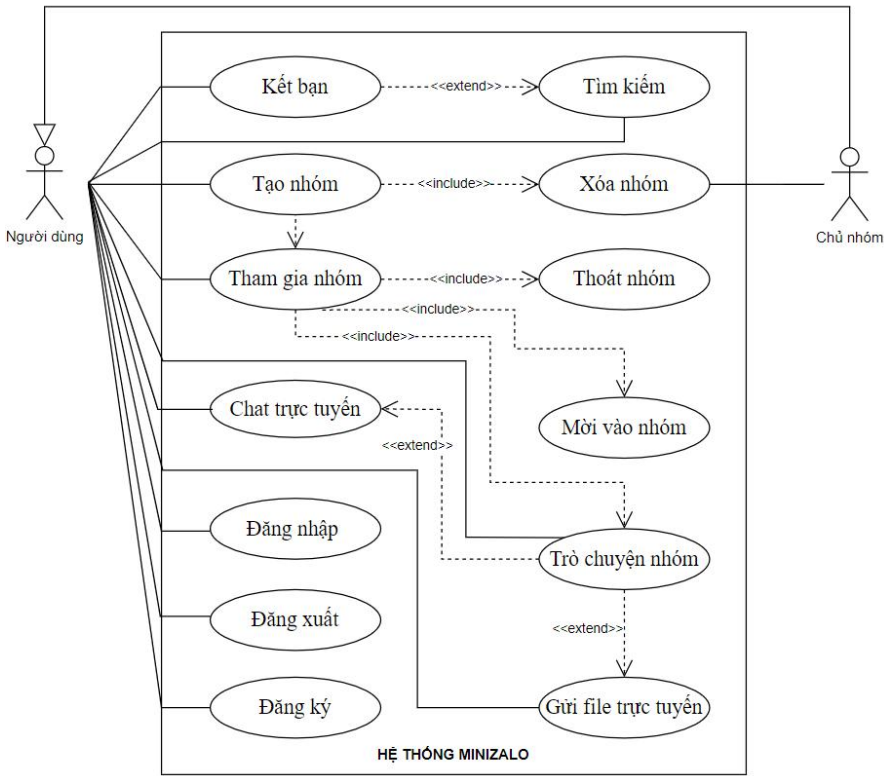
Dễ dàng bảo trì, nâng cấp, phát triển phần mềm sau khi đã triển khai.

Khả năng tương tác, khả năng bảo trì, tính di động, độ tin cậy, tính mạnh mẽ

Khả năng kiểm soát dữ liệu tuyệt đối, an toàn và bảo mật.

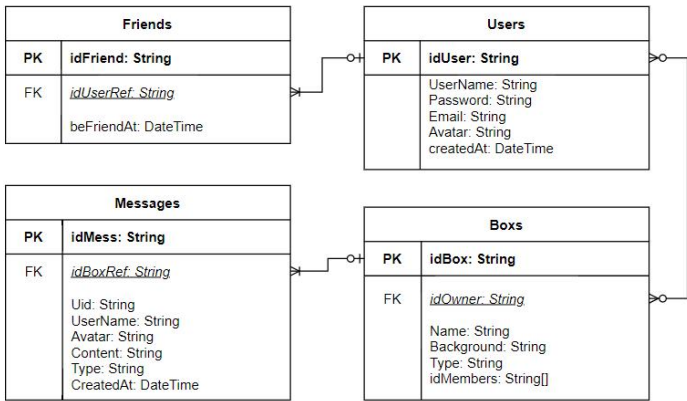
CHƯƠNG III: THIẾT KẾ HỆ THỐNG

3.1 Sơ đồ use case tổng quát



Hình 8. Sơ đồ use case Tổng quát

3.2 Sơ đồ ERD



Hình 9. Sơ đồ ERD

CHƯƠNG IV: HIỆN THỰC HỆ THỐNG VÀ KẾT QUẢ

Ở đây nhóm trình bày phần code phía backend. Phía frontend được dùng để client-side rendering các dữ liệu phù hợp.

4.1 Phần code khởi tạo

4.1.1 DataContext

```
public class DataContext : DbContext, IDataContext
{
    public DataContext(DbContextOptions<DataContext> options) : base(options) {}
    public DbSet<User> Users { get; set; }
    public DbSet<Inbox> Inboxes { get; set; }
    public DbSet<Message> Messages { get; set; }
    public DbSet<Friend> Friends { get; set; }
    public async Task<int> SaveChangesAsync(
        CancellationToken cancellationToken = default
    ){return SaveChanges();}
}
```

4.1.2 Entities

```
public record User
{
    public Guid UserId { get; init; }
    [Required]
    [MinLength(2), MaxLength(100)]
    public string UserName { get; set; }
    [MaxLength(200)]
    public string Avatar { get; set; }
    [Required]
    [EmailAddress]
    public string Email { get; init; }
    [Required]
    public string Password { get; set; }
    [Required]
    public DateTimeOffset CreatedAt { get; init; }
```

```
        public List<Friend> Friends { get; set; }  
    }
```

```
public record Message
```

```
{  
    public Guid MessageId { get; init; }  
    [Required]  
    public Guid Uid { get; init; }  
    [Required]  
    public string Username { get; set; }  
    public string Avatar { get; set; }  
    [Required]  
    public string Content { get; set; }  
    [Required]  
    public string Type { get; init; }  
    [Required]  
    public DateTime CreatedAt { get; init; }  
    [ForeignKey("FK_Inbox")]  
    public Guid InboxRefId { get; init; }  
}
```

```
public record Inbox
```

```
{  
    public Guid InboxId { get; init; }  
    public string Name { get; set; }  
    [Required]  
    public DateTime CreatedAt { get; init; }  
    public string Background { get; set; }  
    [Required]  
    public string Type { get; set; }  
    public string OwnerId { get; init; }  
    public Guid[] MemberIds { get; set; }  
    public List<Message> Messages { get; set; }  
}
```

4.1.3 Repositories

```
public class UserRepository : IUserRepository
{
    private readonly IDataContext _dataContext;

    public UserRepository(IDataContext dataContext)
    {
        _dataContext = dataContext;
    }

    public async Task<User> GetUserByEmail(string email)
    {
        return await _dataContext.Users.FirstOrDefaultAsync(
            user => user.Email == email
        );
    }

    public async Task<User> GetUserById(Guid id)
    {
        return await _dataContext.Users.FirstOrDefaultAsync(
            user => user.UserId == id
        );
    }

    public async Task CreateUser(User user)
    {
        _dataContext.Users.Add(user);
        await _dataContext.SaveChangesAsync();
    }

    public async Task UpdateUser(User user)
    {
        var userToUpdate = await _dataContext.Users.FirstOrDefaultAsync(
            u => u.UserId == user.UserId
        );
        if (userToUpdate is null)
        {
            throw new NullReferenceException();
        }
        _dataContext.Users.Update(user);
        await _dataContext.SaveChangesAsync();
    }
}
```

```
}  
}
```

```
public class MessageRepository : IMessageRepository  
{  
    private readonly IDataContext _dataContext;  
    public MessageRepository(IDataContext dataContext)  
    {  
        _dataContext = dataContext;  
    }  
    public async Task CreateMessage(Message message)  
    {  
        _dataContext.Messages.Add(message);  
        await _dataContext.SaveChangesAsync();  
    }  
}
```

```
public class InboxRepository : IInboxRepository  
{  
    private readonly IDataContext _dataContext;  
    public InboxRepository(IDataContext dataContext)  
    {  
        _dataContext = dataContext;  
    }  
    private List<Message> GetMessagesByInboxRefId(Guid id)  
    {  
        return _dataContext.Messages.Where(  
            message => message.InboxRefId == id  
        ).ToList();  
    }  
    public async Task<IEnumerable<InboxDto>> GetUserInboxes(Guid userId)  
    {  
        var inboxes = await _dataContext.Inboxes.Where(  
            inbox => inbox.MemberIds.Contains(userId)  
        ).ToListAsync();  
        var newInboxes = inboxes.Select(inbox => new InboxDto()  
        {
```

```

        InboxId = inbox.InboxId,
        Name = inbox.Name,
        Background = inbox.Background,
        MemberIds = inbox.MemberIds,
        OwnerId = inbox.OwnerId,
        Type = inbox.Type,
        CreatedAt = inbox.CreatedAt,
        Messages = inbox.Messages
    });
    IEnumerable<InboxDto> userInboxes = newInboxes.ToList();
    foreach (var inbox in userInboxes)
    {
        inbox.Messages = GetMessagesByInboxRefId(inbox.InboxId);
        if (inbox.Type == "personal")
        {
            List<UserDto> users = new List<UserDto>();

            foreach (var memberId in inbox.MemberIds)
            {
                User user = await _dataContext.
                    Users.FindAsync(memberId);
                UserDto newUser = new UserDto()
                {
                    UserId = user.UserId,
                    UserName = user.UserName,
                    Avatar = user.Avatar,
                    Email = user.Email,
                    CreatedAt = user.CreatedAt
                };
                users.Add(newUser);
            }
            inbox.Users = users;
        }
    }
    return userInboxes;
}

public async Task CreateInbox(Inbox inbox)

```

```

        {
            _dataContext.Inboxes.Add(inbox);
            await _dataContext.SaveChangesAsync();
        }
    }

    public class FriendRepository : IFriendRepository
    {
        private readonly IDataContext _dataContext;

        public FriendRepository(IDataContext dataContext)
        {
            _dataContext = dataContext;
        }

        public async Task AddFriend(Friend friend)
        {
            _dataContext.Friends.Add(friend);
            await _dataContext.SaveChangesAsync();
        }

        public async Task<IEnumerable<UserDto>> SearchForFriends(string keyword)
        {
            var matchedUsers = await _dataContext.Users.Where(
                user => user.UserName.ToLower().Trim().Contains(keyword)
            ).Select(user => new UserDto()
            {
                UserId = user.UserId,
                UserName = user.UserName,
                Avatar = user.Avatar,
                Email = user.Email,
                CreatedAt = user.CreatedAt
            }).ToListAsync();

            return matchedUsers;
        }

        public async Task<IEnumerable<FriendDto>> GetUserFriends(Guid id)
        {
            var friends = await _dataContext.Friends.Where(
                friend => friend.FriendId == id || friend.UserRefId ==
            id).ToListAsync();

            var friendsToReturn = friends.Select(friend => new FriendDto()

```



```

    {
        SenderId = friend.UserRefId,
        SenderData = new UserDto(),
        ReceiverId = friend.FriendId,
        ReceiverData = new UserDto(),
        BeFriendAt = friend.BeFriendAt
    }).ToList();
    foreach (var friend in friendsToReturn)
    {
        var senderFriendData = await _dataContext
            .Users.FirstOrDefaultAsync(
                user => user.UserId == friend.SenderId
            );
        var receiverFriendData = await _dataContext
            .Users.FirstOrDefaultAsync(
                user => user.UserId == friend.ReceiverId
            );
        friend.SenderData = new UserDto()
        {
            Avatar = senderFriendData.Avatar,
            CreatedAt = senderFriendData.CreatedAt,
            Email = senderFriendData.Email,
            UserName = senderFriendData.UserName,
            UserId = senderFriendData.UserId
        };
        friend.ReceiverData = new UserDto()
        {
            Avatar = receiverFriendData.Avatar,
            CreatedAt = receiverFriendData.CreatedAt,
            Email = receiverFriendData.Email,
            UserName = receiverFriendData.UserName,
            UserId = receiverFriendData.UserId
        };
    }
    return friendsToReturn;
}
}

```

4.1.4 Hubs

```
public class ChatHub : Hub<IChatClient>{}  
public interface IChatClient  
{  
    Task ReceiveMessage(Message message);  
    Task ReceiveInbox(Inbox inbox);  
}
```

4.1.5 Helpers

```
public class JwtService  
{  
    private string secretKey = "$2a$12$0HZrSq/jPMhyNNCnCwJmMe9cPPxgunxVE  
        h43pQk3oyvk4PxTv5F0.";   
    public string GenerateJWT(string id)  
    {  
        var symmetricSecurityKey =  
            new SymmetricSecurityKey(Encoding.UTF8.GetBytes(secretKey));  
        var signingCredentials = new SigningCredentials(symmetricSecurityKey,  
            SecurityAlgorithms.HmacSha256Signature);  
        var header = new JwtHeader(signingCredentials);  
        var payload = new JwtPayload(id, null, null, null,  
            DateTime.Today.AddDays(5));  
        var securityToken = new JwtSecurityToken(header, payload);  
        return new JwtSecurityTokenHandler().WriteToken(securityToken);  
    }  
    public JwtSecurityToken ValidateJWT(string jwt)  
    {  
        var handler = new JwtSecurityTokenHandler();  
        var key = Encoding.ASCII.GetBytes(secretKey);  
        handler.ValidateToken(jwt, new TokenValidationParameters  
        {  
            ValidateIssuerSigningKey = true,  
            IssuerSigningKey = new SymmetricSecurityKey(key),  
            ValidateIssuer = false,  
            ValidateAudience = false  
        },  
        );  
    }  
}
```

```

        out SecurityToken validatedToken);
        return validatedToken as JwtSecurityToken;
    }
}

```

4.1.6 Startup

```

app.UseCors(options => options
    .WithOrigins(new [] {"http://localhost:3000"})
    .AllowCredentials()
    .AllowAnyMethod()
    .AllowAnyHeader());
app.UseAuthorization();
app.UseEndpoints(endpoints =>
{
    endpoints.MapControllers();
    endpoints.MapHub<ChatHub>("/hubs/chat");
});

```

4.2 Phần code xử lý

Server sẽ nhận các request từ phía client thông qua API Controller và được điều hướng sang các endpoints cụ thể bằng các routes tương ứng.

4.2.1 AuthController

```

[HttpPost("signup")]
public async Task<ActionResult> SignUp(SignUpDto signUpDto)
{
    User existingUser = await _userRepository.GetUserByEmail(signUpDto.Email);
    if (existingUser != null)
    {
        return Ok(new { code = "error", message = "Email has been taken" });
    }
    User user = new()
    {
        UserId = Guid.NewGuid(),
        UserName = signUpDto.UserName,
        Email = signUpDto.Email,
    }
}

```

```

        Password = BCrypt.Net.BCrypt.HashPassword(signUpDto.Password),
        CreatedAt = signUpDto.CreatedAt
    };
    await _userRepository.CreateUser(user);
    return Ok(new { code = "success", message = "Sign up successful" });
}

```

```

[HttpPost("signin")]
public async Task<ActionResult> SignIn(SignInDto signInDto)
{
    User user = await _userRepository.GetUserByEmail(signInDto.Email);
    if (user is null || !BCrypt.Net.BCrypt.Verify(signInDto.Password, user.Password))
    {
        return Ok(new { code = "error", message = "Wrong email or password" });
    }
    var authJWT = _jwtService.GenerateJWT(user.UserId.ToString());
    Response.Cookies.Append("accessToken", authJWT, new CookieOptions() {
        HttpOnly = true,
        SameSite = SameSiteMode.None,
        Secure = true,
    });
    return Ok(new { code = "success", message = "Authorized" });
}

```

```

[HttpGet("signout")]
public ActionResult SignOut()
{
    Response.Cookies.Delete("accessToken");
    return Ok(new { code = "success", message = "Sign out successful" });
}

```

```

[HttpGet("user")]
public async Task<ActionResult<UserDto>> GetUser()
{
    try {
        var authJWT = Request.Cookies["accessToken"];
        var validatedJWT = _jwtService.ValidateJWT(authJWT);
    }
}

```

```

        User user = await _userRepository.GetUserById(
            Guid.Parse(validatedJWT.Issuer)
        );
        return Ok(new UserDto()
        {
            UserId = user.UserId,
            UserName = user.UserName,
            Email = user.Email,
            Avatar = user.Avatar,
            CreatedAt = user.CreatedAt
        });
    }
    catch (Exception ex) {
        return Unauthorized(new { code = "error", message = "Unauthorized" });
    }
}

```

4.2.2 FriendController

```

[HttpGet]
public async Task<ActionResult<FriendDto>> GetUserFriends()
{
    try {
        var authJWT = Request.Cookies["accessToken"];
        var validatedJWT = _jwtService.ValidateJWT(authJWT);
        IEnumerable<FriendDto> userFriends = await
            friendRepository.GetUserFriends(Guid.Parse(validatedJWT.Issuer));
        return Ok(new { code = "success", friends = userFriends });
    }
    catch (Exception ex) {
        return Unauthorized(new { code = "error", message = "Unauthorized" });
    }
}

[HttpPost("search")]
public async Task<ActionResult<UserDto>> SearchForFriends(string keyword)
{

```

```

    try {
        var authJWT = Request.Cookies["accessToken"];
        var validatedJWT = _jwtService.ValidateJWT(authJWT);
        IEnumerable<UserDto> results = await _friendRepository.
            SearchForFriends(keyword);
        return Ok(new { code = "success", results });
    }
    catch (Exception ex) {
        return Unauthorized(new { code = "error", message = "Unauthorized" });
    }
}

```

```

[HttpPost("add")]
public async Task<ActionResult> AddFriend(Friend friend)
{
    try {
        var authJWT = Request.Cookies["accessToken"];
        var validatedJWT = _jwtService.ValidateJWT(authJWT);
        Friend friendToAdd = new Friend()
        {
            FriendId = friend.FriendId,
            BeFriendAt = DateTime.Now,
            UserRefId = friend.UserRefId
        };
        await _friendRepository.AddFriend(friendToAdd);
        return Ok(new { code = "success", message = "Add friend successful" });
    }
    catch (Exception ex) {
        return Unauthorized(new { code = "error", message = "Unauthorized" });
    }
}

```

4.2.3 InboxController

```

[HttpGet("user")]
public async Task<ActionResult<InboxDto>> GetUserInboxes()
{

```

```

    try {
        var authJWT = Request.Cookies["accessToken"];
        var validatedJWT = _jwtService.ValidateJWT(authJWT);
        IEnumerable<Inbox> inboxes = await _inboxRepository.GetUserInboxes(
            Guid.Parse (validatedJWT.Issuer)
        );
        return Ok(new
        {
            code = "success",
            message = "Retrieve user inboxes successful",
            inboxes
        });
    }
    catch (Exception ex) {
        return Unauthorized(new { code = "error", message = "Unauthorized" });
    }
}

```

```

[HttpPost("create")]
public async Task<ActionResult> CreateInbox(CreateInboxDto createInboxDto)
{
    try {
        var authJWT = Request.Cookies["accessToken"];
        var validatedJWT = _jwtService.ValidateJWT(authJWT);
        Inbox inboxToCreate;
        if (createInboxDto.Type == "Group")
        {
            var uploadFile = await UploadFile(createInboxDto.File);
            if (uploadFile.Code == "error")
            {
                return Ok(new {
                    code = "error", message = uploadFile.Message
                });
            }
            inboxToCreate = new()
            {
                InboxId = Guid.NewGuid(),

```

```

        Name = createInboxDto.Name,
        Background = uploadFile.FileName,
        Type = createInboxDto.Type,
        CreatedAt = DateTime.Now,
        OwnerId = createInboxDto.OwnerId,
        MemberIds = createInboxDto.MemberIds
    };
}
else
{
    inboxToCreate = new()
    {
        InboxId = Guid.NewGuid(),
        Name = createInboxDto.Name,
        Background = "No background",
        Type = createInboxDto.Type,
        CreatedAt = DateTime.Now,
        OwnerId = createInboxDto.OwnerId,
        MemberIds = createInboxDto.MemberIds
    };
}
await _inboxRepository.CreateInbox(inboxToCreate);
await _chatHub.Clients.All.ReceiveInbox(inboxToCreate);
return Ok(new {
    code = "success", inbox = inboxToCreate,
    message = "Create inbox successful"
});
}
catch (Exception ex) {
    return Unauthorized(new { code = "error", message = "Unauthorized" });
}
}

[NonAction]
private async Task<UploadDto> UploadFile(IFormFile file)
{
    var fileExtension = Path.GetExtension(file.FileName);

```



```

var acceptedExtension = new[]
{
    ".png",
    ".jpg",
    ".jpeg"
};
if (file.Length > 10000000)
{
    return new UploadDto()
    {
        Code = "error",
        FileName = "",
        Message = "The file size upload is over 10mb"
    };
}
if (!acceptedExtension.Contains(fileExtension))
{
    return new UploadDto()
    {
        Code = "error",
        FileName = "",
        Message = "The file extension is not supported"
    };
}
string fileName = new string(Path.GetFileNameWithoutExtension(
    file.FileName
).ToArray()).Replace(' ', '-');
fileName += "-" + DateTime.Now.ToString("yy-mm-ss-fff") +
    Path.GetExtension(file.FileName);
var filePath = Path.Combine(
    _webHostEnvironment.ContentRootPath, "Resources", fileName
);
await using var stream = new FileStream(filePath, FileMode.Create);
await file.CopyToAsync(stream);
return new UploadDto()
{
    Code = "success",

```

```

        FileName = fileName
    };
}

```

4.2.4 MessageController

```

[HttpPost("create"), DisableRequestSizeLimit]
public async Task<ActionResult> CreateMessage([FromForm] CreateMessageDto
createMessageDto)
{
    try {
        var authJWT = Request.Cookies["accessToken"];
        var validatedJWT = _jwtService.ValidateJWT(authJWT);
        Message messageToCreate;
        if (createMessageDto.Type == "image" ||
            createMessageDto.Type == "video" ||
            createMessageDto.Type == "document"
        )
        {
            var uploadFile = await UploadFile(createMessageDto.File);
            if (uploadFile.Code == "error")
            {
                return Ok(new {
                    code = "error", message = uploadFile.Message
                });
            }
            messageToCreate = new()
            {
                MessageId = Guid.NewGuid(),
                Uid = createMessageDto.Uid,
                Username = createMessageDto.Username,
                Avatar = createMessageDto.Avatar,
                Content = uploadFile.FileName,
                Type = createMessageDto.Type,
                CreatedAt = DateTime.Now,
                InboxRefId = createMessageDto.InboxRefId
            };

```

```

    }
    else
    {
        messageToCreate = new()
        {
            MessageId = Guid.NewGuid(),
            Uid = createMessageDto.Uid,
            Username = createMessageDto.Username,
            Avatar = createMessageDto.Avatar,
            Content = createMessageDto.Content,
            Type = createMessageDto.Type,
            CreatedAt = DateTime.Now,
            InboxRefId = createMessageDto.InboxRefId
        };
    }

    await _messageRepository.CreateMessage(messageToCreate);
    await _chatHub.Clients.All.ReceiveMessage(messageToCreate);
    return Ok(new { code = "success", message = messageToCreate });
}

catch (Exception ex) {
    return Unauthorized(new { code = "error", message = "Unauthorized" });
}
}

```

[NonAction]

private async Task<UploadDto> UploadFile(IFormFile file)

```

{
    var fileExtension = Path.GetExtension(file.FileName);
    var acceptedExtension = new[]
    {
        ".mp4",
        ".png",
        ".jpg",
        ".jpeg",
        ".pdf",
        ".doc",
        ".docx",
    }
}

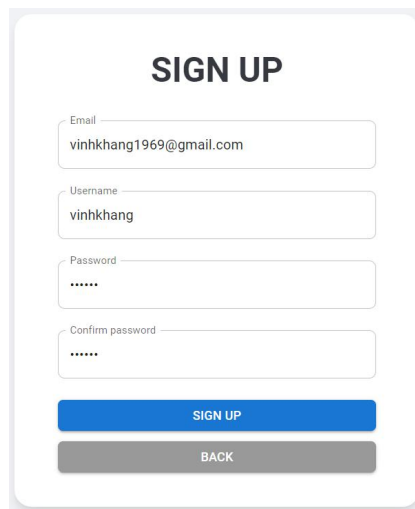
```

```

        ".xls",
        ".xlsx",
        ".txt",
        ".zip",
        ".rar"
    };
    if (file.Length > 10000000)
    {
        return new UploadDto()
        {
            Code = "error",
            FileName = "",
            Message = "The file size upload is over 10mb"
        };
    }
    if (!acceptedExtension.Contains(fileExtension))
    {
        return new UploadDto()
        {
            Code = "error",
            FileName = "",
            Message = "The file extension is not supported"
        };
    }
    string fileName = new string(Path.GetFileNameWithoutExtension(file.FileName)
        .ToArray()).Replace(' ', '-');
    fileName += "-" + DateTime.Now.ToString("yymmssfff") +
        Path.GetExtension(file.FileName);
    var filePath = Path.Combine(
        _webHostEnvironment.ContentRootPath, "Resources", fileName
    );
    await using var stream = new FileStream(filePath, FileMode.Create);
    await file.CopyToAsync(stream);
    return new UploadDto() { Code = "success", FileName = fileName };
}

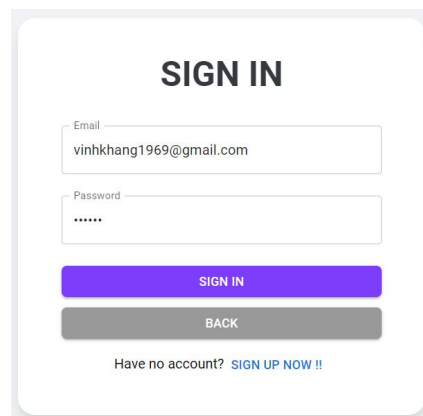
```

4.3 Kết quả đạt được



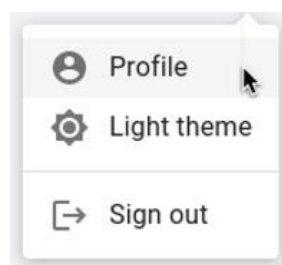
A registration form titled "SIGN UP". It contains four input fields: "Email" with the value "vinhkhong1969@gmail.com", "Username" with the value "vinhkhong", "Password" with masked characters "*****", and "Confirm password" with masked characters "*****". Below the fields are two buttons: a blue "SIGN UP" button and a grey "BACK" button.

Hình 10. Hình ảnh về Đăng ký



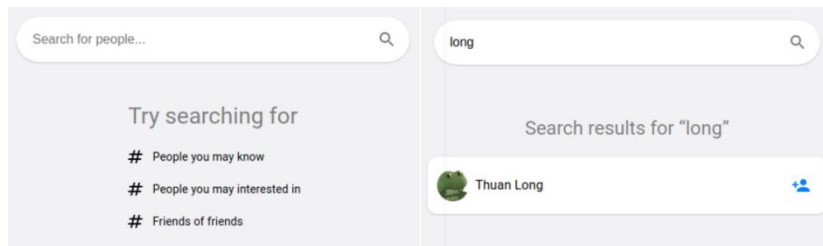
A login form titled "SIGN IN". It contains two input fields: "Email" with the value "vinhkhong1969@gmail.com" and "Password" with masked characters "*****". Below the fields are two buttons: a purple "SIGN IN" button and a grey "BACK" button. At the bottom, there is a link: "Have no account? [SIGN UP NOW !!](#)".

Hình 11. Hình ảnh về Đăng nhập

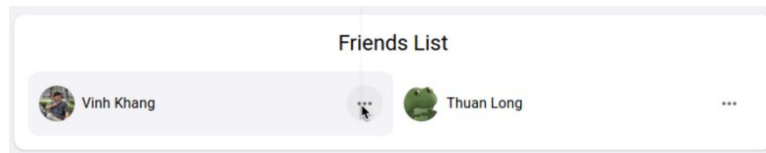


A user menu with three items: "Profile" with a person icon, "Light theme" with a gear icon, and "Sign out" with a door icon. A mouse cursor is hovering over the "Profile" item.

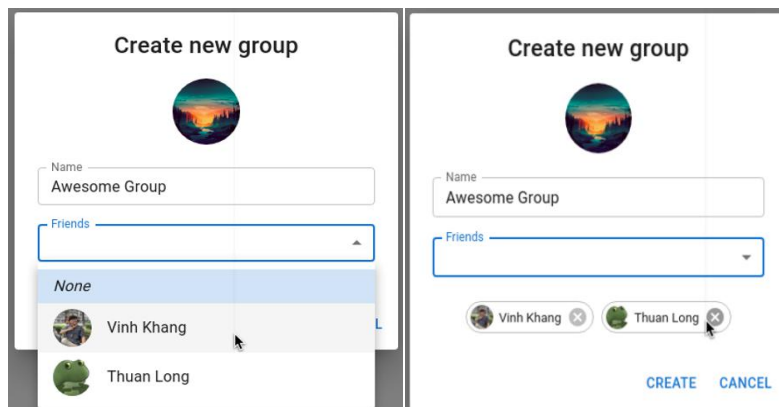
Hình 12. Hình ảnh về Đăng xuất



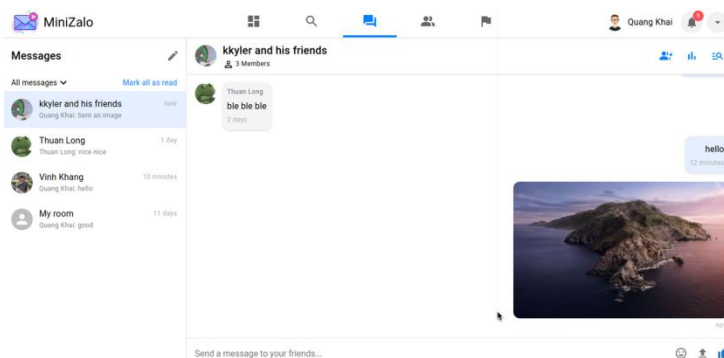
Hình 13. Hình ảnh về Tìm kiếm người dùng để kết bạn



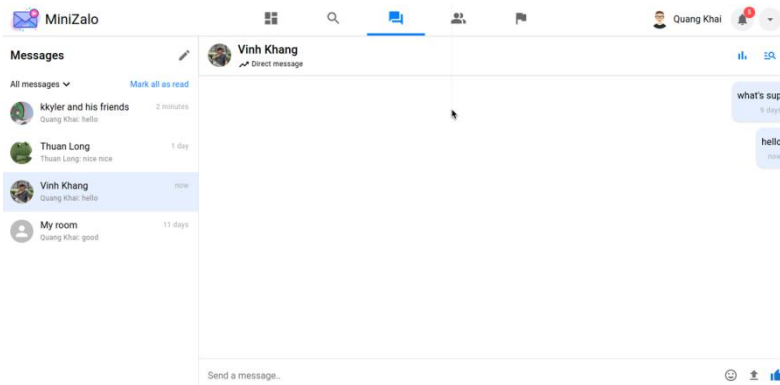
Hình 14. Hình ảnh về Danh sách sau khi kết bạn



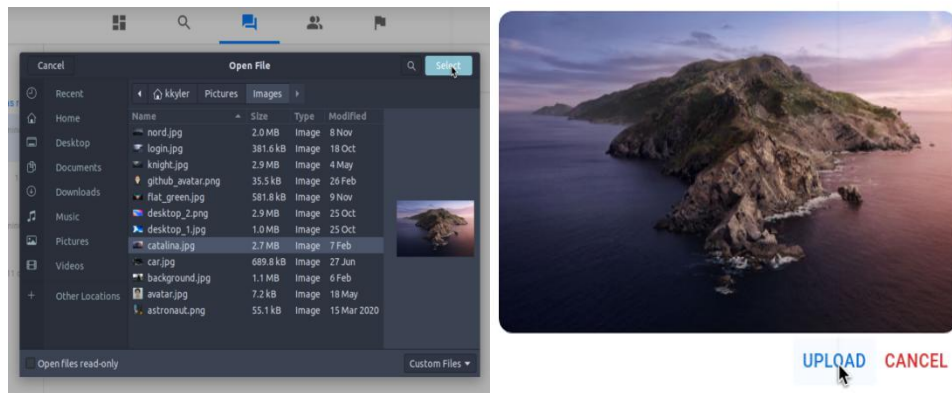
Hình 15. Hình ảnh về Tạo nhóm chat



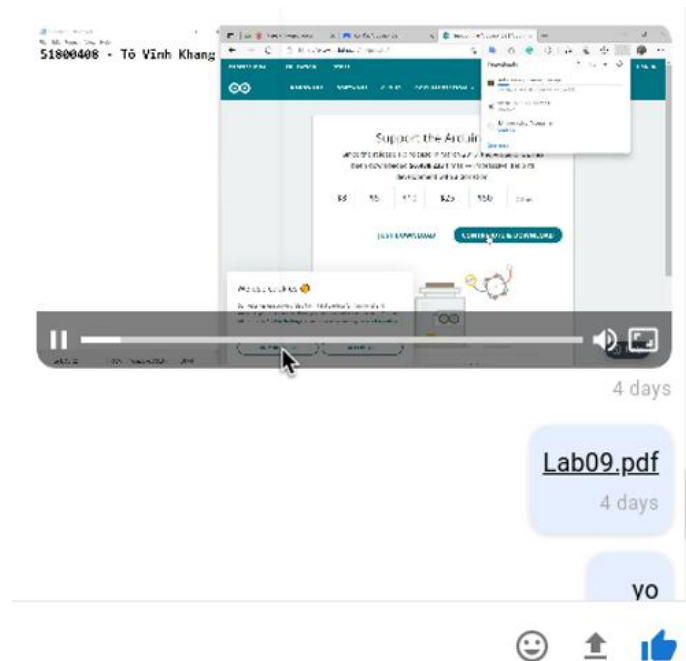
Hình 16. Hình ảnh về Trò chuyện nhóm



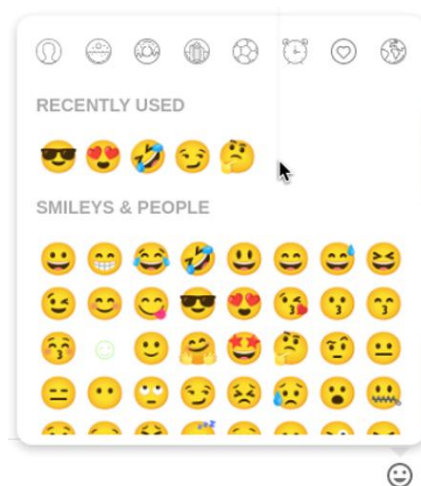
Hình 17. Hình ảnh về Trò chuyện cá nhân



Hình 18. Hình ảnh về Gửi file hình



Hình 19. Hình ảnh về Gửi file tài liệu và video



Hình 20. Hình ảnh về Các emoji

CHƯƠNG IV: TỔNG KẾT

Bài báo cáo này đã phân tích yêu cầu và thiết kế một website thời gian thực, phía back-end sử dụng thư viện SignalR kết hợp nền tảng ASP.NET và lưu trữ dữ liệu thông qua PostgreSQL. Sau đó phía front-end sử dụng React, Redux, TypeScript, MaterialUI để hiển thị dữ liệu lên cho người dùng với mức độ trải nghiệm người dùng đạt được hiệu quả cao nhất. Sơ đồ usecases và ERD đã phần nào miêu tả khái quát hóa các chức năng cốt lõi của hệ thống, cách thức hoạt động của từng chức năng và một số ràng buộc quan trọng.

Với đề tài “Xây dựng Web chat theo thời gian thực sử dụng SignalR kết hợp ASP.NET”, nhóm đã học hỏi thêm được rất nhiều thông qua việc tìm hiểu từ các tài liệu từ nhiều nguồn, các chức năng hỗ trợ tiện lợi từ nền tảng có sẵn để góp phần đưa ra ý tưởng cho việc thiết kế website này qua cái nhìn tổng thể, bao quát nhất. Việc triển khai website này giúp cho việc trao đổi qua lại giữa các bạn sinh viên trong việc học tập trở nên dễ dàng hơn. Tuy nhiên, một vài chức năng vẫn còn chưa hoàn hảo, bên cạnh đó nếu deploy thực tế cần có server mạnh để việc truyền tải dữ liệu được nhanh chóng. Ứng dụng này về cơ bản đã có giao diện với màu sắc và bố cục sắp xếp các thành phần mà theo cảm nhận của nhóm là phù hợp và thân thiện với người dùng.

Trong tương lai, nhóm sẽ tiếp tục phát triển cải thiện hiệu suất các tính năng đang làm và bổ sung thêm một số tính năng mở rộng khác như thông báo, gửi tin nhắn ghi âm, chia sẻ vị trí hiện tại, thả cảm xúc tin nhắn, đánh dấu tin nhắn đã đọc, xem thông tin cá nhân,... cũng như sử dụng Netlify và Railway lần lượt để hosting phía frontend và backend. Việc tạo một web thực tế để nhiều người truy cập sẽ làm cho việc phản hồi về dữ liệu đôi khi bị delay nhưng chức năng vẫn đảm bảo hoạt động tốt. Với việc nhu cầu sử dụng website nhắn tin trực tuyến này đang ngày càng được ưa chuộng, việc tìm hiểu, nghiên cứu và triển khai các ứng dụng tương tự như vậy sẽ giúp cho lĩnh vực phần mềm ngày càng phát triển cũng như những mong đợi mà nhóm đang theo đuổi.

PHÂN CÔNG CÔNG VIỆC VÀ ĐÁNH GIÁ

Phân công công việc

Họ và tên	Thời gian	Mô tả
Tô Vĩnh Khang (Trưởng nhóm)	26/10/2021-31/10/2021	Thảo luận nhóm, đăng ký đề tài, lên ý tưởng, phân công công việc cho từng thành viên nhóm. Tìm hiểu cách sử dụng SignalR trong ASP.NET.
	31/10/2021-15/11/2021	Trình bày báo cáo Chương I, Chương II (2.1) và Chương III (3.1). Xử lý phía front-end chính và hỗ trợ xử lý phía back-end.
	15/11/2021-25/11/2021	Trình bày báo cáo Chương IV(4.1), Chương V. Tổng hợp nội dung từ các thành viên và Trình bày slide thuyết trình.
	25/11/2021-29/11/2021	Thực hiện UAT nội bộ để kiểm tra và sửa chữa các thiết sót của hệ thống. Nộp bài lên hệ thống.
Bùi Quang Khải	26/10/2021-31/10/2021	Thảo luận nhóm, lên ý tưởng. Tìm hiểu cách sử dụng SignalR trong ASP.NET.

	31/10/2021-15/11/2021	Thiết lập môi trường. Xử lý phía front-end chính và hỗ trợ xử lý phía back-end.
	15/11/2021-25/11/2021	Trình bày báo cáo Chương III (3.2), Chương IV(4.2).
	25/11/2021-28/11/2021	Thực hiện UAT nội bộ để kiểm tra và sửa chữa các thiết sót của hệ thống.
Du Thuận Long	26/10/2021-31/10/2021	Thảo luận nhóm, lên ý tưởng. Tìm hiểu cách sử dụng SignalR trong ASP.NET.
	31/10/2021-15/11/2021	Xử lý phía backend chính và hỗ trợ xử lý phía front-end.
	15/11/2021-25/11/2021	Trình bày báo cáo Chương II (2.2), Chương IV(4.3).
	25/11/2021-28/11/2021	Thực hiện UAT nội bộ để kiểm tra và sửa chữa các thiết sót của hệ thống.

Bảng phân công nhiệm vụ

Đánh giá thành viên

Họ và tên	Tô Vĩnh Khang	Bùi Quang Khải	Du Thuận Long
Mức độ hoàn thành	100%	100%	100%

Bảng đánh giá các thành viên trong nhóm

TÀI LIỆU THAM KHẢO

- [1] V.D. Hong, [2021], Slide Lecture Advanced Software Engineering, Ton Duc Thang University, Vietnam.
- [2] D. Roth, R. Anderson, [2021], Introduction to ASP.NET Core, USA.
- [3] J. M. Aguilar, [2014], SignalR Programming in Microsoft ASP.NET, USA.
- [4] K. Nayyeri, D. White, [2014], Pro ASP.NET SignalR, USA.
- [5] A. Verma, Advanced Software Engineering, Jharkhand Rai University, India.
- [6] C. Rippon, [2021], ASP.NET Core 5 and React: Full-stack web development using .NET 5, React 17, and TypeScript 4 2nd Edition, UK.
- [7] A. Boduch, [2019], React Material UI Cookbook, UK.
- [8] A. Banks, E. Porcello, Learning React - Functional Web Development with React and Redux, USA.
- [9] J.W. Satzinger, R.B. Jackson, S.D. Burd, [2011], Systems Analysis and Design in a Changing World, Australia.
- [10] Zalo | <https://zalo.me/features.html>