



TRƯỜNG ĐẠI HỌC FPT

MINISTRY OF EDUCATION AND TRAINING

Capstone Project Document

TripSharing

Group	
Group Members	Hà Văn Thái – SE04801 Nguyễn Văn Phong – SE05051 Trần Văn Phong – SE05048 Lê Xuân Trường – SE04616 Lý Phúc Linh – SE04693
Supervisor	Đào Trọng Duy
Capstone Project Code	TripSharing

Hanoi, August 28th, 2019

This page is intentionally left blank

Table of Contents

Acknowledgements	7
Definitions and Acronyms	8
Chapter 1: Introduction	9
1.1 Purpose.....	9
1.2 Project Information	9
1.3 The People	9
1.3.1 Supervisors	9
1.3.2 Team Members.....	9
1.4 Background	9
1.4.1 The exist sharing travel experience systems are focus on famous travel destination and booking services	11
1.4.2 TripAdvisor's Advantages and Disadvantages	12
1.5 Proposal of System.....	13
1.5.1 Our Proposal System	13
Chapter 2: Project plan	17
2.1 Purpose.....	17
2.2 Project Organization.....	17
2.2.1 Software Development Process	17
2.2.2 Role and Responsibilities	18
2.2.3 Organization Structure	19
2.2.4 Project Team Member	19
2.2.5 Tools and Techniques.....	20
2.3 Project Management Plan	20
2.3.1 Tasks	20
2.3.2 Meeting Minutes.....	21
2.3.3 Code Conventions	22
2.3.4 Risk Management Plan	23
2.3.5 Communication Plan.....	24
Chapter 3: Software Requirement Specification	27
3.1 Purpose.....	27
3.2 Functional Requirements.....	27
3.2.1 Use Case Diagram	27
3.2.2 Business Rules	27

3.2.3 Use Cases	28
3.3 Non-Functional Requirements	78
3.3.1 Security	78
3.3.2 Maintainability & Extensibility	81
3.3.3 Availability and Scalability	81
3.3.4 Performance.....	81
3.3.5 Usability.....	81
Chapter 4: Software Design	82
4.1 Purpose.....	82
4.2 Architecture Overview.....	82
4.2.1 System Architecture	82
4.2.2 System Architecture Explanation	82
4.3.2 Database Design	87
4.3.3 Common Design	97
4.3.4 Detail Design.....	97
Chapter 5: Software Testing Documentation	281
5.1 Introduction	281
5.1.1 Purpose.....	281
5.1.2 Scope of testing.....	281
5.2 Test plan.....	281
5.2.1 Testing tools and environment.....	281
5.2.2 Resources and responsibilities	283
5.2.3 Test strategy	283
5.2.4 Features to be tested.....	286
5.2.5 Features not to be tested	286
5.3 Test Approach.....	286
5.3.1 Unit testing	286
5.3.2 Integration testing and System test.....	290
5.3.4 Acceptance Test.....	290
5.3.5 Defect Log.....	292
5.4 Test Report.....	293
5.4.1 Unit test report	293
5.4.2 Integration test report.....	295

5.4.3 System test report.....	296
Chapter 6: User manual	298
6.1 Deployment guidelines	298
6.1.1 Environment for development	298
6.1.2 Environment for deployment.....	304
6.2 User guidelines	308
6.2.1 User sign up	308
6.2.2 User sign in	310
6.2.3 User sign out.....	310
6.2.4 User update profile.....	311
6.2.5 User searches posts.....	312
6.2.6 User searches other users.....	312
6.2.7 User creates posts	313
6.2.8 User chats with other users	315
6.2.9 User joins a group	316
6.2.10 User follows other users and views followings/followers	317
6.2.11 User bookmarks posts and view bookmarked posts.....	318

This page is intentionally left blank

Acknowledgements

We wish to express our deepest gratitude to our supervisor, Mr. Dao Trong Duy, for his continuously sharing and motivating throughout the project. Following strictly Mr. Duy instructions, the TripSharing team has always felt confident in dealing with problems along the way. Taking aside all the technologies and methodologies, Mr. Duy induced us with the idea of having the right attitude & resolution towards tackling obstacles. That was the most important factor that has led us to the completion of this project.

Finally, we truly appreciate the instructors at FPT University for all the lectures & knowledge shared to us. We hope you will find this project as a reflection of the knowledge and experiences you have given us during this period of three years.

Definitions and Acronyms

Acronym	Definition	Note
API	Application Programming Interface	
JSON	JavaScript Object Notation	
URL	Uniform Resource Locator	
BSON	Binary JSON	
GUI	Graphical User Interface	
UML	Unified Modeling Language	
SDK	Software Development Kit	
OOP	Object-oriented programming	
CLI	Command-line interface	

Chapter 1: Introduction

1.1 Purpose

This chapter provide an overview of the project include background information, a literature review of existing system and raising a proposal for ideas of improvement.

1.2 Project Information

- Project name: **TripSharing**
- Project code: **TRIPSHARING**
- Project group name: **SWP491_G26**
- Project type: **Web Application**
- Timeline: **From 13th May 2019 to 29th August 2019**

1.3 The People

1.3.1 Supervisors

	Full name	Phone	E-mail	Title
Supervisor	Đào Trọng Duy	0983204196	duydt@fe.edu.vn	Lecturer

Table 1-1: Supervisors' information

1.3.2 Team Members

	Full name	Phone	E-mail	Role in Group
1	Hà Văn Thái	0349940158	thaihvse04801@fpt.edu.vn	Leader
2	Nguyễn Văn Phong	0978296187	phongnvse05051@fpt.edu.vn	Member
3	Trần Văn Phong	0987489397	phongtvse05048@fpt.edu.vn	Member
4	Lê Xuân Trường	0964038801	truonglxse04616@fpt.edu.vn	Member
5	Lý Phúc Linh	0362442818	linhlpse04693@fpt.edu.vn	Member

Table 1-2: Team member's information

1.4 Background

Along with the development of technology, tourism demand is getting higher, especially young people. Nowadays, people mostly find information about where they want to go on the internet and social network like Facebook, TripAdvisor, Gody.vn, etc. But many websites only

provide general information about the destination and only focus on advertising and support booking hotel, restaurant, travel tour.... We can find such post on Facebook but it scattered on many pages and there are many posts created for advertising purpose may make people concern about the correctness of the information.

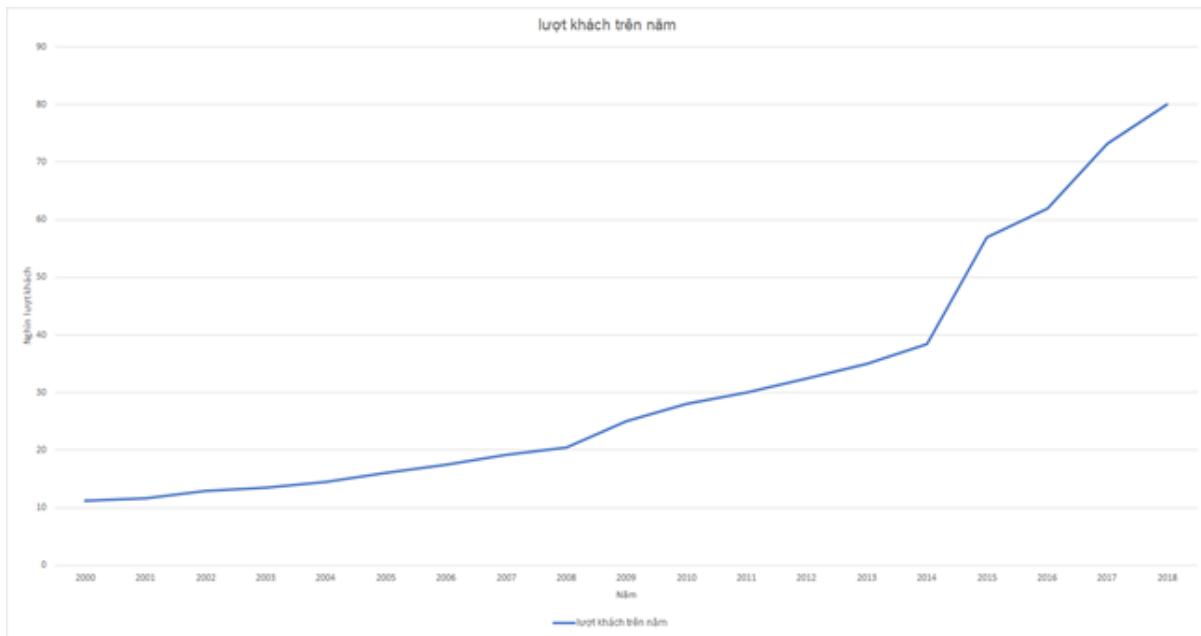


Figure 1-1: Number of local tourists reach 80 million in 2018¹

Among young people, tourism for exploring and experience is becoming a new trend, and people usually want to share their experience online:

“Khảo sát về xu hướng du lịch toàn cầu trong năm 2018 do Visa thực hiện với sự tham gia của hơn 15.000 người đến từ 27 quốc gia, trong đó có Việt Nam đã chỉ ra rằng khách du lịch hiện nay thường mong muốn đạt được cả hai mục tiêu “khám phá” và “tận hưởng” trong những chuyến đi của mình. Kết quả khảo sát đã đưa ra ba nhóm động lực chính cho những chuyến du lịch là: gắn kết gia đình, bạn bè (33%), thư giãn (11%) và trải nghiệm (10%). Ranh giới giữa những động lực thúc đẩy người dân đi du lịch đang dần bị xóa nhòa. Những người du lịch vì cảm giác “tận hưởng” thường đi để gắn kết với bạn bè, người thân hay đơn giản là tận hưởng thời gian nghỉ dưỡng của riêng mình. Riêng những du khách đi du lịch để “khám phá” có xu hướng trải nghiệm nhiều hơn với các nền văn hóa mới và thăm quan những điểm đến hấp dẫn.”

¹ <http://vietnamtourism.gov.vn/index.php/items/13460>

63% người được hỏi trả lời rằng họ đi du lịch vì cả hai lý do này, tỉ lệ tương ứng tại Việt Nam là 72%.²



Figure 1-2: Sharing travel information on social media³

1.4.1 The exist sharing travel experience systems are focus on famous travel destination and booking services

Our study shows that travel guides, travel destinations reviews websites are focusing too much on advertising and booking for travel destination than the travel experience of users. Example on Tripadvisor.com and Toidi.net are 2 big websites, but they don't provide enough interaction between users, Toidi.net is more like a blog then the posts are subjective, on Tripadvisor.com people can post photo or review for a destination but there is no comment

² <http://flcholiday.com/tin-tuc/xu-huong-du-lich-2018-kham-pha-va-tan-huong-cuoc-song-117.htm>

³ <http://www.younetmedia.com/insights/infographic-xu-huong-du-lich-cua-gioi-tre-qua-phan-tich-tren-social-media.html>

section for other to interact with each other. The creation of this project is aiming to improve the lack of interaction between users and focus more on sharing experiences through the trip.

1.4.2 TripAdvisor's Advantages and Disadvantages

1.4.2.1 TripAdvisor's Advantages

- They already have a booking system.

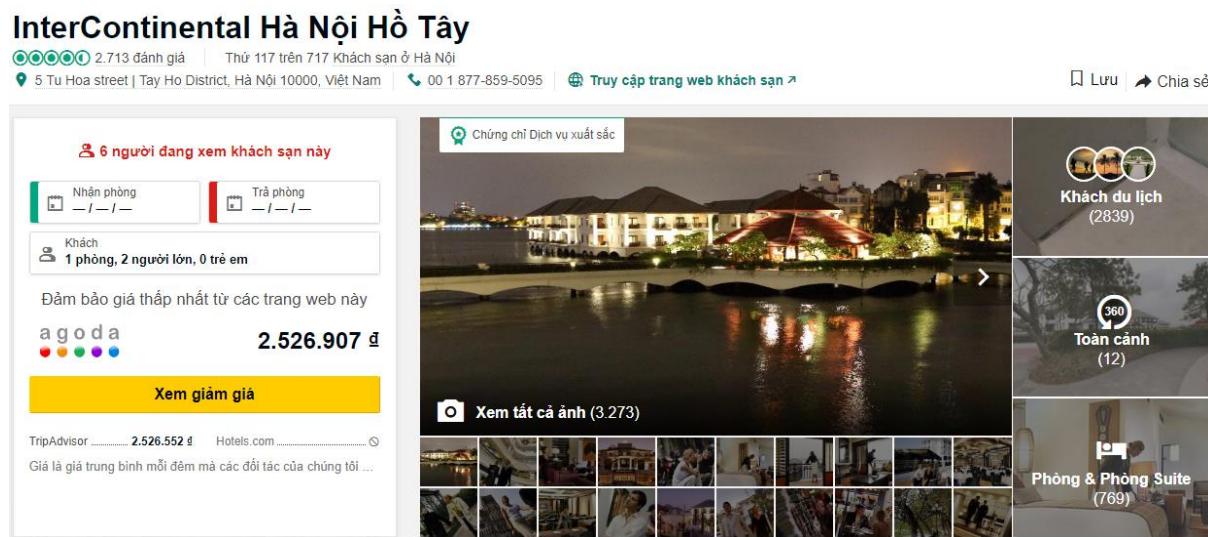


Figure 1-3: Booking function on TripAdvisor

- User can create a virtual trip

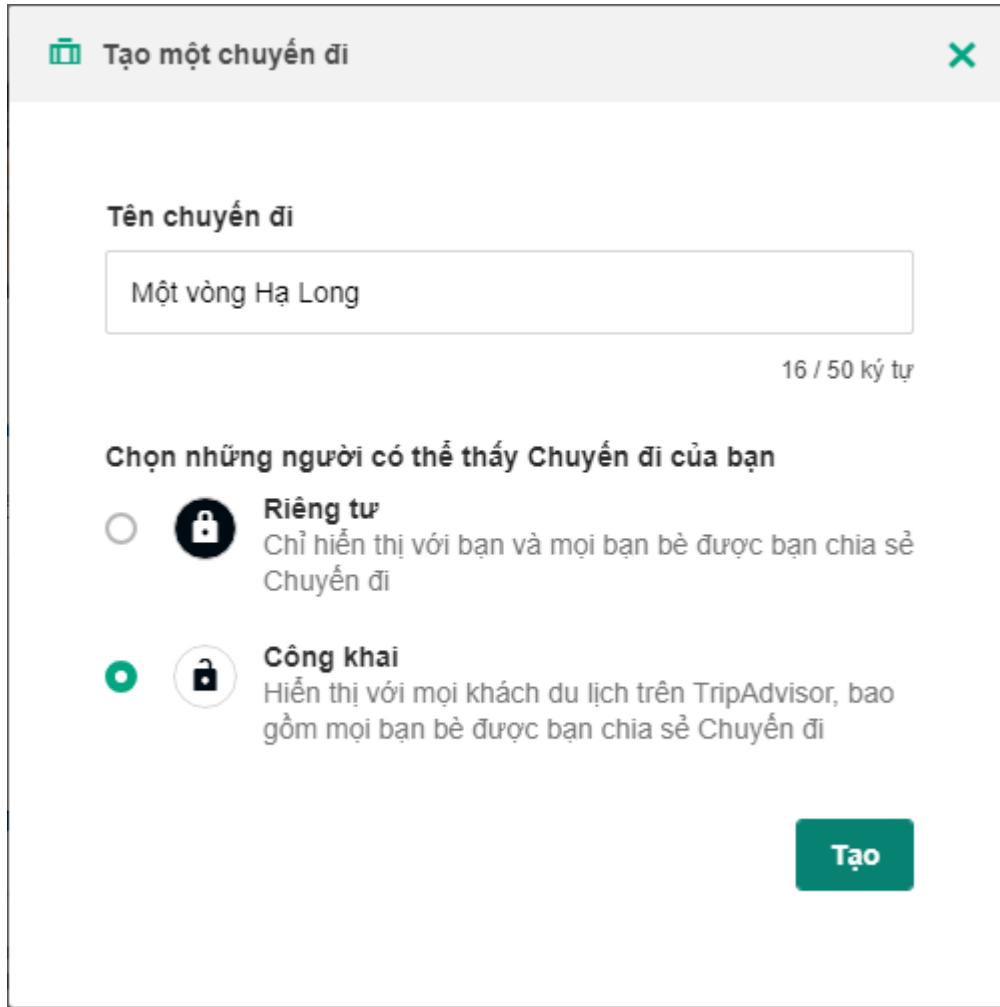


Figure 1-4: Creating a virtual trip function on TripAdvisor

1.4.2.2 TripAdvisor's Disadvantages

- User can't comment on other's posts or photos so they are lack of interaction between users.
- Don't support users for writing a long sharing experience post, like blog.

1.5 Proposal of System

1.5.1 Our Proposal System

After reviewing all properties of the existing systems as well as the travel trend of young people in Viet Nam, we have come to decision to develop a travel sharing system which allows

travelers to share their travel experiences by images, articles or blogs. It will make easier for other travelers to prepare their plans. The purpose of the system is creating a travel community to exchange information. The system is going to encourage travelers to share their experiences by recording their contributions as points and rank all contributors. If contributors have a high rank, they will receive many incentives from the system.

The system does not focus on booking services. A common trip includes three stages. The first stage is planning, the second is experiencing, and the last is sharing. Our system is going to focus on the sharing stage.

1.5.1.1 System Functions

- ❖ Allow users to register/sign in with email address.
- ❖ Allow users to create/edit their own profile.
- ❖ Allow users to search travel destination.
- ❖ Allow users to choose interesting topic.
- ❖ Allow users to create/edit/delete their post.
- ❖ Allow users to like/unlike a post.
- ❖ Allow users to share a post to social media.
- ❖ Allow users to bookmark a post.
- ❖ Allow users to comment to a post.
- ❖ Allow users to report a post/comment.
- ❖ Allow users to create/edit/delete their virtual trips.
- ❖ Allow users to follow/unfollow other users.
- ❖ Allow users to create/edit/delete their finding companion posts.
- ❖ Allow users to join a finding companion group.
- ❖ Allow users to send/receive messages to other users.
- ❖ Allow Administrators to manage users' account, users' posts, users' comments.

1.5.1.2 The TripSharing User Process

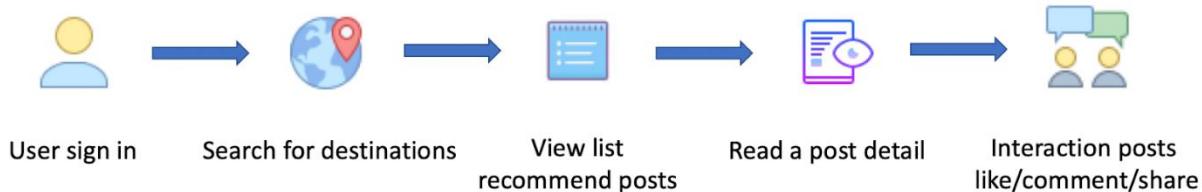


Figure 1-5: User reads and interacts with a post

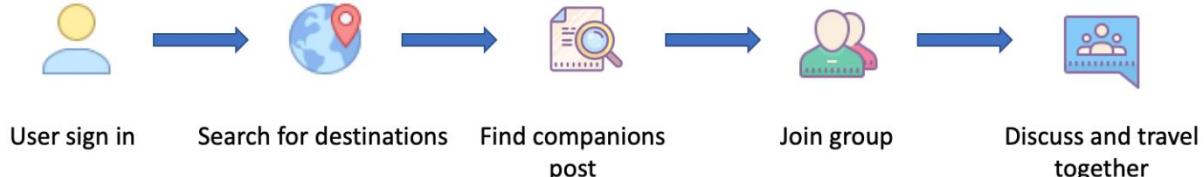


Figure 1-6: Users use Finding Companions function

1.5.1.3 Out of Scope Functions

- ❖ Manage group of people who use Finding Companions function.
- ❖ Manage travel places obtained from Google Map.

1.5.1.4 Special Approaches

- ❖ Using Microservices architecture for application's deployment.
- ❖ Using Docker to compose the project into containers to deploy on Google Cloud.
- ❖ Using Google Cloud Kubernetes Engine as deployment host for scaling and load balancing.

- ❖ Use Git as source code version control, hosted on Github.
- ❖ Having a separate background service to handle heavy tasks.
- ❖ Using Google Pub Sub for communicating between microservices.
- ❖ Using Google Cloud Storage for images.
- ❖ For web frontend system:
 - Using Angular 7 for web component rendering.
 - Using Angular Material to create all web form components.

Chapter 2: Project plan

2.1 Purpose

This chapter provides an overview of the project plan includes project organization and project management plan

2.2 Project Organization

2.2.1 Software Development Process



Figure 2-1: Iterative and Incremental Software Development Process⁴

TripSharing project uses the Iterative and Incremental Software Process Model as shown in the figure above, which describes the overall lifecycle process.

The Iterative and Incremental Software Process Model is mostly used when the scope of the project is big, the major requirements are defined clearly, some more details will be added later. By using this process model, we break down the developing system tasks into series of smaller tasks which will complete separately, evaluated, and subsequently re-worked until the system perform adequately. In addition, the iterative model is easier than other models when the issues are discovered. The feedbacks are immediately given, and solutions are proposed right on the spot.

⁴ <https://www.topsinfsolutions.com/methodology/>

2.2.2 Role and Responsibilities

Role	Responsibilities
Project Manager	Planning, developing schedules, coordinating communication, generally responsible for keeping the team's focus on the main goal
Business Analysis Leader	Responsible for managing Business Analyst team and working with them in Analyzing an organization or business domain.
Business Analyst	Analyzes an organization or business domain and documents its business or processes or systems.
Design Leader	Responsible for managing Design team and working with them to create a best user interface design for our product.
Designer	Involve to design product's user interface.
Technical Leader	Responsible for choosing and deciding what technologies should be used, as well as for overseeing the work being done by other developers.
Developer	Involve to code the product and review code of other developers.
Test Leader	Responsible for choosing and deciding what testing tools and techniques should be used, as well as keep the progress of test team on track.
Tester	Involve testing the product.

Table 2-1: Role and Responsibilities

2.2.3 Organization Structure

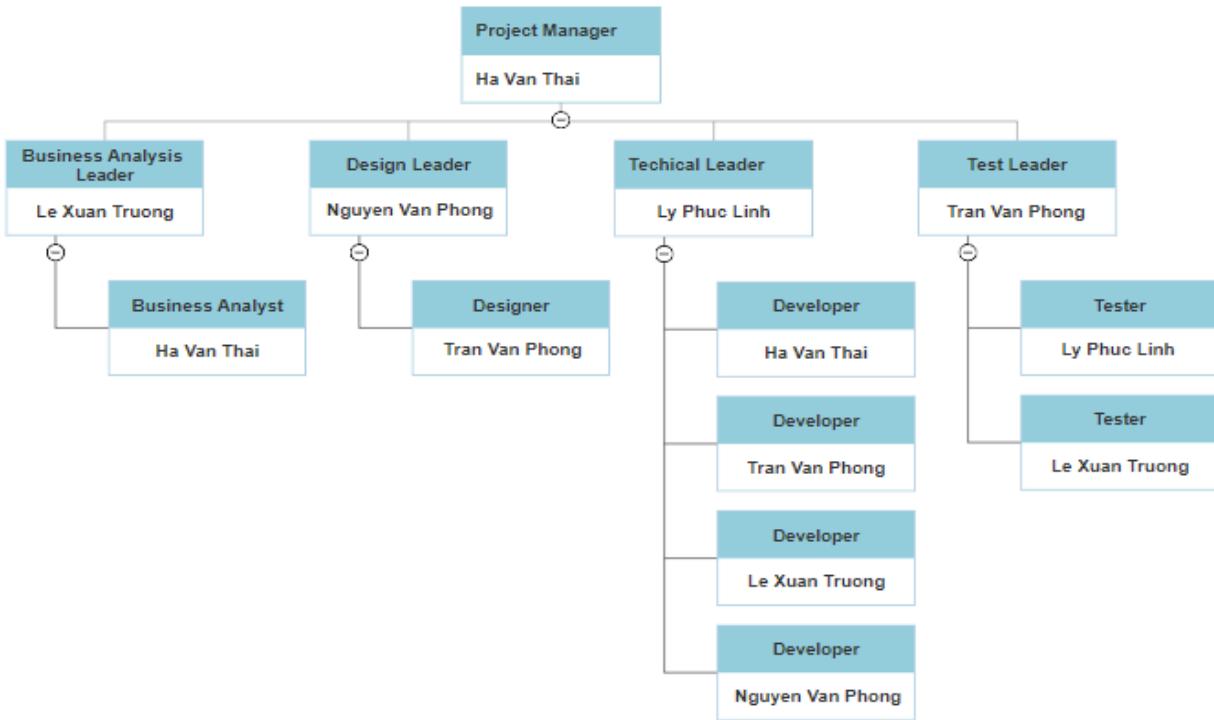


Figure 2-2: Organization Structure

2.2.4 Project Team Member

Team Member	Role
Ha Van Thai	Project manager, business analyst, developer
Nguyen Van Phong	Technical leader, designer
Tran Van Phong	Design leader, developer, tester
Le Xuan Truong	Business analysis leader, developer, tester
Ly Phuc Linh	Test leader, developer

Table 2-2: Project Team Member

2.2.5 Tools and Techniques

Programming Languages:	C#, JavaScript
Frameworks:	Angular 7, Angular material
Software Architecture:	Microservices
Version control:	Git
IDEs / Editors:	Visual studio, Visual studio code
UML Tools:	Astah UML
Web Server:	Nginx
DBMS:	MongoDB
Deployment Server:	Google cloud
Project Management Tools:	Microsoft Project 2013
Process Model:	Iterative and Incremental Software Process Model

Table 2-3: Tools and Techniques

2.3 Project Management Plan

2.3.1 Tasks

	Task Mode	Task Name	Duration	Start	Finish	Predecessors
0	▶	▲ TripSharing	88 days?	Mon 5/13/19	Wed 8/28/19	
1	▶	▷ Initiating	5 days?	Mon 5/13/19	Fri 5/17/19	
11	➤	▲ Executing	76 days	Sat 5/18/19	Mon 8/19/19	
12	▶	▷ Phase 1	54 days?	Sat 5/18/19	Fri 7/19/19	
13	➤	▷ Planning	11 days	Sat 5/18/19	Tue 5/28/19	1
36	▶	▷ Requirements	4 days	Mon 5/27/19	Thu 5/30/19	13
44	➤	▷ Analysis and Design	7 days	Sat 6/1/19	Sat 6/8/19	36
54	➤	▷ Implementation	31 days	Mon 6/10/19	Wed 7/17/19	44
61	▶	▷ Testing	30 days	Wed 6/12/19	Thu 7/18/19	44
66	➤	▷ Evaluation	1 day?	Fri 7/19/19	Fri 7/19/19	
71	▶	▷ Phase 2	21 days?	Mon 7/22/19	Mon 8/19/19	12
72	▶	▷ Planning	1.5 days?	Mon 7/22/19	Tue 7/23/19	
92	▶	▷ Requirements	2 days?	Tue 7/23/19	Wed 7/24/19	72
100	▶	▷ Analysis and Design	1 day?	Thu 7/25/19	Thu 7/25/19	92
110	▶	▷ Implementation	14 days	Fri 7/26/19	Wed 8/14/19	100
117	▶	▷ Testing	16 days	Fri 7/26/19	Fri 8/16/19	100
122	➤	▷ Evaluation	2 days?	Sat 8/17/19	Mon 8/19/19	
127	▶	▷ Closing	7 days	Tue 8/20/19	Wed 8/28/19	
128	▶	▷ Documentation	7 days	Tue 8/20/19	Wed 8/28/19	
135	▶	▷ Deployment	5 days	Tue 8/20/19	Mon 8/26/19	

Figure 2-2: Work Breakdown Structure

2.3.2 Meeting Minutes

All meeting minutes will be written following this template:

Meeting/Project Name:	TripSharing		
Date of Meeting:	06/06/2019	Time: (Type)	3 hours (face-to-face)
Meeting Called by:	ThaiHV	Location:	HB206L
Note Taker:	PhongTV	Time Keeper:	PhongNV
1. Meeting Objective			
-			
2. Attendance			
Name	Role	Email	Phone

ThaiHV	Project Manager Developer Business Analyst	thaihvse048010@fpt.edu.vn	0349940158
PhongNV	Technical Leader Designer	phongnvse05051@fpt.edu.vn	0978296187
PhongTV	Design Leader Developer Tester	phongtvse05048@fpt.edu.vn	0987489397
TruongLX	Business Analysis Leader Developer Tester	truonglxse04616@fpt.edu.vn	0964038801
LinhLP	Test Leader Developer	linhlpse04693@fpt.edu.vn	0362442818
3. Content			
-			
4. Note			
-			

Table 2-4: Meeting Minutes

2.3.3 Code Conventions

We strictly follow Google JavaScript Style Guide and Microsoft C# Programming Guide as the coding convention for the project. Specific convention rules can be found on these pages.

JavaScript: <https://google.github.io/styleguide/jsguide.html>

C#: <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions>

Google JavaScript Style Guide

Table of Contents

1 Introduction	5.9 <code>this</code>
1.1 Terminology notes	5.10 Disallowed features
1.2 Guide notes	
2 Source file basics	6 Naming
2.1 File name	6.1 Rules common to all identifiers
2.2 File encoding: UTF-8	6.2 Rules by identifier type
2.3 Special characters	6.3 Camel case defined
3 Source file structure	7 JSDoc
3.1 License or copyright information, if present	7.1 General form
3.2 @fileoverview JSDoc, if present	7.2 Markdown
3.3 <code>goog.module</code> statement	7.3 JSDoc tags
3.4 <code>goog.require</code> statements	7.4 Line wrapping
3.5 The file's implementation	7.5 Top/file-level comments
4 Formatting	7.6 Class comments
4.1 Braces	7.7 Enum and typedef comments
4.2 Block indentation: +2 spaces	7.8 Method and function comments
4.3 Statements	7.9 Property comments
4.4 Column limit: 80	7.10 Type annotations
4.5 Line-wrapping	7.11 Visibility annotations
4.6 Whitespace	
4.7 Grouping parentheses: recommended	8 Policies
4.8 Comments	8.1 Issues unspecified by Google Style: Be Consistent!
5 Language features	8.2 Compiler warnings
5.1 Local variable declarations	8.3 Deprecation
5.2 Array literals	8.4 Code not in Google Style
5.3 Object literals	8.5 Local style rules
5.4 Classes	8.6 Generated code mostly exempt
5.5 Functions	
5.6 String literals	9 Appendices
5.7 Number literals	9.1 JSDoc tag reference
5.8 Control structures	9.2 Commonly misunderstood style rules
	9.3 Style-related tools
	9.4 Exceptions for legacy platforms

Figure 2-3: Google JavaScript Style Guide

2.3.4 Risk Management Plan

No	Name	Prevention	Correction	Status
R1	Miscommunication	<p>Any problem raised has to be documented or filed.</p> <p>Team members should express their opinion clearly.</p>	Miscommunication must be resolved early.	Closed
R2	New technology	<p>Choosing technology carefully and based on team member's qualification. All member must develop self-learning skill.</p>	The technology choice should be consider carefully, well explained and training for member if necessary.	Closed

R3	Member idea conflict	Member is free to express their idea and discuss with others to find the most suitable solution.	Analyze any idea carefully and base on reality and possibility, solve conflict as soon as possible.	Closed
R4	Business problem	Understand business of project, consider if its fit applicability, reality and technology possibility. Ask supervisor if needed.	Project business must be tight and realistic.	Closed
R5	Illness or absence of team members	Member must notice to the team about absence period and the plan of how to keep up with the work process.	Ensure that the absence of a member won't affect others and always have plans to deal with this problem.	Closed
R6	Resources were lost or deleted	Team must have several online and offline back up copy of project.	At least 3 back up way such as Github, Google Drive, etc. and offline copy on each member computer.	Closed
R7	Illness or absence of team members	Member must notice to the team about absence period and the plan of how to keep up with the work process.	Ensure that the absence of a member won't affect others and always have plans to deal with this problem.	Closed
R8	Changes of software architecture	Read and understand the software architecture thoroughly before starting to apply it to the project	Changes of software architecture must be done early.	Closed

Table 2-5: Risk Management Plan

2.3.5 Communication Plan

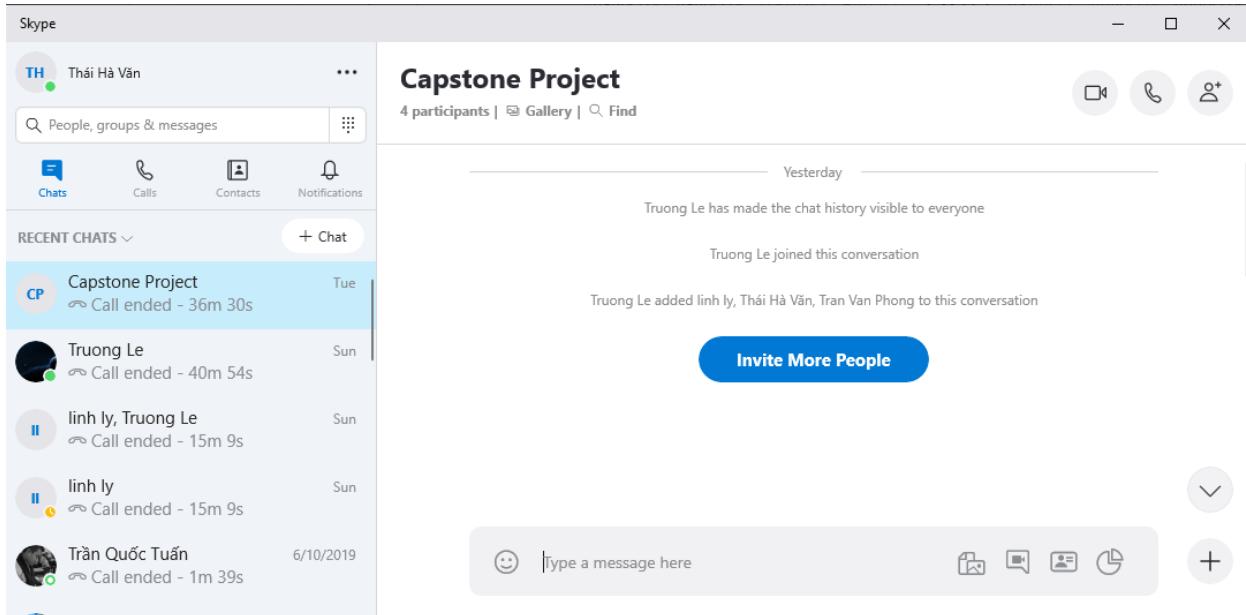
Weekly meeting schedule: We follow the Iterative and Incremental Process Model, and we separate the project into two main team, back-end API team and front-end web app team. Each team will do the tasks assigned to team members by the Team Leader and depending on difficulty the Technical Leader will assign deadlines for each task. We have a meeting every Thursday to update all team members about what has been done during last week.

Daily meeting schedule: Each sub team has one development team with different schedule and deadlines. Before the daily meetings, each member will be telling:

- What he has completed.
- What he is working on.
- When he will finish.
- What issues he is encountering, or if he needs assistance.
- What he will be doing after finishing the task.

Unscheduled meeting: Assuming someone has encountered a serious problem that he wants to solve immediately, we will have a meeting via some online channel: Skype, or Phone. Face to face meeting in any emergency cases.

Communication channel: Our main communication channel is Skype and Facebook group. We use TeamViewer for assisting team member when he meets technical issues.



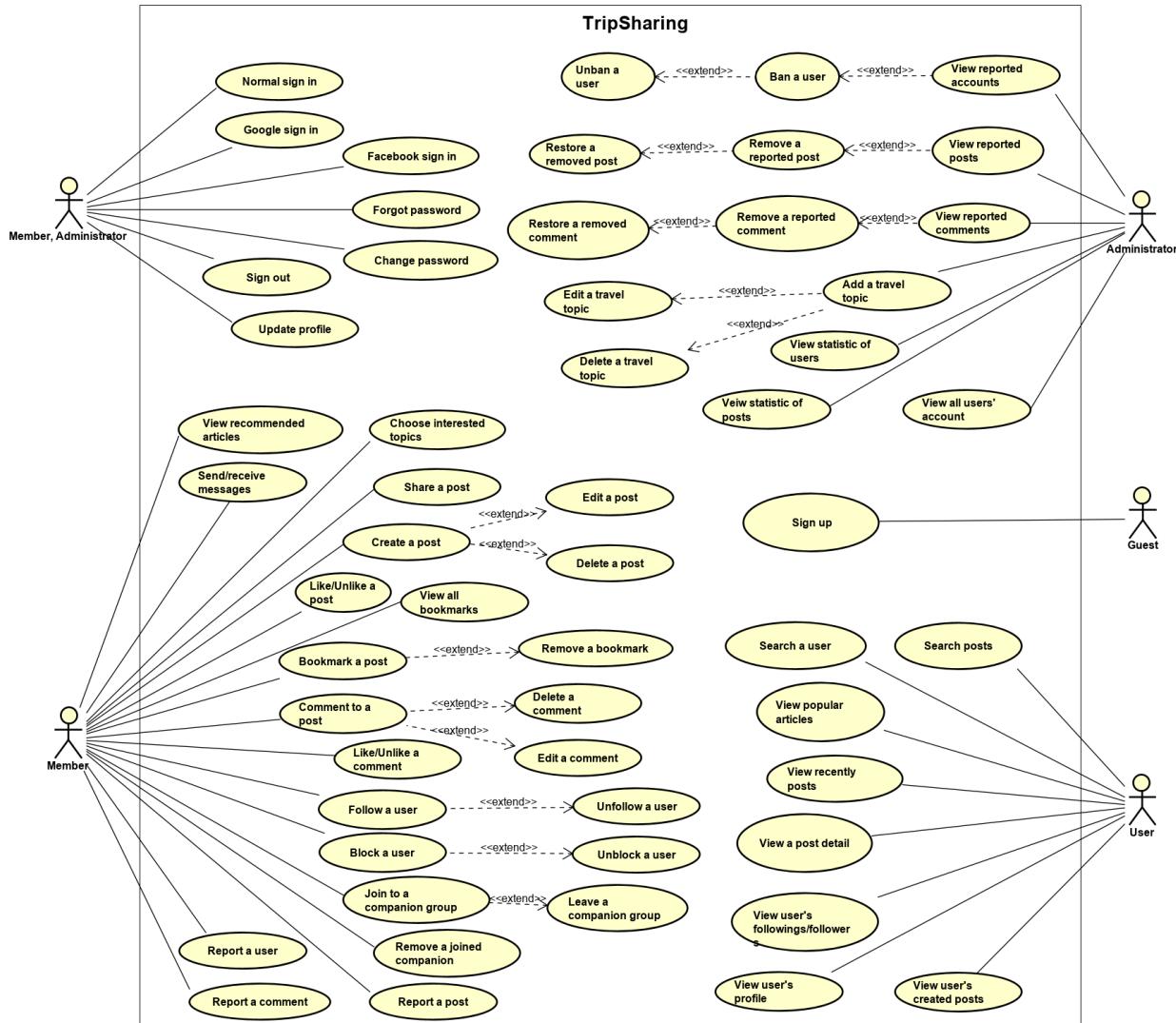
Chapter 3: Software Requirement Specification

3.1 Purpose

This chapter contains details about functional and non-functional requirements the website. Constraints and detailed requirements are specifically described for developers to follow when completing the tasks.

3.2 Functional Requirements

3.2.1 Use Case Diagram



3.2.2 Business Rules

No	Description
B1	The email must not be empty and valid.

B2	The confirm password must be matched with the new password.
B3	Username must not be empty and not be longer than 100 characters
B4	Image file's mime type must be: "image/png" or "image/jpeg".
B5	Maximum file size allowed is 10Mb
B6	Eve
B7	The root Administrator has the highest privilege.
B8	An Administrator can: ban/unban users, remove/restore posts, remove/restore comments.
B9	A banned user can't post articles, comment to a post or chat with other users.
B10	When user sign out, his/her current access token is blacklisted.
B11	An article title must not be empty and not be longer than 1000 characters.
B12	An article content must not be empty and not be longer than 100,000 characters.
B13	An article must be tagged to at least one travel place.
B14	An article must be tagged to at least one travel topic.
B15	A comment must not be empty and not be longer than 10,000 characters.
B16	Each destination in virtual trip has a note and not be longer than 500 characters.
B17	A chat message must not be empty and not be longer than 10,000 characters.
B18	Member can report any other member, post or comment
B19	When a finding companion post is expired, users can't join to that companion group.
B20	Contribution points formula: <ul style="list-style-type: none"> - Create a post: +5 points - For each 100 likes on a post: +1 points

3.2.3 Use Cases

Actor	Description
Guest	Anyone who visits the website and doesn't have an account yet.
Member	Those who has registered an account on TripSharing.
Administrator	Who has the highest the permission level and has responsible for managing the entire website.

User	Any normal user of following types: Guest, Member, and Administrator
------	--

ID	Actor	Name
UC-001	Guest	Sign up
UC-002	Member, Administrator	Google sign in
UC-003	Member, Administrator	Facebook sign in
UC-004	Member, Administrator	Normal sign in
UC-005	Member, Administrator	Forgot password
UC-006	Member, Administrator	Sign out
UC-007	Member, Administrator	Change password
UC-008	Member, Administrator	Update profile
UC-009	Member	Choose interested topic
UC-010	Member	Create a post
UC-011	Member	Edit a post
UC-012	Member	Delete a post
UC-013	Member	Like/Unlike a post
UC-014	Member	Share a post
UC-015	Member	Report a post
UC-016	Member	Bookmark a post
UC-017	Member	View all bookmarks
UC-018	Member	Remove a bookmark
UC-019	Member	Comment to a post
UC-020	Member	Edit a comment
UC-021	Member	Delete a comment
UC-022	Member	Like/Unlike a comment
UC-023	Member	Report a comment
UC-024	Member	Follow a user
UC-025	Member	Unfollow a user
UC-028	Member	Report a user
UC-029	Member	Join to a companion group

UC-030	Member	Leave a companion group
UC-031	Member	Remove a joined companion
UC-032	Member	Send/receive messages
UC-033	Member	View recommended articles
UC-034	Administrator	View all users' account
UC-035	Administrator	View reported accounts
UC-036	Administrator	Ban a user
UC-037	Administrator	Unban a user
UC-038	Administrator	View reported posts
UC-039	Administrator	Remove a reported post
UC-040	Administrator	Restore a removed post
UC-041	Administrator	View reported comments
UC-042	Administrator	Remove a reported comment
UC-043	Administrator	Restore a removed comment
UC-044	Administrator	Add a travel topic
UC-045	Administrator	Edit a travel topic
UC-046	Administrator	Delete a travel topic
UC-047	Administrator	View statistic of users
UC-048	Administrator	View statistic of posts
UC-049	User	Search post by travel destinations
UC-050	User	Search a user
UC-051	User	View popular articles
UC-052	User	View recently posts
UC-053	User	View a post detail
UC-054	User	View user's profile
UC-055	User	View user's followings/followers
UC-056	User	View user's created posts

3.2.3.1 Guest

3.2.3.1.1 Sign up

Use Case Specification

UC ID and Name:	UC-001 Sign Up		
Created by:	PhongTV	Date created:	01/06/2019
Primary Actor:	Guest	Secondary Actor:	
Trigger:	N/A		
Description:	Allow user to create an account.		
Preconditions:	Guest has not signed in the website.		
Post conditions:	New account is added to database.		
Normal Flow:	<ol style="list-style-type: none"> 1. User clicks “Đăng nhập” button on the header. 2. System display a form for sign in function 3. User clicks “Đăng ký” button at the bottom of the form. 4. Browser redirects to the sign up page. 5. User enters email, password and re-enter password and must agree with policy. 6. Click “Đăng kí” button, system send verify email, save user information. 7. System directs to home page. 		
Alternative Flows:	N/A		
Exceptions:	N/A		
Priority:	High		
Frequency of Use:	Medium		
Business Rules:	B1, B2		

3.2.3.2 Member, Administrator

3.2.3.2.1 Google sign in

Use Case Specification

UC ID and Name:	UC-002 Google sign in		
Created by:	PhongTV	Date created:	05/06/2019
Primary Actor:	Member	Secondary Actor:	Administrator
Trigger:	N/A		
Description:	Allow user sign in to website with a Google account		
Preconditions:	1. User has registered an account 2. User has not signed in the website		
Post conditions:	User has signed in successfully		
Normal Flow:	1. User clicks “Đăng nhập” button on the header 2. System display Login form 3. User clicks “G” button on the login form. 4. Browser displays another window that contains Google sign in form. 5. User selects a Google account or enters email and password 6. Google verifies login and closes the popup. 7. System directs to home page.		
Alternative Flows:	N/A		
Exceptions:	1. Cannot communicate with Google API 2. Google API return error		
Priority:	High		
Frequency of Use:	Medium		
Business Rules:	N/A		

3.2.3.2.2 Facebook sign in

Use Case Specification

UC ID and Name:	UC-003 Facebook sign in		
Created by:	PhongTV	Date created:	02/08/2019

Primary Actor:	Member	Secondary Actor:	Administrator
Trigger:	N/A		
Description:	Allow user sign in to website with a Facebook account		
Preconditions:	1. User has registered an account 2. User has not signed in the website		
Post conditions:	User has signed in successfully		
Normal Flow:	1. User clicks “Đăng nhập” button on the header. 2. System display Login form 3. User clicks “F” button on the login form. 4. Browser displays another window that contains Facebook sign in form. 5. User enters his/her Facebook username and password 6. Facebook verifies login and closes the popup. 7. System directs to home page.		
Alternative Flows:	N/A		
Exceptions:	1. Cannot communicate with Facebook API 2. Facebook API return error		
Priority:	High		
Frequency of Use:	Medium		
Business Rules:	N/A		

3.2.3.2.3 Normal sign in

Use Case Specification

UC ID and Name:	UC-004 Normal sign in		
Created by:	PhongTV	Date created:	01/06/2019
Primary Actor:	Member	Secondary Actor:	Administrator
Trigger:	N/A		

Description:	Allow user sign in to website with a registered account
Preconditions:	1. User has registered an account 2. User has not signed in the website
Post conditions:	User has signed in successfully
Normal Flow:	1. User clicks “Đăng nhập” button on the header 2. System display login form 3. User enters his/her Email and Password 4. User click “Đăng nhập” button on login form 5. System verifies username, password and close the popup. 6. System directs to home page.
Alternative Flows:	N/A
Exceptions:	N/A
Priority:	High
Frequency of Use:	Medium
Business Rules:	B1, B3

3.2.3.2.4 Forgot password

Use Case Specification

UC ID and Name:	UC-005 Forgot Password		
Created by:	PhongTV	Date created:	02/06/2019
Primary Actor:	Member	Secondary Actor:	Administrator
Trigger:	N/A		
Description:	Allow user reset password if his/her forgot the password		

Preconditions:	User has signed up with an account successfully
Post conditions:	New password has updated in database
Normal Flow:	<ol style="list-style-type: none"> 1. User clicks “Đăng nhập” button on the header 2. System display Login Form 3. User clicks “Quên mật khẩu” link on login form 4. Browser redirects to reset password page 5. User enter email registered account 6. Click “Lấy lại mật khẩu” button on the form 7. Display success message and redirect to home page
Alternative Flows:	N/A
Exceptions:	User enters an email that doesn't exist
Priority:	High
Frequency of Use:	Medium
Business Rules:	N/A

3.2.3.2.5 Sign out

Use Case Specification

UC ID and Name:	UC-006 Sign out		
Created by:	PhongTV	Date created:	02/06/2019
Primary Actor:	Member	Secondary Actor:	Administrator
Trigger:	N/A		
Description:	Allow user to sign out from the website		
Preconditions:	User has signed in with an account successfully		

Post conditions:	User has signed out
Normal Flow:	<ol style="list-style-type: none"> 1. User clicks the user avatar on the header 2. System display a list of options 3. User clicks option “Đăng xuất” 4. User has signed out from the website
Alternative Flows:	N/A
Exceptions:	N/A
Priority:	High
Frequency of Use:	Medium
Business Rules:	B10

3.2.3.2.6 Change password

Use Case Specification

UC ID and Name:	UC-007 Change password		
Created by:	PhongTV	Date created:	02/06/2019
Primary Actor:	Member	Secondary Actor:	Administrator
Trigger:	N/A		
Description:	Allow user to change his/her password		
Preconditions:	User has signed in with an account successfully		
Post conditions:	New password has updated on the database		
Normal Flow:	<ol style="list-style-type: none"> 1. User clicks user avatar on the header. 2. System display a list option 3. User clicks option “Thay đổi mật khẩu” on the list 4. Browser redirects to the change-password page. 5. User enter old password, new password and reenter password. 		

	<p>6. After complete, user click “Lưu” button</p> <p>7. System will display message confirm and change new password</p>
Alternative Flows:	N/A
Exceptions:	User sign up with Google account or Facebook account don't have password to change.
Priority:	Medium
Frequency of Use:	Medium
Business Rules:	N/A

3.2.3.2.7 Update profile

Use Case Specification

UC ID and Name:	UC-008 Update profile		
Created by:	PhongTV	Date created:	05/06/2019
Primary Actor:	Member	Secondary Actor:	Administrator
Trigger:	N/A		
Description:	Allow user update his/her profile		
Preconditions:	User has signed in successfully		
Post conditions:	New information has updated on database		
Normal Flow:	<ol style="list-style-type: none"> 1. User clicks user avatar on the header 2. System display a list option 3. User clicks option “Trang cá nhân” 4. Browser redirects to his/her personal page 5. Click “Chỉnh sửa trang cá nhân” 6. User update information on the displayed popup 7. Click “Cập nhật” to finish update 		

Alternative Flows:	N/A
Exceptions:	N/A
Priority:	Medium
Frequency of Use:	Medium
Business Rules:	N/A

3.2.3.3 Member

3.2.3.3.1 Choose interesting topic

Use Case Specification

UC ID and Name:	UC-009 Choose interesting topic		
Created by:	PhongTV	Date created:	02/06/2019
Primary Actor:	Member	Secondary Actor:	
Trigger:	N/A		
Description:	Allow user choose travel topics he/she interests in.		
Preconditions:	User has signed in to the website		
Post conditions:	His/her interested topic information has updated on database		
Normal Flow:	<ol style="list-style-type: none"> 1. User sign-in for the first time 2. System display a page to complete user information 3. User fill all the required field and click “Tiếp tục” button 4. System display a list interesting topic 5. User choose three or more topics 6. User clicks “Tiếp tục” button 7. System redirects to home page. 		
Alternative Flows:	<ol style="list-style-type: none"> 1. On his/her personal page, user click “Chỉnh sửa thông tin cá nhân” 2. On the displayed popup, click “Tiếp tục” button 		

	<p>3. User selects/unselects travel topics</p> <p>4. User clicks “Cập nhật” button</p>
Exceptions:	N/A
Priority:	Medium
Frequency of Use:	Medium
Business Rules:	N/A

3.2.3.3.2 Create a post

Use Case Specification

UC ID and Name:	UC-010 Create a post		
Created by:	PhongTV	Date created:	04/06/2019
Primary Actor:	Member	Secondary Actor:	
Trigger:	N/A		
Description:	Allow user create a post to share his/her travel experience		
Preconditions:	User has signed in to the website		
Post conditions:	<p>1. Post information has created on database</p> <p>2. System displays a popup to notify that a post has created successfully.</p>		
Normal Flow:	<p>1. User clicks “+” button on the header.</p> <p>2. System display a list of post's types.</p> <p>3. User choose the post's type he/she wants to create (article, virtual-trip, finding-companion post).</p> <p>4. System redirect to the create-post page.</p> <p>5. User fills all needed information then click “Tạo bài viết” button.</p> <p>6. System displays a popup notify that a post is created successfully.</p>		

Alternative Flows:	N/A
Exceptions:	System can't communicate with Google Map API in creating a virtual-trip function.
Priority:	Medium
Frequency of Use:	Medium
Business Rules:	B4, B5, B6, B11, B12, B13, B14

3.2.3.3.3 Edit a post

Use Case Specification

UC ID and Name:	UC-011 Edit an article		
Created by:	PhongTV	Date created:	04/06/2019
Primary Actor:	Member	Secondary Actor:	
Trigger:	N/A		
Description:	Allow user to edit his/her created post.		
Preconditions:	1. User has signed in to the website 2. User has created a post		
Post conditions:	Post's information has updated on database		
Normal Flow:	1. From post detail page, user click “...” icon. 2. System displays a list of options 3. User click “Chỉnh sửa” option 4. User edit the information he/she want to change 5. User click “Lưu thay đổi” button 6. System displays a popup that notify a post is updated successful.		
Alternative Flows:	N/A		

Exceptions:	System can't communicate with Google Map API in editing a virtual-trip function.
Priority:	Medium
Frequency of Use:	Medium
Business Rules:	B11, B12, B13, B14

3.2.3.3.4 Delete a post

Use Case Specification

UC ID and Name:	UC-012 Delete a post		
Created by:	PhongTV	Date created:	05/06/2019
Primary Actor:	Member	Secondary Actor:	
Trigger:	N/A		
Description:	Allow user to delete a post		
Preconditions:	1. User has signed in to the website 2. User has created a post		
Post conditions:	Post's is_active property has changed to "false" on database		
Normal Flow:	1. From post detail page, user click "..." icon. 2. System displays a list of options 3. User click "Xóa" option. 4. System display popup confirm that if user really want to delete this post. 5. System display pop-up to notify that a post has been deleted.		
Alternative Flows:	N/A		
Exceptions:	N/A		
Priority:	Medium		
Frequency of Use:	Medium		

Business Rules:	N/A
-----------------	-----

3.2.3.3.5 Like/Unlike a post

Use Case Specification

UC ID and Name:	UC-013 Like/Unlike a post		
Created by:	PhongTV	Date created:	07/06/2019
Primary Actor:	Member	Secondary Actor:	
Trigger:	N/A		
Description:	Allow user to like/unlike a post		
Preconditions:	User has signed in to the website		
Post conditions:	1. Like information has added on database. 2. Number of post's like is increase/decrease by one.		
Normal Flow:	1. From post detail page, user click "like" button (thumb-up button)		
Alternative Flows:	N/A		
Exceptions:	N/A		
Priority:	Medium		
Frequency of Use:	Medium		
Business Rules:	N/A		

3.2.3.3.6 Share a post<PENDING>

Use Case Specification

UC ID and Name:	UC-014 Share a post		
Created by:	PhongTV	Date created:	02/08/2019
Primary Actor:	Member	Secondary Actor:	

Trigger:	N/A
Description:	Allow user to share a post on facebook
Preconditions:	User has signed in to the website
Post conditions:	N/A
Normal Flow:	< PENDING >
Alternative Flows:	N/A
Exceptions:	<ol style="list-style-type: none"> 1. Can't communicate with Facebook API 2. Facebook responds error
Priority:	Medium
Frequency of Use:	Medium
Business Rules:	N/A

3.2.3.3.7 Report a post

Use Case Specification

UC ID and Name:	UC-015 Report a post		
Created by:	PhongTV	Date created:	20/07/2019
Primary Actor:	Member	Secondary Actor:	
Trigger:	User sees a post is violated and want to report that post.		
Description:	Allow user to report a post		
Preconditions:	User has signed in with an account successfully		
Post conditions:	A new report record has added on database.		

Normal Flow:	<ol style="list-style-type: none"> 1. From post detail page, user click “...” icon. 2. System displays a list of options 3. User clicks “Báo cáo vi phạm” option 4. System displays a list of violation reasons. 5. User chooses one or more reason. 6. User click “Gửi” button. 7. System displays a message to notify that user has sent report successfully.
Alternative Flows:	N/A
Exceptions:	N/A
Priority:	High
Frequency of Use:	Medium
Business Rules:	N/A

3.2.3.3.8 Bookmark a post

Use Case Specification

UC ID and Name:	UC-016 Bookmark a post		
Created by:	PhongTV	Date created:	7/06/2019
Primary Actor:	Member	Secondary Actor:	
Trigger:	User sees a post that he/she want to save to read later		
Description:	Allow user to bookmark a post		
Preconditions:	User has signed in with an account successfully		
Post conditions:	<ol style="list-style-type: none"> 1. A new bookmark record has added on database. 2. Bookmark icon has change color 		
Normal Flow:	<ol style="list-style-type: none"> 1. From a page that display a list of post. 2. User click bookmark icon in the post cover image. 		

Alternative Flows:	<ol style="list-style-type: none"> 1. From a page that display a list of post. 2. User click on title and go to post detail page. 3. Click bookmark icon in left of the post.
Exceptions:	N/A
Priority:	Medium
Frequency of Use:	Medium
Business Rules:	N/A

3.2.3.3.9 View all bookmark

Use Case Specification

UC ID and Name:	UC-017 View all bookmarks		
Created by:	PhongTV	Date created:	7/06/2019
Primary Actor:	Member	Secondary Actor:	
Trigger:	N/A		
Description:	Allow user to view all bookmarked posts.		
Preconditions:	<ol style="list-style-type: none"> 1. User has signed in with an account successfully 2. User has bookmarked at least a post 		
Post conditions:	A list of bookmarked posts is displayed.		
Normal Flow:	<ol style="list-style-type: none"> 1. From personal page, user click “Bài viết đã lưu” button 2. Browser redirects to the list-bookmarks page. 		
Alternative Flows:	N/A		
Exceptions:	N/A		
Priority:	Medium		
Frequency of Use:	Medium		
Business Rules:	N/A		

3.2.3.3.10 Remove a bookmark

Use Case Specification

UC ID and Name:	UC-018 Remove a bookmark		
Created by:	PhongTV	Date created:	7/06/2019
Primary Actor:	Member	Secondary Actor:	
Trigger:	N/A		
Description:	Allow user to remove a bookmarked post from his/her list.		
Preconditions:	1. User has signed in with an account successfully 2. User has at least one bookmarked post.		
Post conditions:	A bookmark record has been deleted on database		
Normal Flow:	1. From personal page, user click “Bài viết đã lưu” button 2. Browser redirects to the list-bookmarks page. 3. User click “x” icon on post that he/she wants to remove. 4. A post is removed from list.		
Alternative Flows:	If a bookmarked post is displayed on list of post page: 1. From a page that display a list of post. 2. User click bookmark icon in the post cover image. If user is viewing bookmarked post on detail page: 1. Click bookmark icon on the left of the post.		
Exceptions:	N/A		
Priority:	Medium		
Frequency of Use:	Medium		
Business Rules:	N/A		

3.2.3.3.11 Comment to a post

Use Case Specification

UC ID and Name:	UC-019 Comment to a post		
Created by:	PhongNV	Date created:	07/06/2019
Primary Actor:	Member	Secondary Actors:	
Trigger:	N/A		
Description:	Allow user to comment on a post		
Preconditions:	User has signed in to the website		
Post conditions:	1. A new comment record has added on database 2. A new comment is displayed on comment section of post detail page.		
Normal Flow:	1. At post detail page, user scrolls to the end of page 2. User write a comment. 3. User presses click “comment” button. 4. System displays a new comment at the comment section.		
Alternative Flow:	N/A		
Exceptions:	N/A		
Priority:	High		
Frequency of Use:	High		
Business Rules:	B16		

3.2.3.3.12 Edit a comment

Use Case Specification

UC ID and Name:	UC-020 Edit a comment		
Created by:	PhongNV	Date created:	8/06/2019
Primary Actor:	Member	Secondary Actors:	
Trigger:	N/A		
Description:	Allow user to edit a comment which he/she has created.		

Preconditions:	<ol style="list-style-type: none"> 1. User is signed in to the website 2. User has commented to a post.
Post conditions:	<ol style="list-style-type: none"> 1. Comment information has updated on database 2. Comment data has changed on front-end
Normal Flow:	<ol style="list-style-type: none"> 1. At post detail page, user scrolls to the end of page 2. At the comment section, user clicks “...” on the comment he/she want to edit. 3. A popup is displayed with a list of options. 4. User clicks on “Chỉnh sửa” option in popup. 5. User edits comment. 6. User presses key enter to finish.
Alternative Flow:	N/A
Exceptions:	N/A
Priority:	Low
Frequency of Use:	Low
Business Rules:	B16

3.2.3.3.13 Delete a comment

Use Case Specification

UC ID and Name:	UC-021 Delete a comment		
Created by:	PhongNV	Date created:	08/06/2019
Primary Actor:	Member	Secondary Actors:	
Trigger:	N/A		
Description:	Allow user to delete a comment which he/she has created.		
Preconditions:	<ol style="list-style-type: none"> 1. User is signed in to the website 2. User has commented to a post. 		
Post conditions:	<ol style="list-style-type: none"> 1. Comment is_active property has been changed to false on database. 2. That comment has been removed on front-end 		

Normal Flow:	<ol style="list-style-type: none"> 1. At post detail page, user scrolls to the end of page 2. At the comment section, user clicks “...” on the comment he/she want to edit. 3. A popup is displayed with a list of options. 4. User clicks on “Xóa” option in popup. 5. System display a popup to confirm if user is really want to delete this comment. 6. User clicks on “Tiếp tục” button to delete this comment.
Alternative Flow:	N/A
Exceptions:	N/A
Priority:	Low
Frequency of Use:	Low
Business Rules:	N/A

3.2.3.3.14 Like/Unlike a comment

Use Case Specification

UC ID and Name:	UC-022 Like/Unlike a comment		
Created by:	PhongNV	Date created:	08/06/2019
Primary Actor:	Member	Secondary Actors:	
Trigger:	N/A		
Description:	Allow user to like/unlike a comment		
Preconditions:	User has signed in to the website.		
Post conditions:	<ol style="list-style-type: none"> 1. A “like” has been added/removed on database 2. Number of likes on the comment is increased/decreased. 		
Normal Flow:	<ol style="list-style-type: none"> 1. From a post detail page, user scrolls to the comment section. 2. User click on thumb-up button to like/unlike a comment 		
Alternative Flow:	N/A		
Exceptions:	N/A		
Priority:	Low		

Frequency of Use:	Low
Business Rules:	N/A

3.2.3.3.15 Report a comment

Use Case Specification

UC ID and Name:	UC-023 Report a comment		
Created by:	PhongNV	Date created:	08/06/2019
Primary Actor:	Member	Secondary Actors:	
Trigger:	N/A		
Description:	User wants to report a comments		
Preconditions:	1. User is signed in to the website 2. User has commented to a post.		
Post conditions:	A new report record has been added on database.		
Normal Flow:	1. From post detail page, user scrolls to the comment section. 2. User clicks on “...” icon at the comment he/she wants to report 3. User clicks on “Báo cáo vi phạm” option in popup. 4. System displays a list of violation reason for user to choose. 10. User selects one or more reasons to report or write another reason. 11. User clicks to button “Gửi” to send report.		
Alternative Flow:	N/A		
Exceptions:	N/A		
Priority:	Low		
Frequency of Use:	Low		
Business Rules:	N/A		

3.2.3.3.16 Follow a user

Use Case Specification

UC ID and Name:	UC-024 Follow a user
-----------------	-----------------------------

Created by:	PhongNV	Date created:	15/06/2019
Primary Actor:	Member	Secondary Actors:	
Trigger:	N/A		
Description:	Allow user to follow other users		
Preconditions:	User has signed in to the website		
Post conditions:	<ol style="list-style-type: none"> 1. Follow record has been added on database 2. The “Theo dõi” button is disappeared or changed to “Đã theo dõi”. 		
Normal Flow:	<ol style="list-style-type: none"> 1. User go to the user’s personal page that he/she wants to follow. 2. Click on “Theo dõi” button at the right of user’s name 3. The “Theo dõi” button is changed to “Đã theo dõi”. 		
Alternative Flow:	<p>At list-posts page:</p> <ol style="list-style-type: none"> 1. User click on “Theo dõi” button on the post item. 2. The “Theo dõi” button is disappeared. <p>At list-users page:</p> <ol style="list-style-type: none"> 1. User click on “Theo dõi” button on the user item. 2. The “Theo dõi” button is disappeared. 		
Exceptions:	N/A		
Priority:	Low		
Frequency of Use:	Low		
Business Rules:			

3.2.3.3.17 Unfollow a user

Use Case Specification

UC ID and Name:	UC-025 Unfollow a user		
Created by:	PhongNV	Date created:	15/06/2019
Primary Actor:	Member	Secondary Actors:	
Trigger:	N/A		
Description:	Allow user to unfollow a user that he/she has already followed.		
Preconditions:	User has signed in to the website		

Post conditions:	<ol style="list-style-type: none"> 1. Follow record has been delete on database 2. The “Theo dõi” button is appeared or changed from “Đã theo dõi”.
Normal Flow:	<ol style="list-style-type: none"> 1. User go to the user’s personal page that he/she wants to unfollow. 2. Click on “Đã theo dõi” button at the right of user’s name 3. The “Đã theo dõi” button is changed to “Theo dõi”.
Alternative Flow:	<ol style="list-style-type: none"> 1. User go to his/her personal page. 2. At the left side of page, click on “n – đang theo dõi” / “n – theo dõi” text. 3. A popup is displayed with list of users he/she has already followed 4. User click on ‘Hủy theo dõi’ button 5. The user is removed from the list.
Exceptions:	N/A
Priority:	Low
Frequency of Use:	Low
Business Rules:	N/A

3.2.3.3.18 Report a user

Use Case Specification

UC ID and Name:	UC-028 Report a user		
Created by:	PhongNV	Date created:	18/06/2019
Primary Actor:	Member	Secondary Actors:	
Trigger:	N/A		
Description:	Allow users to report a violated user.		
Preconditions:	User has signed in to the website		
Post conditions:	Report record has been added on database		
Normal Flow:	<ol style="list-style-type: none"> 1. At list of posts/users page or post detail page, click on “...” button at the right of user’s name. 2. System displays a list of options 3. User click on “Báo cáo vi phạm” options. 		

	<p>4. A popup is displayed with list of violated reasons.</p> <p>5. User choose one or more reason, then click “Gửi” button.</p>
Alternative Flow:	N/A
Exceptions:	N/A
Priority:	Low
Frequency of Use:	Low
Business Rules:	N/A

3.2.3.3.19 Join to a companion group.

Use Case Specification

UC ID and Name:	UC-29 Join to a companion group		
Created by:	PhongNV	Date created:	26/06/2019
Primary Actor:	Member	Secondary Actors:	
Trigger:	N/A		
Description:	Allow user to join a group to discuss and plan for traveling together		
Preconditions:	User has signed in to the website		
Post conditions:	<ol style="list-style-type: none"> 1. User information has been added to the group on database 2. User can access to the group chat 		
Normal Flow:	<ol style="list-style-type: none"> 1. At the finding-companion-post detail page, user clicks on “Tham gia” button. 2. System displays a message notify that his/her request has been sent. 3. After the group’s admin accept his/her request, system will send a notification to the user to notify that he/she has joined to the group. 		
Alternative Flow:	N/A		
Exceptions:	N/A		
Priority:	Low		
Frequency of Use:	Low		

Business Rules:	N/A
-----------------	-----

3.2.3.3.20 Leave a companion group.

Use Case Specification

UC ID and Name:	UC-30 Leave a companion group		
Created by:	PhongNV	Date created:	26/06/2019
Primary Actor:	Member	Secondary Actors:	
Trigger:	N/A		
Description:	Allow user to leave a group that he/she has joined.		
Preconditions:	User has signed in to the website		
Post conditions:	<ol style="list-style-type: none"> 1. User information has been removed from the group on database 2. User can't access to the group chat any more. 		
Normal Flow:	<ol style="list-style-type: none"> 1. At the home page, User click on message icon. 2. A popup is displayed with a list of conversations, then user click on “Xem tất cả” link to redirect to chat pages 3. At the left side of the page, user select on the conversation that related to the group he/she wants to leave. 4. System displays a popup to confirm that if user really want to leave. 5. User click on “Tiếp tục” button. 		
Alternative Flow:	N/A		
Exceptions:	N/A		
Priority:	Low		
Frequency of Use:	Low		
Business Rules:	N/A		

3.2.3.3.21 Remove a joined companion

Use Case Specification

UC ID and Name:	UC-31 Remove a joined companion		
Created by:	PhongNV	Date created:	26/06/2019
Primary Actor:	Member	Secondary Actors:	
Trigger:	N/A		
Description:	Allow group's admin to remove a member in the group.		
Preconditions:	1. User has signed in to the website 2. User has created a companion group.		
Post conditions:	1. User information has been removed from the group on database 2. User item has been removed in the list of group members.		
Normal Flow:	1. At the home page, User click on message icon. 2. A popup is displayed with a list of conversations, then user click on “Xem tất cả” link to redirect to chat pages 3. User selects the group conversation he/she wants to remove a group member. 4. At the right side of page, user click on “Mọi người” button. 5. A list of group members is displayed, then user click on “...” button at the user item that he/she wants to remove 6. A popup with list of options is displayed, user click on “Xóa khỏi nhóm” button. 7. System displays a popup to confirm that if user really wants to remove that user. 8. User click “Tiếp tục” to continue		
Alternative Flow:	N/A		
Exceptions:	N/A		
Priority:	Low		
Frequency of Use:	Low		
Business Rules:	N/A		

3.2.3.3.22 Send/recieve a message.

Use Case Specification

UC ID and Name:	UC-32 Send/recieve a message		
Created by:	PhongNV	Date created:	02/07/2019
Primary Actor:	Member	Secondary Actors:	
Trigger:	N/A		
Description:	Allow user to chat with other users		
Preconditions:	User has signed in to the website		
Post conditions:	1. Messages information has been added on database 2. The user he/she sends message to get a new message notification.		
Normal Flow:	1. At the homepage, user click on the message icon 2. A list of conversation is displayed 3. User click on the conversation he/she wants to chat 4. User enters message then press “Enter” button or click send button		
Alternative Flow:	For the first time user send message to another user: 1. From the user’s personal page that he/she wants to send message, user click on “Gửi tin nhắn” button 2. A popup is displayed, then user enter message to the text area. 3. User click “Gửi” button		
Exceptions:	N/A		
Priority:	High		
Frequency of Use:	High		
Business Rules:	N/A		

3.2.3.3.23 View recommended articles

Use Case Specification

UC ID and Name:	UC-33 View recommended articles		
Created by:	PhongNV	Date created:	08/07/2019
Primary Actor:	Member	Secondary Actors:	
Trigger:	N/A		
Description:	Allow user to see a list of posts which he/she may like.		

Preconditions:	User has signed in to the website
Post conditions:	A list of articles is displayed for users base on their interests.
Normal Flow:	<ol style="list-style-type: none"> 1. At the home page, user can see a small list of recommended post 2. User click “Xem thêm” at the right side of the page to see more recommended post
Alternative Flow:	N/A
Exceptions:	N/A
Priority:	High
Frequency of Use:	Low
Business Rules:	N/A

3.2.3.4 Administrator

3.2.3.4.1 View all users' account

Use Case Specification

UC ID and Name:	UC-034 View all users' account		
Created by:	LinhLP	Date created:	15/07/2019
Primary Actor:	Administrator	Secondary Actor:	
Trigger:	N/A		
Description:	Allow admin to view all users' account of the system		
Preconditions:	Admin has successfully logged in the system		
Post conditions:	List of user account is displayed.		
Normal Flow:	<ol style="list-style-type: none"> 1. At the admin-dashboard page, admin click on “Người dùng” tab at the left side of the page. 2. A list of user account is displayed 		

Alternative Flows:	N/A
Exceptions:	N/A
Priority:	High
Frequency of Use:	High
Business Rules:	N/A

3.2.3.4.2 View reported accounts

Use Case Specification

UC ID and Name:	UC-035 View reported accounts		
Created by:	LinhLP	Date created:	15/07/2019
Primary Actor:	Administrator	Secondary Actor:	
Trigger:	N/A		
Description:	Allow admin to view reported accounts		
Preconditions:	Admin has successfully logged in the system		
Post conditions:	Reported accounts is displayed to admin		
Normal Flow:	<ol style="list-style-type: none"> 1. Admin click on “Vi phạm” tab at the left side of the page. 2. An expansion options is displayed, then admin click on “Người dùng” tab 3. A list of reported users is displayed 		
Alternative Flows:	N/A		

Exceptions:	N/A
Priority:	High
Frequency of Use:	High
Business Rules:	N/A

3.2.3.4.3 Ban a user

Use Case Specification

UC ID and Name:	UC-036 Ban a user		
Created by:	LinhLP	Date created:	12/07/2019
Primary Actor:	Administrator	Secondary Actor:	
Trigger:	N/A		
Description:	Allow admin to ban a user		
Preconditions:	Admin has successfully logged in the system		
Post conditions:	User's is_active property has been changed to false on database		
Normal Flow:	<ol style="list-style-type: none"> 1. At the admin dashboard page, admin click on “Người dùng” tab. 2. A list of user is displayed 3. Admin click on option on top right of user item. 4. A list of options is displayed 5. Admin click “Định chỉ” option 6. System displays a popup to confirm 7. Admin clicks “Tiếp tục” to continue. 		
Alternative Flows:	N/A		
Exceptions:	N/A		

Priority:	High
Frequency of Use:	High
Business Rules:	N/A

3.2.3.4.4 Unban a user

Use Case Specification

UC ID and Name:	UC-037 Unban a user		
Created by:	LinhLP	Date created:	12/07/2019
Primary Actor:	Administrator	Secondary Actor:	
Trigger:	N/A		
Description:	Allow admin to unban a user		
Preconditions:	Admin has successfully logged in the system		
Post conditions:	User's is_active property has been changed to true on database		
Normal Flow:	<ol style="list-style-type: none"> 1. At the admin dashboard page, admin click on “Người dùng” tab. 2. A list of users is displayed 3. Admin click on option on top right of user item who admin wants to unban. 4. A list of options is displayed 5. Admin click “Bỏ định chỉ” option 6. System displays a popup to confirm 7. Admin clicks “Tiếp tục” to continue. 		
Alternative Flows:	N/A		
Exceptions:	N/A		
Priority:	High		

Frequency of Use:	High
Business Rules:	N/A

3.2.3.4.5 View reported posts

Use Case Specification

UC ID and Name:	UC-038 View reported posts		
Created by:	LinhLP	Date created:	15/07/2019
Primary Actor:	Administrator	Secondary Actor:	
Trigger:	N/A		
Description:	Allow admin to view reported posts		
Preconditions:	Admin has successfully logged in the system		
Post conditions:	A list of reported post is displayed		
Normal Flow:	<ol style="list-style-type: none"> 1. Admin click on “Vi phạm” tab at the left side of the page. 2. An expansion options is displayed, then admin click on “Bài viết” tab 3. A list of reported posts is displayed 		
Alternative Flows:	N/A		
Exceptions:	N/A		
Priority:	High		
Frequency of Use:	High		
Business Rules:	N/A		

3.2.3.4.6 Remove a reported post

Use Case Specification

UC ID and Name:	UC-039 Remove a reported post		
Created by:	LinhLP	Date created:	15/07/2019
Primary Actor:	Administrator	Secondary Actor:	
Trigger:	N/A		
Description:	Allow admin to remove a reported post		
Preconditions:	Admin has successfully logged in the system		
Post conditions:	Post's is_active property has been changed to false on database		
Normal Flow:	<ol style="list-style-type: none"> 1. Admin click on “Vi phạm” tab at the left side of the page. 2. An expansion options is displayed, then admin click on “Bài viết” tab 3. A list of reported posts is displayed 4. Admin click on “...” button on the post item 5. A list of options is displayed, then admin click on “Gỡ bài viết”. 6. A popup is displayed to confirm 7. Admin click on “Tiếp tục” button. 		
Alternative Flows:	N/A		
Exceptions:	N/A		
Priority:	High		
Frequency of Use:	High		
Business Rules:	N/A		

3.2.3.4.7 Restore a removed post

Use Case Specification

UC ID and Name:	UC-040 Restore a removed post		
Created by:	LinhLP	Date created:	15/07/2019
Primary Actor:	Administrator	Secondary Actor:	
Trigger:	N/A		
Description:	Allow admin to restore a post which admin has already removed.		
Preconditions:	Admin has successfully logged in the system		
Post conditions:	Post's <code>is_active</code> property has been changed to true on database		
Normal Flow:	<ol style="list-style-type: none"> 1. Admin click on “Bài viết” tab at the left side of the page 2. A list of posts is displayed 3. Admin click on “...” button on the post item 4. A list of options is displayed, then admin click on “Khôi phục”. 5. A popup is displayed to confirm 6. Admin click on “Tiếp tục” button. 		
Alternative Flows:	N/A		
Exceptions:	N/A		
Priority:	High		
Frequency of Use:	High		
Business Rules:	N/A		

3.2.3.4.8 View reported comments

Use Case Specification

UC ID and Name:	UC-041 View reported comments		
Created by:	LinhLP	Date created:	18/07/2019
Primary Actor:	Administrator	Secondary Actor:	
Trigger:	N/A		
Description:	Allow admin to restore a removed post		
Preconditions:	Admin has successfully logged in the system		
Post conditions:	A list of reported comments is displayed		
Normal Flow:	<ol style="list-style-type: none"> 1. Admin click on “Vi phạm” tab at the left side of the page. 2. An expansion options is displayed, then admin click on “Bình luận” tab 3. A list of reported comments is displayed 		
Alternative Flows:	N/A		
Exceptions:	N/A		
Priority:	High		
Frequency of Use:	High		
Business Rules:	N/A		

3.2.3.4.9 Remove a reported comment

Use Case Specification

UC ID and Name:	UC-042 Remove a reported comment		
Created by:	LinhLP	Date created:	18/07/2019
Primary Actor:	Administrator	Secondary Actor:	

Trigger:	N/A
Description:	Allow admin to remove a reported comment.
Preconditions:	Admin has successfully logged in the system
Post conditions:	Comment's is_active property is changed to false.
Normal Flow:	<ol style="list-style-type: none"> 1. Admin click on “Vi phạm” tab at the left side of the page. 2. An expansion options is displayed, then admin click on “Bình luận” tab 3. A list of comments is displayed 4. Admin click on “...” button on the comment item 5. A list of options is displayed, then admin click on “Gỡ bình luận”. 6. A popup is displayed to confirm 7. Admin click on “Tiếp tục” button.
Alternative Flows:	N/A
Exceptions:	N/A
Priority:	High
Frequency of Use:	High
Business Rules:	N/A

3.2.3.4.10 Restore a removed comment

Use Case Specification

UC ID and Name:	UC-043 Restore a removed comment		
Created by:	LinhLP	Date created:	18/07/2019

Primary Actor:	Administrator	Secondary Actor:	
Trigger:	N/A		
Description:	Allow admin to restore a removed comment		
Preconditions:	Admin has successfully logged in the system		
Post conditions:	Comment's is_active property is changed to true.		
Normal Flow:	<ol style="list-style-type: none"> 1. Admin click on “Bình luận” tab at the left side of the page 2. A list of comments is displayed 3. Admin click on “...” button on the comment item 4. A list of options is displayed, then admin click on “Khôi phục”. 5. A popup is displayed to confirm 6. Admin click on “Tiếp tục” button. 		
Alternative Flows:	N/A		
Exceptions:	N/A		
Priority:	High		
Frequency of Use:	High		
Business Rules:	N/A		

3.2.3.4.11 Add a travel topic

Use Case Specification

UC ID and Name:	UC-044 Add a travel topic		
Created by:	LinhLP	Date created:	05/07/2019
Primary Actor:	Administrator	Secondary Actor:	

Trigger:	N/A
Description:	Allow admin to add a travel topic
Preconditions:	Admin has successfully logged in the system
Post conditions:	A travel topic is added on database
Normal Flow:	<ol style="list-style-type: none"> 1. At the admin dashboard page, admin click on “Chủ đề” tab at the left side of the page 2. A list of topic is displayed 3. Admin click “Thêm” button 4. System displays a popup form to add topic 5. Admin fill the form and click “OK” button
Alternative Flows:	N/A
Exceptions:	N/A
Priority:	High
Frequency of Use:	High
Business Rules:	N/A

3.2.3.4.12 Edit a travel topic

Use Case Specification

UC ID and Name:	UC-045 Edit a travel topic		
Created by:	LinhLP	Date created:	05/07/2019
Primary Actor:	Administrator	Secondary Actor:	
Trigger:	N/A		
Description:	Allow admin to edit a travel topic		

Preconditions:	Admin has successfully logged in the system
Post conditions:	Travel topic's information is updated on database
Normal Flow:	<ol style="list-style-type: none"> 1. At the admin dashboard page, admin click on “Chủ đề” tab at the left side of the page 2. A list of topic is displayed 3. Admin choose one topic admin want to edit. 4. Admin click on “Sửa” button 5. System displays a popup form to edit topic information 6. Admin fill the form and click “OK” button
Alternative Flows:	N/A
Exceptions:	N/A
Priority:	High
Frequency of Use:	High
Business Rules:	N/A

3.2.3.4.13 Delete a travel topic

Use Case Specification

UC ID and Name:	UC-046 Delete a travel topic		
Created by:	LinhLP	Date created:	05/07/2019
Primary Actor:	Administrator	Secondary Actor:	
Trigger:	N/A		
Description:	Allow admin to delete a travel topic		
Preconditions:	Admin has successfully logged in the system		

Post conditions:	Topic's is_active property is changed to false.
Normal Flow:	<ol style="list-style-type: none"> 1. At the admin dashboard page, admin click on “Chủ đề” tab at the left side of the page 2. A list of topic is displayed 3. Admin choose one or more topics admin wants to delete. 4. Admin click on “Xóa” button 5. System displays a popup form to edit topic information 6. Admin fill the form and click “OK” button
Alternative Flows:	N/A
Exceptions:	N/A
Priority:	High
Frequency of Use:	High
Business Rules:	N/A

3.2.3.4.14 View statistic of users

Use Case Specification

UC ID and Name:	UC-047 View statistic of users		
Created by:	ThaiHV	Date created:	12/07/2019
Primary Actor:	Administrator	Secondary Actor:	
Trigger:	N/A		
Description:	Allow admin to view statistic of new registered user in day, month, year		
Preconditions:	User has signed in as Administrator		

Post conditions:	Chart of users' statistic is displayed.
Normal Flow:	<ol style="list-style-type: none"> 1. User visits admin dashboard page. 2. User click on “Tổng quan” tab at the left side of the page 3. A chart of users statistic is displayed
Alternative Flows:	N/A
Exceptions:	N/A
Priority:	High
Frequency of Use:	High
Business Rules:	N/A

3.2.3.4.15 View statistic of posts

Use Case Specification

UC ID and Name:	UC-056 View statistic of posts		
Created by:	ThaiHV	Date created:	12/07/2019
Primary Actor:	Administrator	Secondary Actor:	
Trigger:	N/A		
Description:	Allow admin to view statistic of new post user in day, month, year		
Preconditions:	User has signed in as Administrator		
Post conditions:	Chart of posts' statistic is displayed.		
Normal Flow:	<ol style="list-style-type: none"> 1. User visits admin dashboard page. 2. User click on “Tổng quan” tab at the left side of the page 3. A chart of posts statistic is displayed 		

Alternative Flows:	N/A
Exceptions:	N/A
Priority:	High
Frequency of Use:	High
Business Rules:	N/A

3.2.3.5 User

3.2.3.5.1 Search posts

Use Case Specification

UC ID and Name:	UC-049 Search posts		
Created by:	LinhLP	Date created:	05/07/2019
Primary Actor:	User	Secondary Actor:	
Trigger:	N/A		
Description:	Allow user to search posts by location or by text		
Preconditions:	N/A		
Post conditions:	Posts related to search parameters is displayed		
Normal Flow:	Search by location: 1. At the homepage, user type destination in search box “Bạn muôn đi đâu” 2. User chose destination from drop down list 3. A list of posts is displayed		
	Search by text:		

	<ol style="list-style-type: none"> 1. User enter text parameter on the search box at the top-right of the page 2. A list of posts is displayed
Alternative Flows:	N/A
Exceptions:	Google Map API can't return list of location while user typing in the search box.
Priority:	High
Frequency of Use:	High
Business Rules:	N/A

3.2.3.5.2 Search a member

Use Case Specification

UC ID and Name:	UC-050 Search a member		
Created by:	LinhLP	Date created:	05/07/2019
Primary Actor:	User	Secondary Actor:	
Trigger:	N/A		
Description:	Allow user to search a member		
Preconditions:	N/A		
Post conditions:	Member search result is displayed		
Normal Flow:	<ol style="list-style-type: none"> 1. User type text parameter into search box top-right of screen and hit enter 2. Result is displayed 3. User click “Mọi người” tab 4. A list of member related to search name is displayed 		

Alternative Flows:	N/A
Exceptions:	N/A
Priority:	High
Frequency of Use:	High
Business Rules:	N/A

3.2.3.5.3 View popular articles

Use Case Specification

UC ID and Name:	UC-051 View popular articles		
Created by:	TruongLX	Date created:	13/06/2019
Primary Actor:	User	Secondary Actor:	
Trigger:	N/A		
Description:	Allow user to view popular articles.		
Preconditions:	User visit the website.		
Post conditions:	A list of popular articles is displayed.		
Normal Flow:	<ol style="list-style-type: none"> 1. User visit website. 2. User see some post on “phố biến” section at home page. 3. Click “Xem thêm >” link on the top right of section. 4. System display popular articles. 		
Alternative Flows:	N/A		
Exceptions:	N/A		
Priority:	High		
Frequency of Use:	High		
Business Rules:	N/A		

3.2.3.5.4 View recently posts

Use Case Specification

UC ID and Name:	UC-052 View recently posts		
Created by:	TruongLX	Date created:	13/06/2019
Primary Actor:	User	Secondary Actor:	
Trigger:	N/A		
Description:	Allow user to view the newest post		
Preconditions:	Guest visit the website.		
Post conditions:	A list of newest posts is displayed.		
Normal Flow:	<ol style="list-style-type: none"> 1. User visit website. 2. User see some post on “Mới nhất” section at home page. 3. Click “Xem thêm >” link on the top right of section. 4. System display popular posts. 		
Alternative Flows:	N/A		
Exceptions:	N/A		
Priority:	High		
Frequency of Use:	High		
Business Rules:	N/A		

3.2.3.5.5 View a post detail

Use Case Specification

UC ID and Name:	UC-053 View a post detail		
Created by:	TruongLX	Date created:	11/06/2019
Primary Actor:	User	Secondary Actor:	

Trigger:	N/A
Description:	Allow user to view a post detail
Preconditions:	User has visited the website
Post conditions:	An post detail page is displayed
Normal Flow:	<ol style="list-style-type: none"> 1. User visit website. 2. From list-post page, user clicks on a post's title or post's cover image. 3. A post detail page displayed.
Alternative Flows:	N/A
Exceptions:	N/A
Priority:	High
Frequency of Use:	High
Business Rules:	N/A

3.2.3.5.6 View user's profile

Use Case Specification

UC ID and Name:	UC-054 View user's profile		
Created by:	TruongLX	Date created:	01/06/2019
Primary Actor:	User	Secondary Actor:	
Trigger:	N/A		
Description:	Allow user to view profile of other users.		
Preconditions:	If user has signed in		

Post conditions:	A profile page of member displayed.
Normal Flow:	<ol style="list-style-type: none"> 1. User visit website. 2. At the list of posts/users, or at the detail post page, user click on user's name or avatar. 3. Browser redirects to user's personal page. 4. User's profile is displayed at the left side of the page
Alternative Flows:	N/A
Exceptions:	N/A
Priority:	High
Frequency of Use:	High
Business Rules:	N/A

3.2.3.5.7 View user's followings/followers

Use Case Specification

UC ID and Name:	UC-055 View user's followings/followers		
Created by:	TruongLX	Date created:	20/06/2019
Primary Actor:	User	Secondary Actor:	
Trigger:	N/A		
Description:	Allow user to view other users' followings/followers.		
Preconditions:	User has signed in		
Post conditions:	A list of followings/followers is displayed.		

Normal Flow:	<ol style="list-style-type: none"> 1. User visit user's personal page. 2. User click on “n - đang theo dõi” or “n - theo dõi” link on the left side panel. 3. A list of followers/followings displayed.
Alternative Flows:	N/A
Exceptions:	N/A
Priority:	High
Frequency of Use:	High
Business Rules:	N/A

3.2.3.5.8 View user's created posts

Use Case Specification

UC ID and Name:	UC-056 View user's created posts		
Created by:	TruongLX	Date created:	20/06/2019
Primary Actor:	User	Secondary Actor:	
Trigger:	N/A		
Description:	Allow user to view created posts of other users.		
Preconditions:	If user has signed in		
Post conditions:	A list of created posts is displayed.		
Normal Flow:	<ol style="list-style-type: none"> 1. User visit user's personal page. 2. User choose list post he/she wants to see 3. A list of posts is displayed 		
Alternative Flows:	N/A		

Exceptions:	N/A
Priority:	High
Frequency of Use:	High
Business Rules:	N/A
Other Information:	N/A
Assumptions:	N/A

3.3 Non-Functional Requirements

3.3.1 Security

- ✓ Token-Based Authentication, relies on a signed token that is sent to the server on each request.
- ✓ All requests are secured by HTTPS protocol.
- ✓ Because we use Token-Based Authentication, so all request must be made on the correct frontend website otherwise it will be rejected, which made Cross-Site Request Forgery (CSRF) is impossible.
- ✓ All query made to MongoDB must be escaped so no NoSQL Injection.
- ✓ Apply SSL encryption to restrict user access and prevent man-in-the-middle attacking.
- ✓ All passwords are encrypted by PBKDF2 algorithm with salt.
- ✓ The security matrix is as following table:

Function	Guest	Member	Administrator
Sign up	✓		
Google sign in		✓	✓
Facebook sign in		✓	✓
Normal sign in		✓	✓
Forgot password		✓	✓
Sign out		✓	✓
Change password		✓	✓
Update profile		✓	✓
Choose interested topic		✓	

Update interested topic		✓	
Create a post		✓	
Edit a post		✓	
Delete a post		✓	
Like/Unlike a post		✓	
Share a post		✓	
Report a post		✓	
Bookmark a post		✓	
View all bookmarks		✓	
Remove a bookmark		✓	
Comment to a post		✓	
Edit a comment		✓	
Delete a comment		✓	
Like/Unlike a comment		✓	
Report a comment		✓	
Create a virtual trip		✓	
Edit a virtual trip		✓	
Delete a virtual trip		✓	
Follow a user		✓	
Unfollow a user		✓	
Report a user		✓	
Create a finding companions post		✓	
Edit a finding companions post		✓	
Delete a finding companions post		✓	
Join to a companion group		✓	
Leave a companion group		✓	
Remove a joined companion		✓	
Send/receive messages		✓	
View followers' posts		✓	
View recommended posts		✓	

View all users' account			✓
View reported accounts			✓
Ban a user			✓
Unban a user			✓
View reported posts			✓
Remove a reported post			✓
Restore a removed post			✓
View reported comments			✓
Remove a reported comment			✓
Restore a removed comment			✓
Add a travel topic			✓
Edit a travel topic			✓
Delete a travel topic			✓
View statistic of users			✓
View statistic of posts			✓
Search post by travel destinations	✓	✓	✓
Search a member	✓	✓	✓
Search finding companion posts	✓	✓	✓
View popular posts	✓	✓	✓
View recently posts	✓	✓	✓
View a post detail	✓	✓	✓
View a virtual trip	✓	✓	✓
View member's profile	✓	✓	✓
View member's followings/followers	✓	✓	✓
View member's created posts	✓	✓	✓
View member's created virtual trips	✓	✓	✓
View member's created finding companions' posts	✓	✓	✓

3.3.2 Maintainability & Extensibility

- Microservices architecture for backend.
- Unit Tests and Integration Tests are written using NUnit.
- Using TSLint for a consistent coding convention.
- Maintainable source code and easy deployment since we follow a single coding convention & a highly maintainable architecture design.
- Object Oriented Programming paradigm is applied in order to ensure scalability & maintainability.

3.3.3 Availability and Scalability

- Google Cloud service for automatic horizontal scalability.
- Third-party services for convenient development:
 - SendGrid for sending emails.
 - Google storage for storing images.

3.3.4 Performance

- Web front-end is a Single Page Application with the goal of providing a more fluid user experience similar to a desktop application.
- Backend system uses background job for external requests, heavy processing requests such as:
 - Sending email

3.3.5 Usability

- The interface should be elegant and simple.
- Alt attributes are provided for non-text elements, such as images and maps.
- Links, buttons and checkboxes are easily clickable, for example a user can select a checkbox by clicking the text, not just the checkbox.
- Search box is wide enough, so that users can see what they've typed.
- Search is available on every page, not just the homepage.
- Links are easily recognizable. They look clickable. Items that aren't links don't look clickable, for example underlining text is avoided.
- Important commands are displayed as buttons, not links. For example, "Đăng nhập" or "Đăng kí" is a button, not a link.

Chapter 4: Software Design

4.1 Purpose

This chapter is to give the development team aware of what the system architecture is, and how they should be implemented. This chapter consists of:

- ✓ Architecture overview
- ✓ Detailed design
- ✓ Database design

4.2 Architecture Overview

4.2.1 System Architecture

4.2.1.1 Diagram

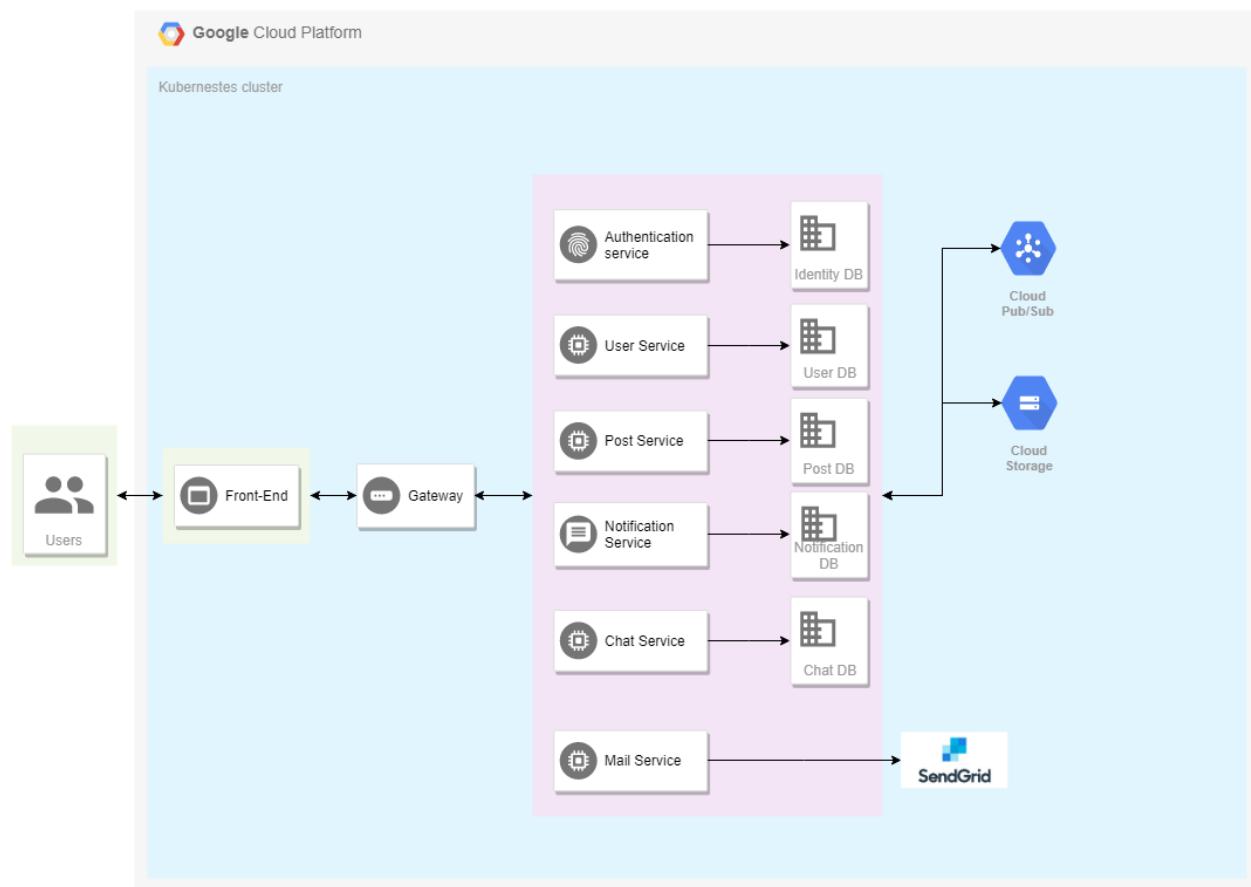


Figure 4-1: TripSharing System Architecture

4.2.2 System Architecture Explanation

The entire project will be deployed on Google Cloud. We aim at delivering a secured, responsive, and highly available system. In the following section, we will explain the function and mechanism of each unit in the system architecture design.

4.2.2.1 Google Cloud



Google Cloud

Figure 4-5: Google cloud

Google Cloud Platform is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products. Alongside a set of management tools, it provides a series of modular cloud services including computing, data storage, data analytics and machine learning. Apart from Cloud Storage and Pub/Sub mentioned above, the main part of operation lies in the usage of the Kubernetes Engine that supports easy deployment and scaling from docker images.

4.2.2.2 Google Cloud Storage



Figure 4-2: Google cloud storage

Google Cloud Storage is a web-services interface data storage & retrieval. We use Google Cloud Storage for storing images.

4.2.2.3 Google Cloud Pub/Sub



Figure 4-3: Google cloud pub/sub

Cloud Pub/Sub is a fully-managed real-time messaging service that allows you to send and receive messages between independent applications. We use Cloud Pub/Sub to handle communication among microservices.

4.2.2.4 Docker

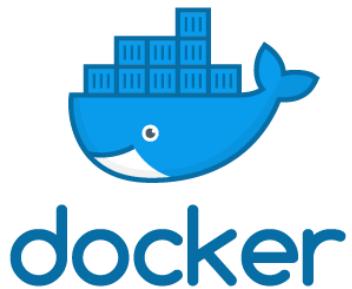


Figure 4-4: Docker

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package. We use Docker to build and push images to Docker Hub for deployment and testing.

4.2.2.5 Nginx



Figure 4-6: Nginx

NGINX is a free, open-source, high-performance HTTP server and reverse proxy, as well as an IMAP/POP3 proxy server. NGINX is known for its high performance, stability, rich feature set, simple configuration, and low resource consumption. We use Nginx to serve the frontend service of TripSharing.

4.2.2.6 MongoDB



Figure 4-7: MongoDB

MongoDB is an open source database and is the leading NoSQL database, used by millions of people. MongoDB is written in C++. In addition, MongoDB is a cross-platform database, operating on the concepts of Collection and Document, it provides high performance, high availability and easy scalability.

4.2.2.7 Angular 7



Figure 4-8: Angular 7

Angular 7 is the library of Angular, a framework created by Google that supports Flat & Material design website. Supporting Progressive Web Application, this framework has become very popular nowadays.

4.2.2.8 Angular Material



Figure 4-9: Angular Material

Angular Material is a great JavaScript framework for creating themed UI components that ignite good user experience.

4.2.2.9 C#



Figure 4-10: C# (C Sharp)

C# is a general object-oriented programming (OOP) language for networking and Web development. C# is specified as a common language infrastructure (CLI) language. Its .NET framework promotes multiple Web technologies. We use C# to write back-end code.

4.2.2.10 ASP.NET Core



Figure 4-11: ASP.NET Core

ASP.NET Core is a cross-platform, high-performance, open-source framework for building modern, cloud-based, Internet-connected applications. We use ASP.NET Core to write back-end API services.

4.2.2.11 ASP.NET Core SignalR



Figure 4-12: SignalR

ASP.NET Core SignalR is an open-source library that simplifies adding real-time web functionality to apps. Real-time web functionality enables server-side code to push content to clients instantly. We use SignalR for chat and notification.

4.2.2.12 SendGrid



Figure 4-13: SendGrid

SendGrid is a platform for transactional and marketing email. We use SendGrid to send emails to users for verification.

4.3 Design of TripSharing System

4.3.1 Architecture Layers Design

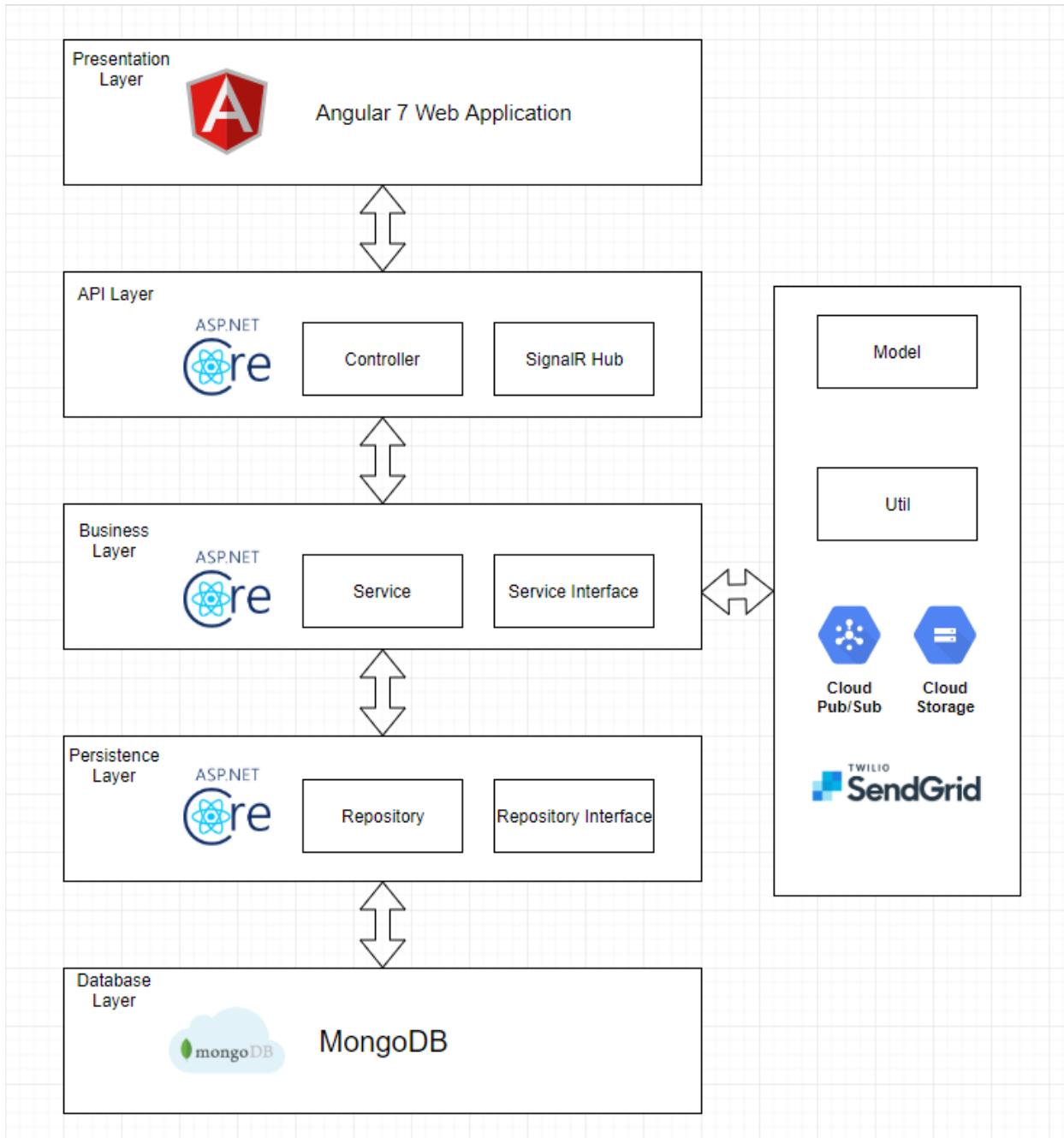


Figure 4-14: Architecture Layer Design

4.3.2 Database Design

4.3.2.1 Explanation of database decision

4.3.2.1.1 Operational Database – MongoDB



MongoDB is our first choice for storing operational data because it has powerful features that suit our needs:

- ✓ The very basic feature of MongoDB is that it is a schema-less database. No schema migrations anymore.
- ✓ Stores the data in the form of BSON (Binary JSON), ruby hashes etc., helps to store the data in a very rich way while being capable of holding arrays and other documents.
- ✓ Reliability, Performance & Ease of Use.

4.3.2.2 Database Diagram

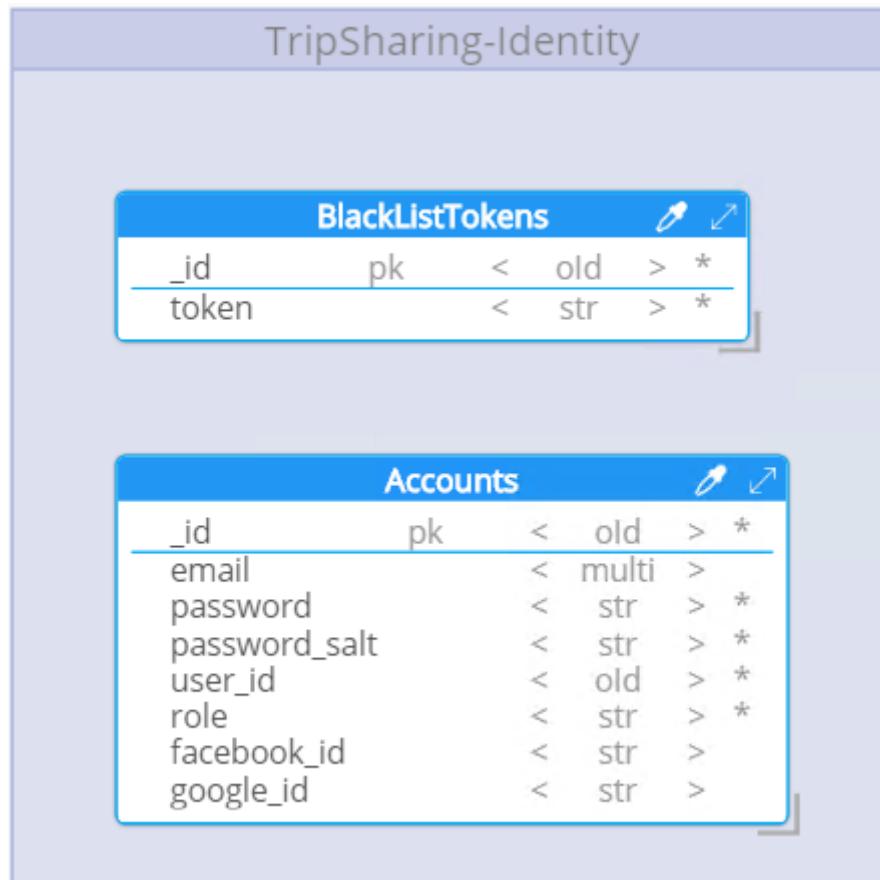


Figure 4-15: Identity Provider Database Design

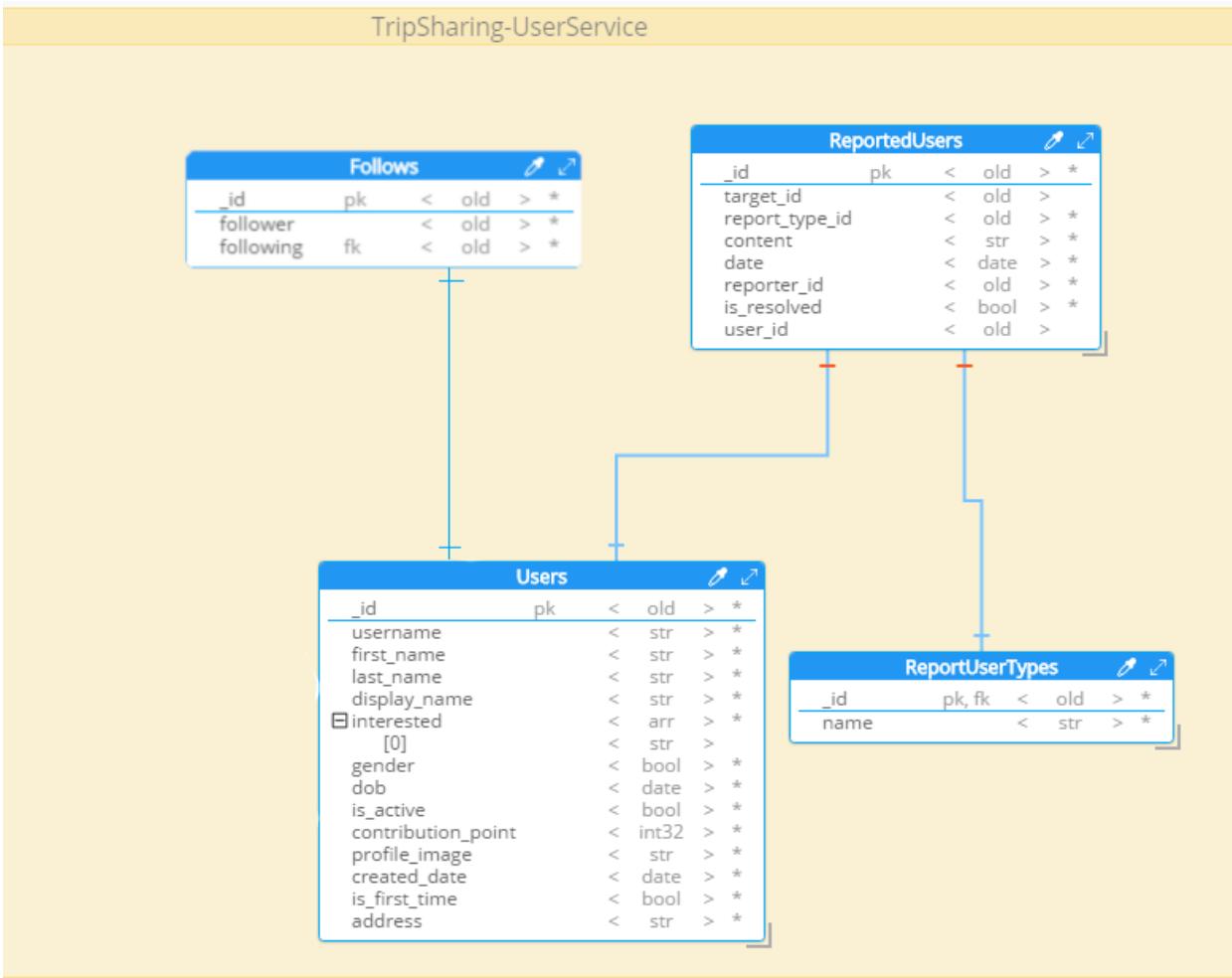


Figure 4-16: User Database Design

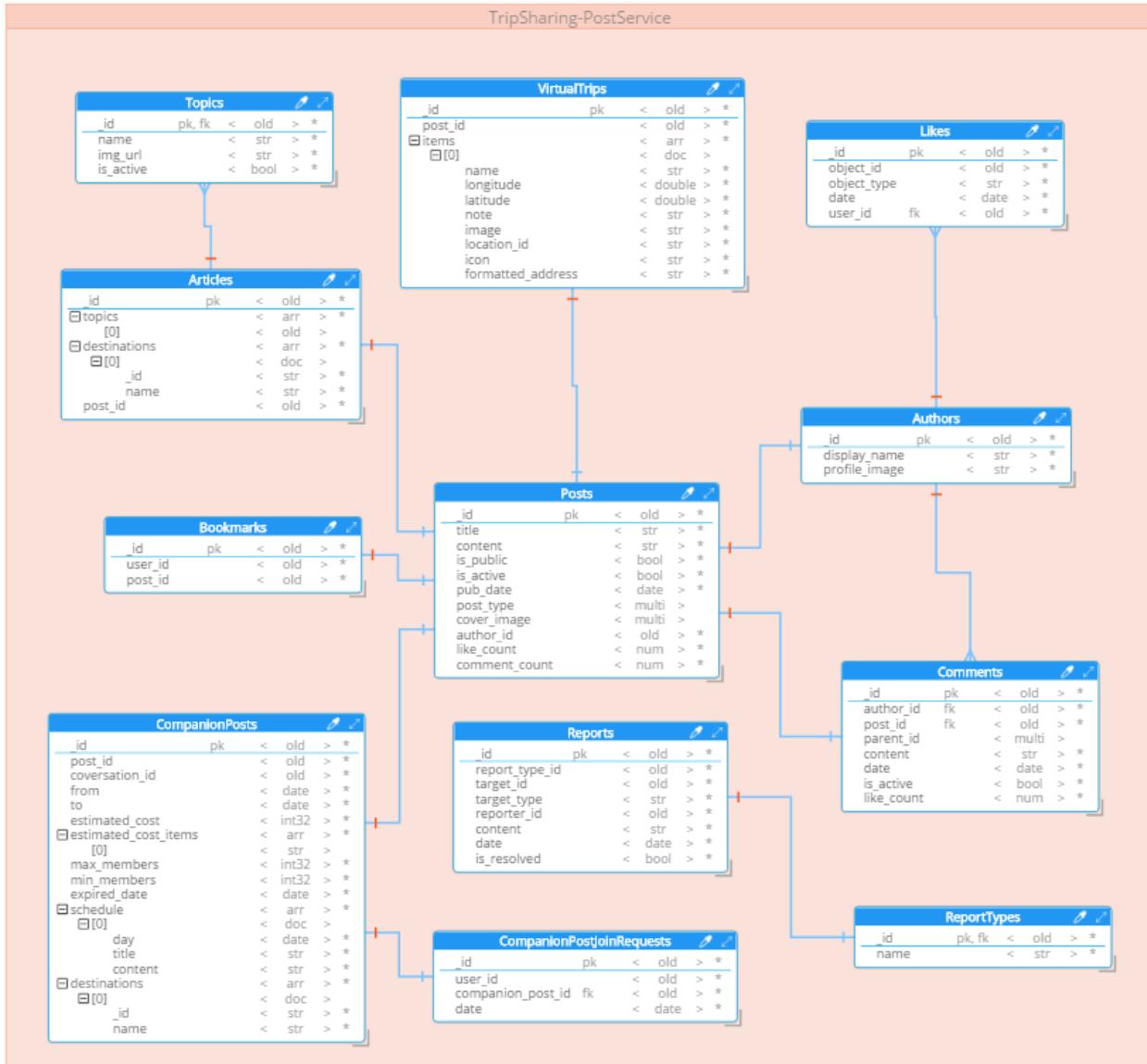


Figure 4-17: Post Database Design

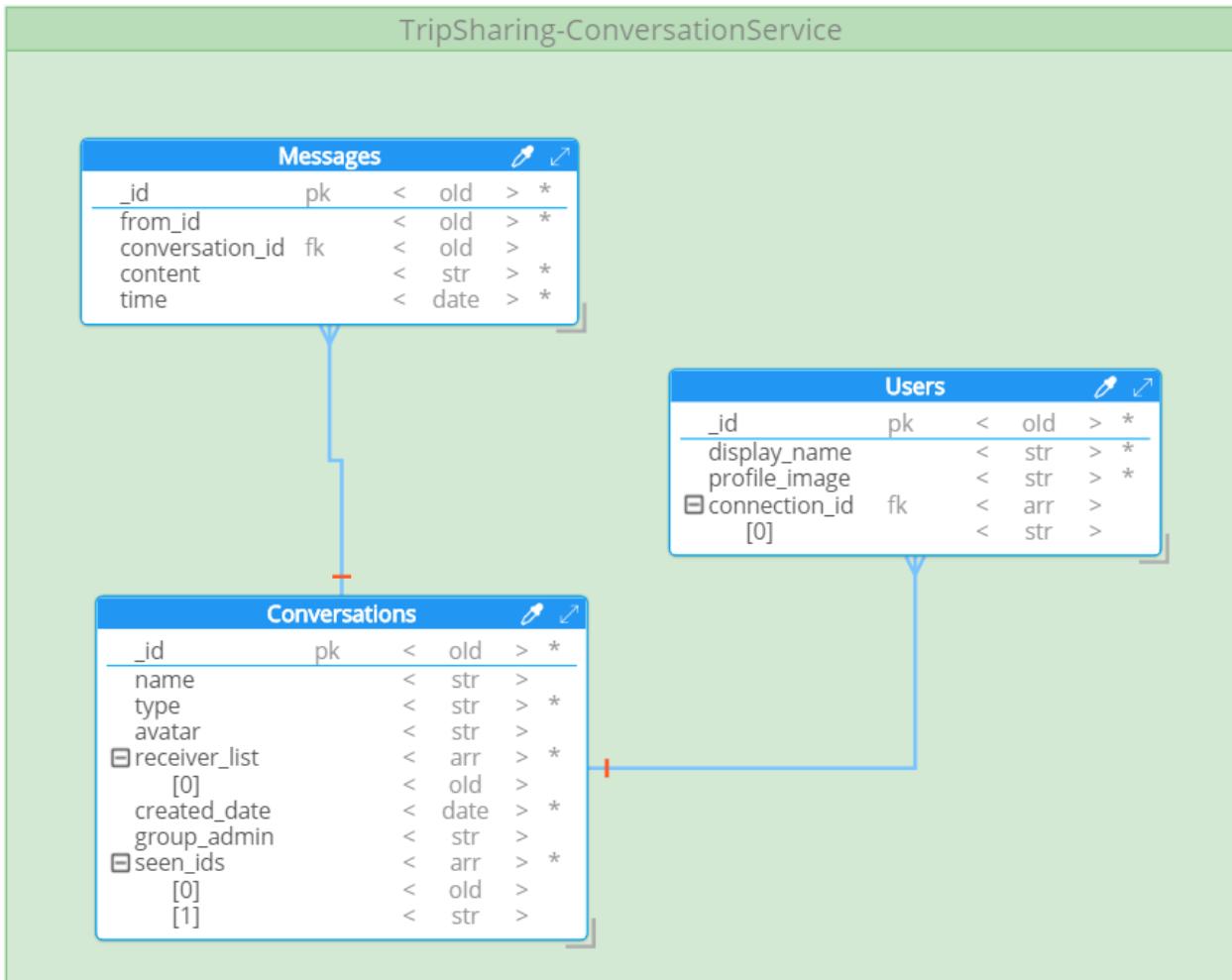


Figure 4-18: Conversation Database Design

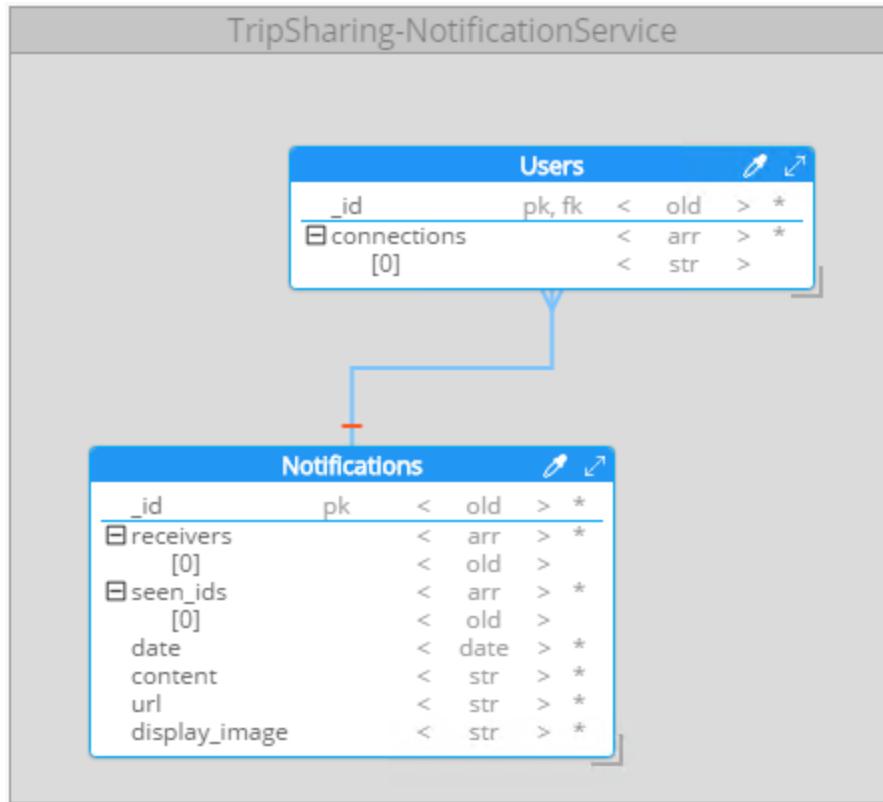


Figure 4-19: Notification Database Design

4.3.2.3 Data Dictionary

4.3.2.3.1 TripSharing – Operational database

4.3.2.3.1.1 Entities

No	Entity Name	Description	Alias	Occurrence
1	Users	General term describing a user profiles.		Each user has one account. Each user has many interested topics.
2	Follows	General term describing a follow relation between users.		Each user has many followers. Each user can follow many others.
3	Bookmarks	General term describing bookmarks of each user.		Each user has many bookmarks.
4	Accounts	General term describing a registered account of a user.		Each account belongs to one user. Each account has one email.

5	Posts	General term describing a post.		Each post has one type (companion, article or virtual trip). Each post has many comments. Each post has many likes. Each post has one author.
6	Likes	General term describing a “like” action of user.		Each user can like many posts or comments.
7	Comments	General term describing comments of user.		Each user has many comments. Each comment has many child comments.
8	Author	General term describing a user who is author of a post.		Each user has many posts.
9	Articles	General term describing an article type of post.		Each article has one post. Each article has many topics. Each article has many destinations.
10	Topics	General term describing a topic of an article.		
11	VirtualTrips	General term describing a virtual trip type of post.		Each virtual trip has many items, each item contains destination information.
12	CompanionPosts	General term describing a companion type of post.		Each companion post has one conversation.
13	CompanionJoinRequest	General term describing a request to join companion.		
14	Notifications	General term describing a notification.		Each notification has many receivers(user).
15	Conversations	General term describing a conversation between two or more users.		Each conversation has many receivers(user). Each conversation has many messages.
16	Messages	General term describing a message send to conversation.		Each message has one sender(user).

17	BlackListTokens	General term describing a token that no longer valid		Token will be moved to black list if user sign-out.
----	-----------------	--	--	---

4.3.2.3.1.2 Attributes

No	Entities Name	Field Name	Data Type	Description	Not Null
1	Users	_id	Object Id		Yes
		account_id	Object Id		Yes
		username	String		Yes
		first_name	String		Yes
		last_name	String		Yes
		display_name	String		Yes
		interested	Array	User interested topic	Yes
		gender	Boolean		Yes
		dob	Date		Yes
		is_active	Boolean	Whether user is banned.	Yes
		contribution_point	Integer		Yes
		profile_image	String		Yes
		created_date	Date		Yes
		is_first_time	Boolean	Whether user is first time sign-in.	Yes
		address	String		Yes
2	Accounts	_id	Object Id		Yes
		email	String		Yes
		password	String	Encrypted password	Yes
		password_salt	string		Yes
		user_id	Object Id		Yes
		role	String	Role of user	Yes
3	BlackListTokens	_id	Object Id		Yes
		token	String		Yes
4	Follows	_id	Object Id		Yes
		follower	Object Id		Yes
		following	Object Id		Yes
5	Bookmarks	_id	Object Id		Yes
		user_id	Object Id		Yes
		post_id	Object Id		Yes
		title	String		Yes
		cover_image	String		Yes
		post_type	String		Yes
		title	String		Yes

6	Authors	<u>_id</u>	Object Id		Yes
		<u>display_name</u>	String		Yes
		<u>profile_image</u>	String		Yes
7	Posts	<u>_id</u>	Object Id		Yes
		<u>title</u>	String		Yes
		<u>content</u>	String		Yes
		<u>is_public</u>	Boolean	Whether post is show for another user or just author.	Yes
		<u>is_active</u>	Boolean	Whether post is banned.	Yes
		<u>pub_date</u>	Date		Yes
		<u>post_type</u>	String		Yes
		<u>author_id</u>	Object Id		Yes
		<u>like_count</u>	Integer		Yes
		<u>comment_count</u>	Integer		Yes
8	Articles	<u>cover_image</u>	String		Yes
		<u>_id</u>	Object Id		Yes
		<u>topics</u>	Array	An array of string to store topics.	Yes
		<u>destinations</u>	Array	An array of object store information about destinations on the trip.	Yes
9	Topics	<u>post_id</u>	Object Id		Yes
		<u>_id</u>	Object Id		Yes
		<u>name</u>	String		Yes
10	VirtualTrips	<u>img_url</u>	String		Yes
		<u>_id</u>	Object Id		Yes
		<u>post_id</u>	Object Id		Yes
		<u>items</u>	Array	List of object store information about destinations on the trip.	Yes
11	CompanionPosts	<u>_id</u>	Object Id		Yes
		<u>post_id</u>	Object Id		Yes
		<u>conversation_id</u>	Object Id		Yes
		<u>from</u>	Date		Yes
		<u>to</u>	Date		Yes
		<u>estimated_cost</u>	Double		Yes
		<u>estimated_cost_item</u>	Array		Yes
		<u>max_members</u>	Integer		Yes
		<u>min_members</u>	Integer		Yes
		<u>expired_date</u>	Date		Yes

		schedule	Array	List of notes what to do on the trip, added by user.	Yes
		destinations	Array	List of object store information about destinations on the trip.	Yes
12	CompanionPostJoinRequest	_id	Object Id		Yes
		user_id	Object Id		Yes
		companion_post_id	Object Id		Yes
		date	Date		Yes
13	Likes	_id	Object Id		Yes
		object_id	Object Id		Yes
		object_type	String	Whether user like post or comment	Yes
		date	Date		Yes
		user_id	Object Id		Yes
14	Comments	_id	Object Id		Yes
		Author_id	Object Id		Yes
		Post_id	Object Id		Yes
		Parent_id	Object Id		No
		Content	String		Yes
		Date	Date		Yes
		Is_active	Boolean	Whether comment is deleted	Yes
		Like_count	Integer		Yes
15	Conversation	_id	Object Id		Yes
		name	String		Yes
		type	String		Yes
		receiver_list	Array	List of users receive message from conversation	Yes
		last_message	Object		Yes
16	Messages	_id	Object Id		Yes
		from_id	Object Id		Yes
		conversation_id	Object Id		Yes
		content	String		Yes
		time	Date		Yes
		seen_id	Object Id	List of users already see the message.	Yes
17	Notifications	_id	Object Id		Yes
		receivers	Array	List of users receive notification.	Yes

	seen_ids	Array	List of users already see the notification.	Yes
	date	Date		Yes
	content	String		Yes
	url	String		Yes
	display_image	String		Yes

4.3.3 Common Design

4.3.3.1 Microservice Architecture

4.3.3.1.1 What is Microservice?

Microservices is a software architecture style in which complex applications are composed of small, independent processes communicating with each other using language-agnostic APIs. These services are small, highly decoupled and focus on doing a small task, facilitating a modular approach to system-building.

4.3.3.1.2 Philosophy

Philosophy of Microservices architecture essentially equals the Unix philosophy of "Do one thing and do it well". It is described as follows:

- The services are small - fine-grained to perform a single function.
- The organization culture should embrace automation of deployment and testing. This eases the burden on management and Operations.
- The culture and design principles should embrace failure and faults, similar to anti-fragile systems.
- Each service is elastic, resilient, composable, minimal, and complete.

4.3.3.1.3 Microservice in TripSharing

In TripSharing, we implement Microservices as the following things:

- The overall system is divided into small microservices according to their domain of operation.
- Deployment of services is done by using Kubernetes.
- Use third-party services: SendGrid for sending mail.

4.3.4 Detail Design

4.3.4.1 Sign-up

Screen design

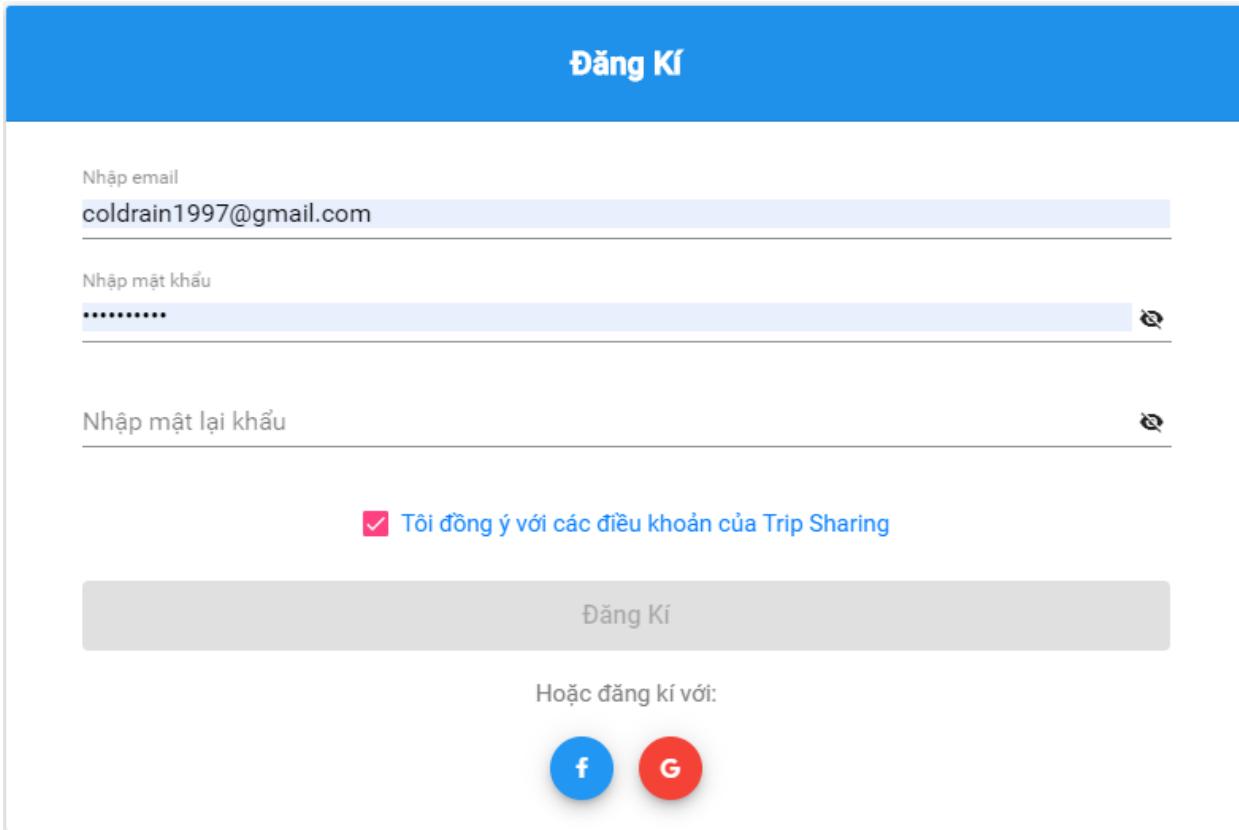


Figure 4-20: Sign up screen

Class Diagram

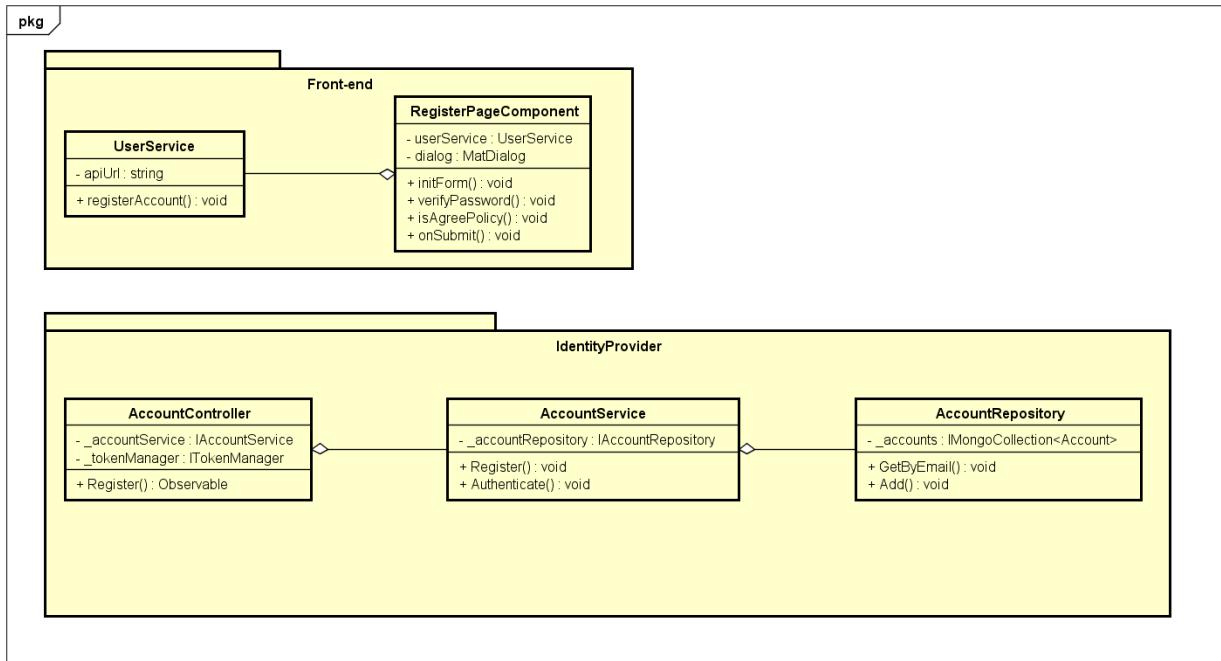


Figure 4-21: Sign-up Class Diagram

Class Specification

RegisterPageComponent

RegisterPageComponent			
Physical address	src\app\pages\register-page\ register-page.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	userService	UserService	
2	dialog	MatDialog	
Operations			
No	Name	Return Type	Description
1	initForm	Void	Create sign-up form
2	verifyPassword	Void	Verify if re-enter password match
3	isAgreePolicy	Void	Check if guest agree with policy
4	onSubmit	Void	Submit register form

UserService

UserService			
Physical address	src\app\core\services\user-service\user.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	apiURL	String	
Operations			
No	Name	Return Type	Description
1	registerAccount	Void	

AccountController

AccountController			
Physical address	\src\Services\IdentityProvider\IdentityProvider\Controllers\AccountController.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	_accountService	IAccountService	
2	_tokenManager	ITokenManager	
Operations			
No	Name	Return Type	Description

1	Register	Observable	
---	----------	------------	--

AccountService

AccountService			
Physical address	\src\Services\IdentityProvider\IdentityProvider\Services\AccountService.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	_accountRepository	IAccountRepository	
Operations			
No	Name	Return Type	Description
1	Register	Void	
2	Authenticate	Void	

AccountRepository

AccountRepository			
Physical address	\src\Services\IdentityProvider\IdentityProvider\Repositories\AccountRepository.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	_accounts	IMongoCollection<Account>	
Operations			
No	Name	Return Type	Description
1	GetByEmail	Void	
2	Add	Void	

Sequence Diagram

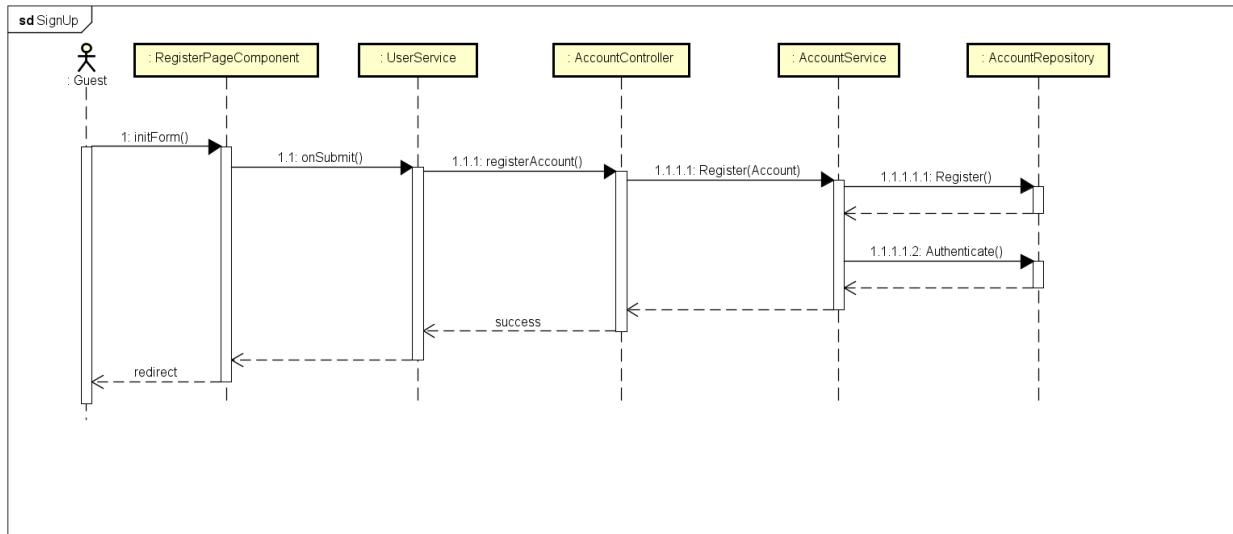


Figure 4-22: Sign-up Sequence Diagram

4.3.4.2 Google Sign-in

Screen Design

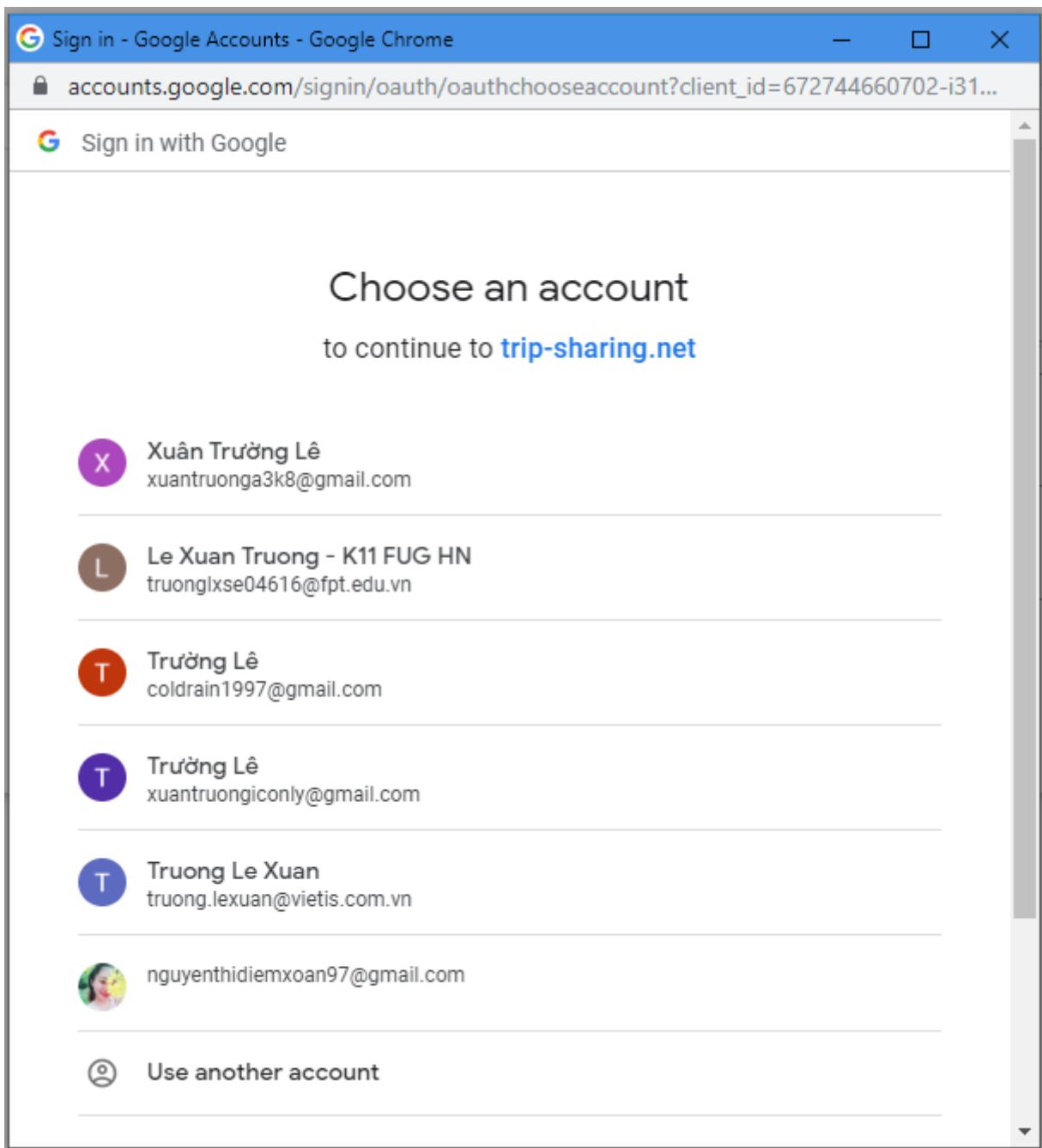


Figure 4-23: Google Sign in Screen Design

Class Diagram

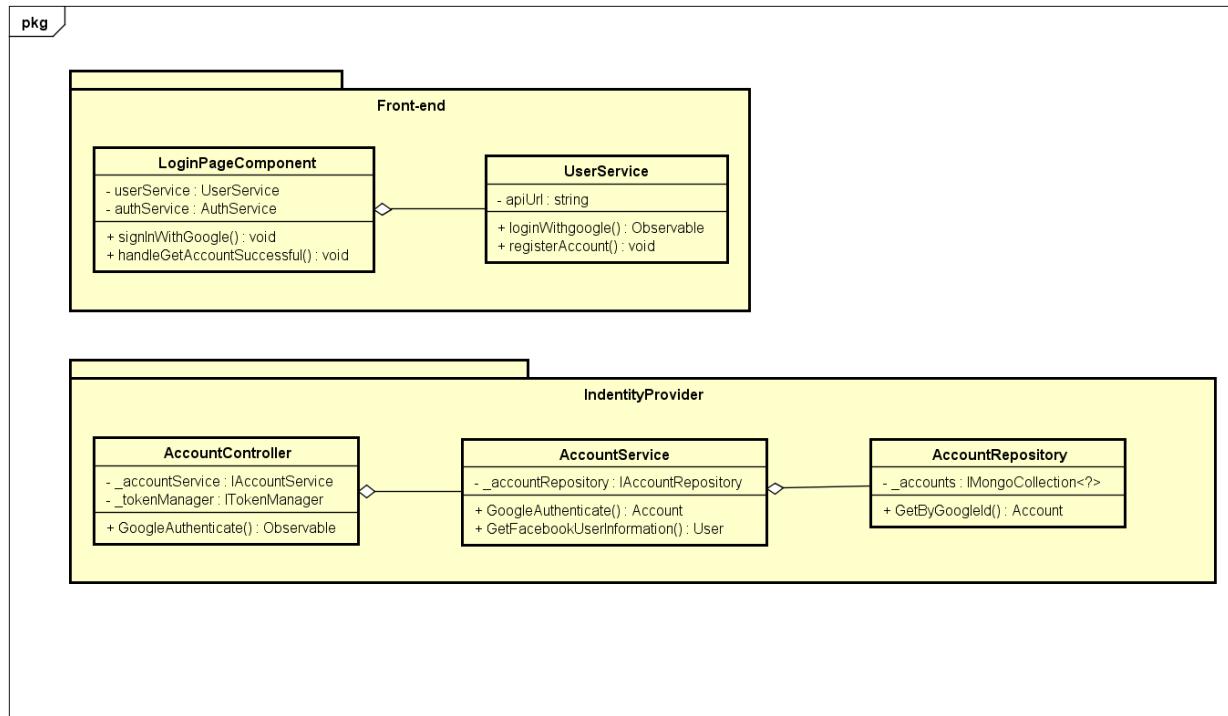


Figure 4-24: Google Sign-in Class Diagram

Class Specification

LoginPageComponent

LoginPageComponent			
Physical address	src\app\pages\login-page\login-page.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	userService	UserService	
2	authService	AuthService	
Operations			
No	Name	Return Type	Description
1	signInWithGoogle	Void	
2	handleGetAccountSuccessful	Void	

UserService

UserService			
Physical address	src\app\core\services\user-service\user.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	apiURL	String	

Operations			
No	Name	Return Type	Description
1	loginWithGoogle	Observable	

AccountController

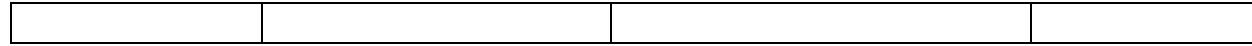
AccountController			
Physical address	\src\Services\IdentityProvider\IdentityProvider\Controllers\AccountController.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	_accountService	IAccountService	
2	_tokenManager	ITokenManager	
Operations			
No	Name	Return Type	Description
1	GoogleAuthenticate	Observable	

AccountService

AccountService			
Physical address	\src\Services\IdentityProvider\IdentityProvider\Services\AccountService.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	_accountRepository	IAccountRepository	
Operations			
No	Name	Return Type	Description
1	GoogleAuthenticate	Void	
2	GetGoogleUserInformation	Void	

AccountRepository

AccountRepository			
Physical address	\src\Services\IdentityProvider\IdentityProvider\Repositories\AccountRepository.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	_accounts	IMongoCollection<Account>	
Operations			
No	Name	Return Type	Description
1	GetByGoogleId	Void	



Sequence Diagram

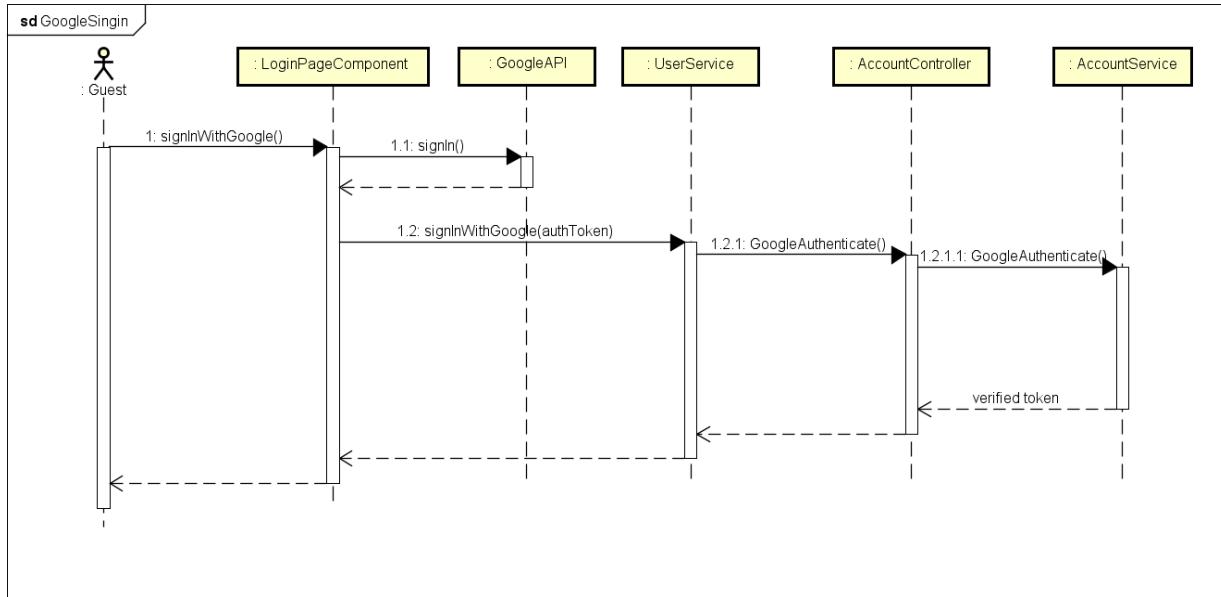


Figure 4-25: Google Sign-in Sequence Diagram

4.3.4.3 Facebook Sign-in

Screen Design

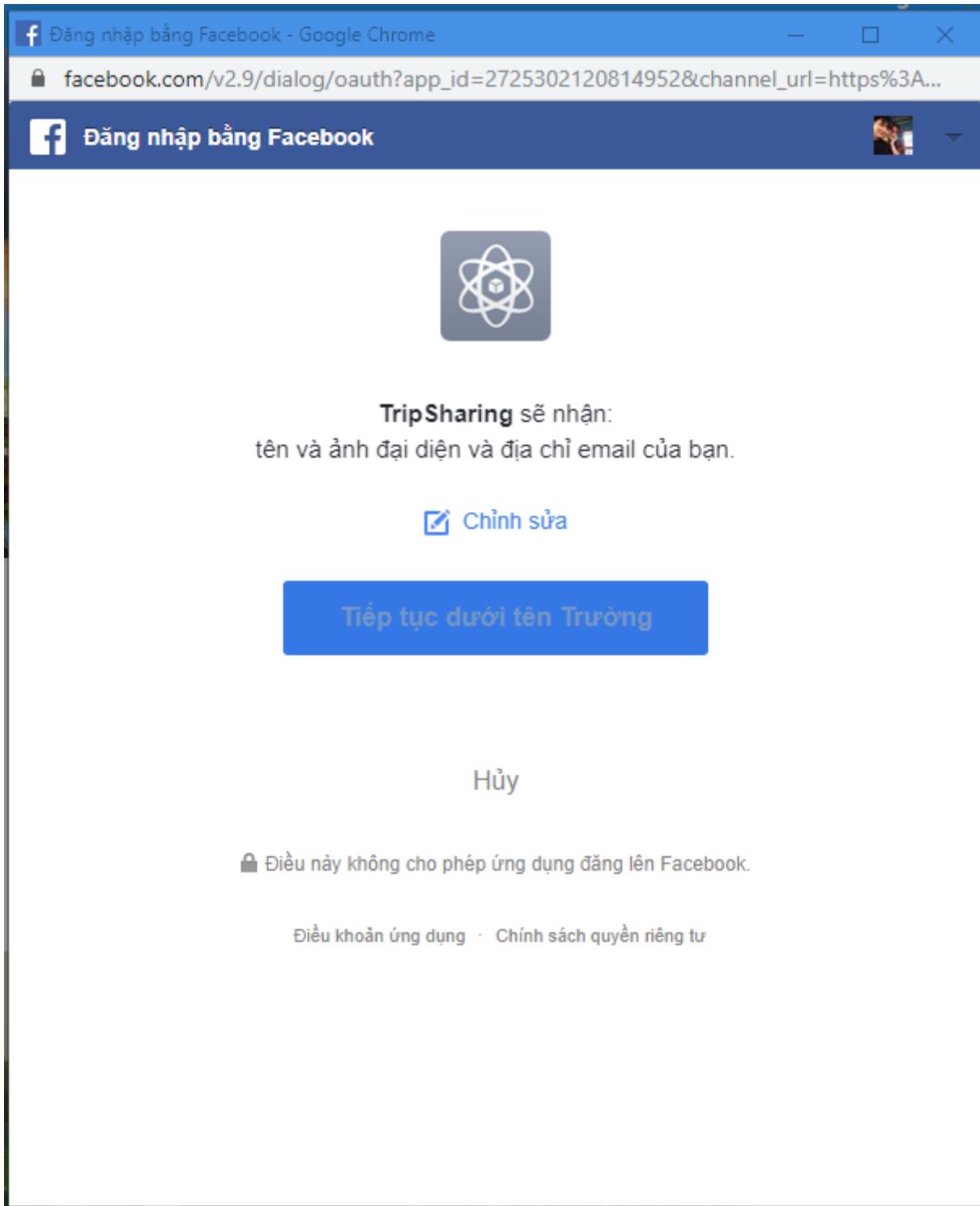


Figure 4-26: Facebook Sign-in Sequence Diagram

Class Diagram

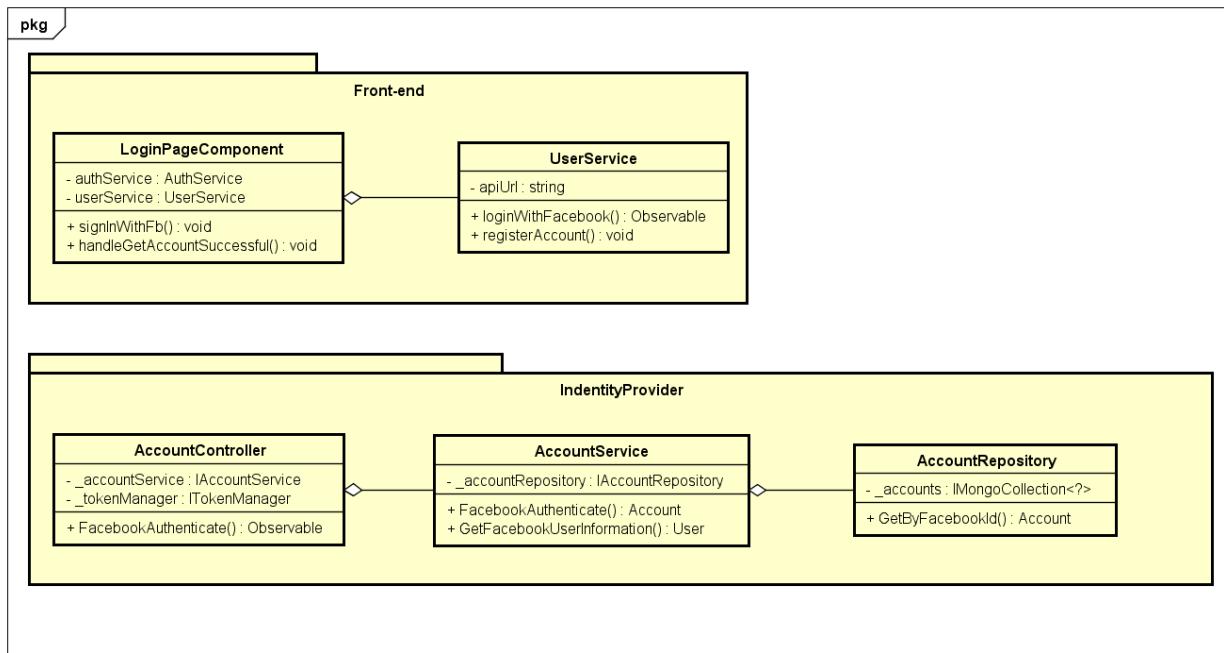


Figure 4-27: Facebook Sign-in Class Diagram

Class Specification

LoginPageComponent

LoginPageComponent			
Physical address	src\app\pages\login-page\login-page.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	userService	UserService	
2	authService	AuthService	
Operations			
No	Name	Return Type	Description
1	signInWithFb	Void	
2	handleGetAccountSuccessful	Void	

UserService

UserService			
Physical address	src\app\core\services\user-service\user.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	apiURL	String	
Operations			

No	Name	Return Type	Description
1	loginWithFacebook	Observable	

AccountController

AccountController			
Physical address	\src\Services\IdentityProvider\IdentityProvider\Controllers\AccountController.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	_accountService	IAccountService	
2	_tokenManager	ITokenManager	
Operations			
No	Name	Return Type	Description
1	FacebookAuthenticate	Observable	

AccountService

AccountService			
Physical address	\src\Services\IdentityProvider\IdentityProvider\Services\AccountService.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	_accountRepository	IAccountRepository	
Operations			
No	Name	Return Type	Description
1	FacebookAuthenticate	Void	
2	GetFacebookUserInformation	Void	

AccountRepository

AccountRepository			
Physical address	\src\Services\IdentityProvider\IdentityProvider\Repositories\AccountRepository.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	_accounts	IMongoCollection<Account>	
Operations			
No	Name	Return Type	Description
1	GetByFacebookId	Void	

Sequence Diagram

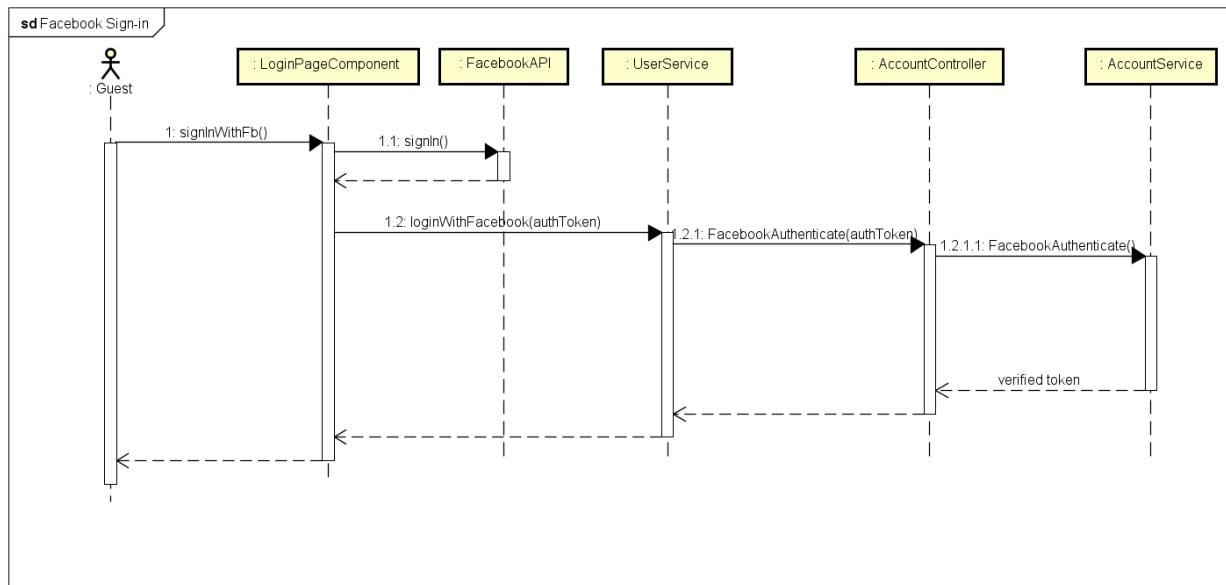


Figure 4-28: Facebook Sign-in Sequence Diagram

4.3.4.4 Normal Sign-in

Screen Design

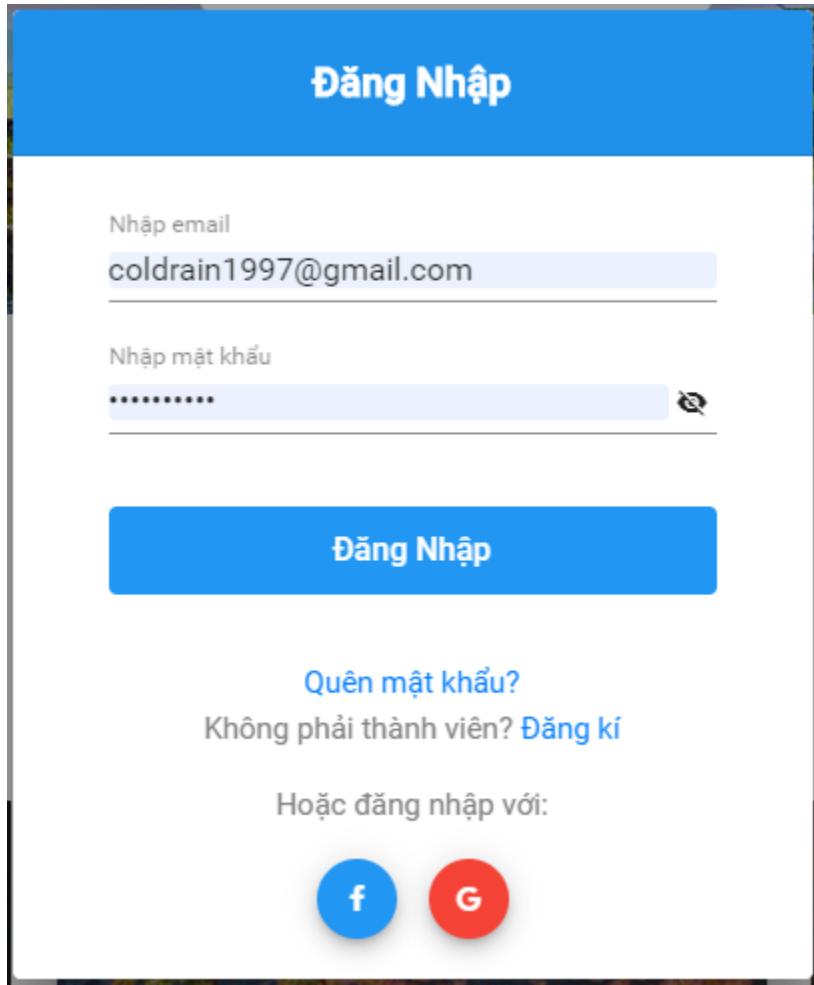


Figure 4-29: Normal Sign-in Screen Design

Class Diagram

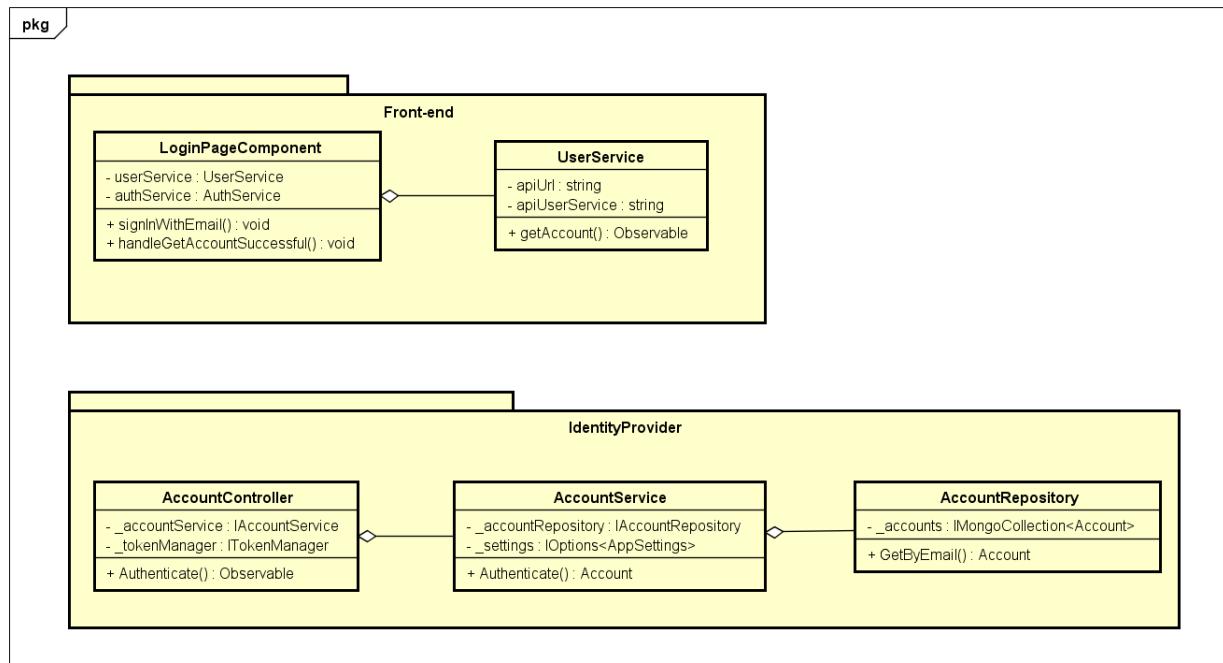


Figure 4-30: Normal Sign-in Class Diagram

Class Specification

LoginPageComponent

LoginPageComponent			
Physical address	src\app\pages\login-page\login-page.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	userService	UserService	
2	authService	AuthService	
Operations			
No	Name	Return Type	Description
1	signInWithEmail	Void	
2	handleGetAccountSuccessful	Void	

UserService

UserService			
Physical address	src\app\core\services\user-service\user.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	apiURL	String	
2	apiUserService	String	

Operations			
No	Name	Return Type	Description
1	getAccount	Observable	

AccountController

AccountController			
Physical address	src\Services\IdentityProvider\IdentityProvider\Controllers\AccountController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_accountService	IAccountService	
2	_tokenManager	ITokenManager	
Operations			
No	Name	Return Type	Description
1	Authenticate	Observable	

AccountService

AccountService			
Physical address	\src\Services\IdentityProvider\IdentityProvider\Services\AccountService.cs		
Base Class	IAccountService		
Attributes			
No	Name	Type	Description
1	_accountRepository	IAccountRepository	
Operations			
No	Name	Return Type	Description
1	Authenticate	Void	

AccountRepository

AccountRepository			
Physical address	\src\Services\IdentityProvider\IdentityProvider\Repositories\AccountRepository.cs		
Base Class	IAccountRepository		
Attributes			
No	Name	Type	Description
1	_accounts	IMongoCollection<?>	
Operations			
No	Name	Return Type	Description
1	GetByEmail	Account	

Sequence Diagram

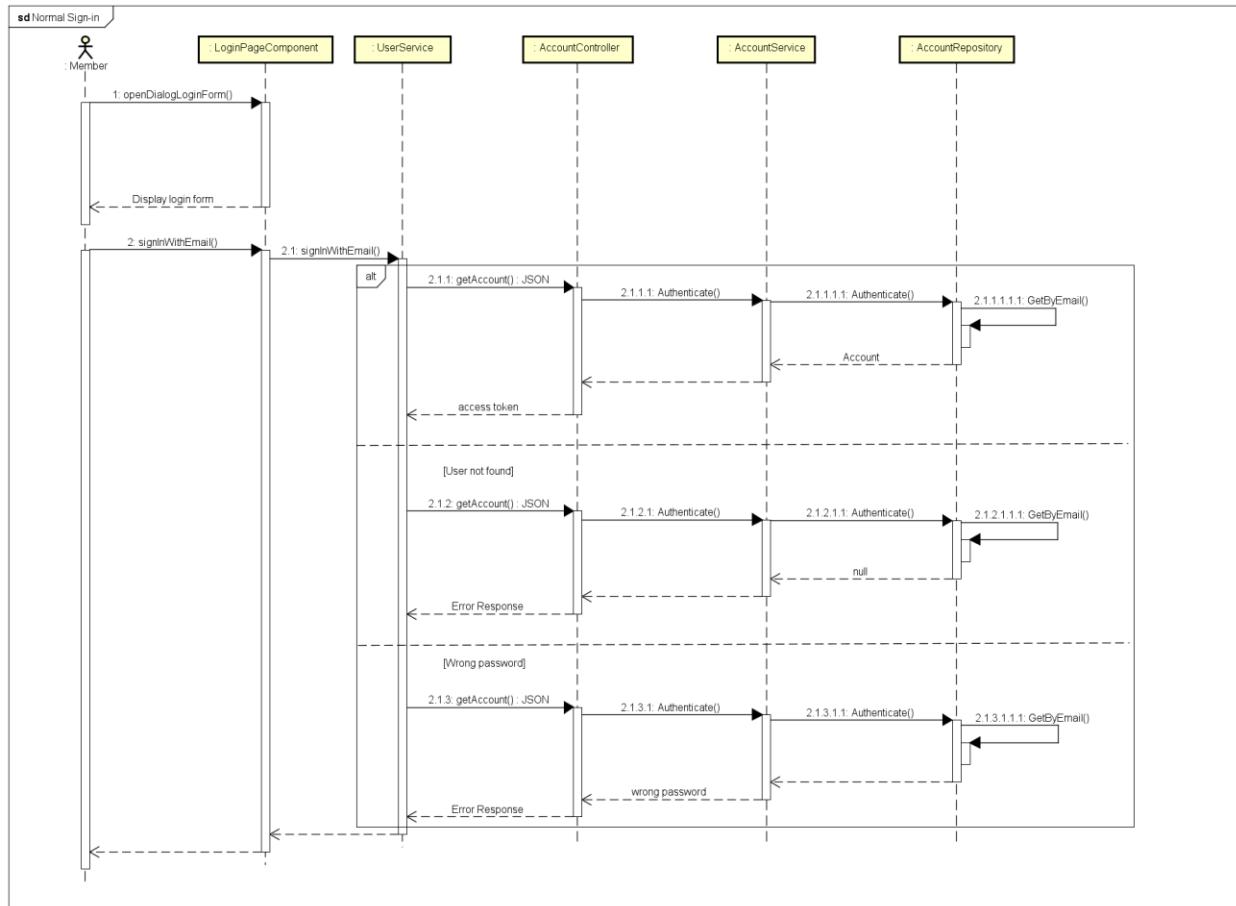


Figure 4-31: Normal Sign-in Sequence Diagram

4.3.4.5 Forgot Password

Screen Design

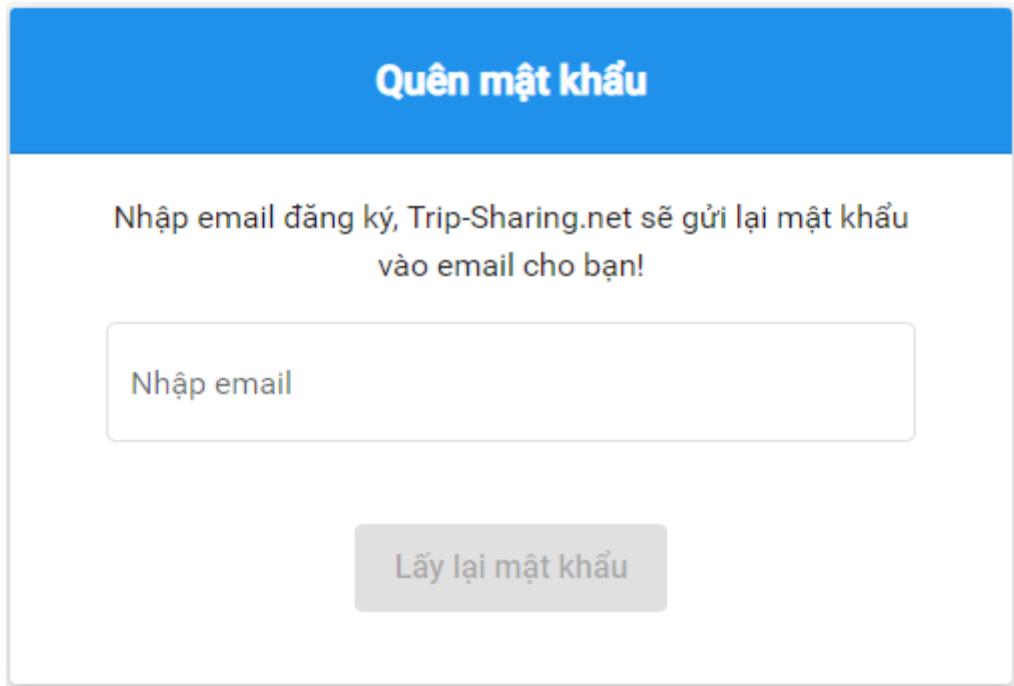


Figure 4-32: Forgot Password Screen Design

Class Diagram

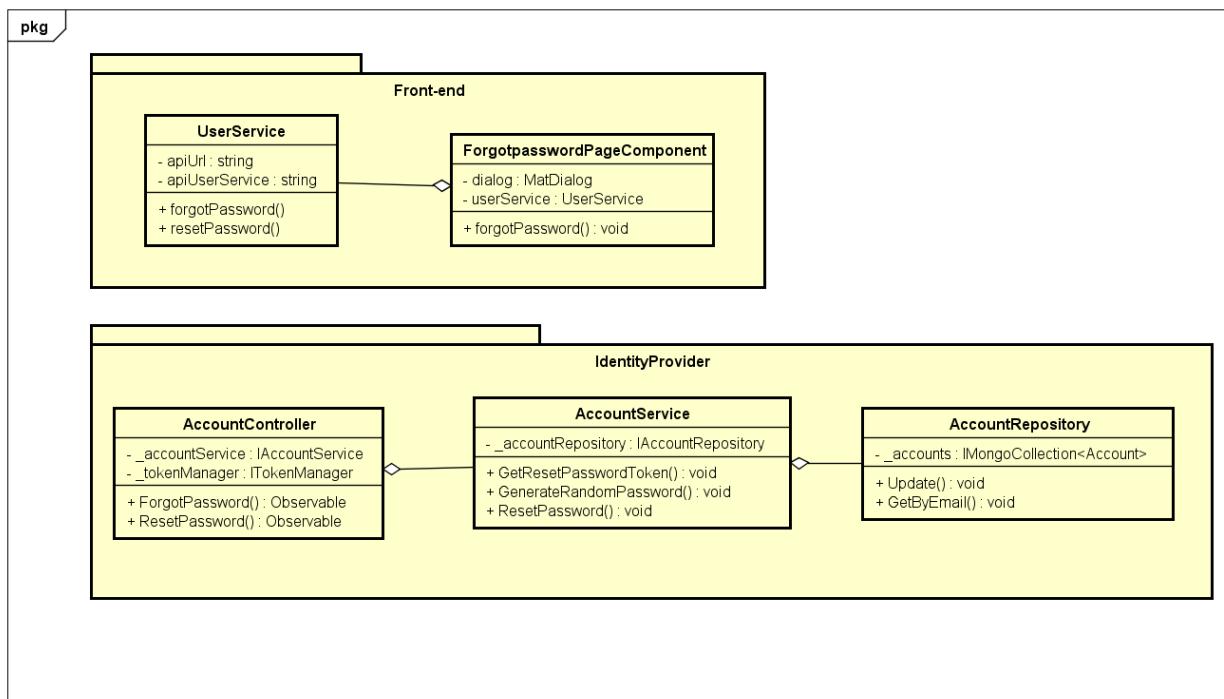


Figure 4-33: Forgot Password Class Diagram

Class Specification

ForgotpasswordPageComponent

ForgotpasswordPageComponent			
Physical address	src\app\pages\login-page\forgotpassword-page.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	dialog	MatDialog	
2	userService	UserService	
Operations			
No	Name	Return Type	Description
1	forgotPassword	Void	
2	resetPassword	Void	

UserService

UserService			
Physical address	src\app\core\services\user-service\user.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	apiUrl	String	
2	apiUserService	String	
Operations			
No	Name	Return Type	Description
1	forgotPassword	Observable	
2	resetPassword	Observable	

AccountController

AccountController			
Physical address	src\Services\IdentityProvider\IdentityProvider\Controllers\AccountController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_accountService	IAccountService	
2	_tokenManager	ITokenManager	
Operations			
No	Name	Return Type	Description
1	ForgotPassword	Observable	
2	ResetPassword	Observable	

AccountService

AccountService			
Physical address	\src\Services\IdentityProvider\IdentityProvider\Services\AccountService.cs		
Base Class	IAccountService		
Attributes			
No	Name	Type	Description
1	_accountRepository	IAccountRepository	
Operations			
No	Name	Return Type	Description
1	GetResetPasswordToken	Void	
2	GenerateRandomPassword	Void	
3	ResetPassword	Void	

AccountRepository

AccountRepository			
Physical address	\src\Services\IdentityProvider\IdentityProvider\Repositories\AccountRepository.cs		
Base Class	IAccountRepository		
Attributes			
No	Name	Type	Description
1	_accounts	IMongoCollection<?>	
Operations			
No	Name	Return Type	Description
1	GetByEmail	Account	
2	Update	Void	

Sequence Diagram

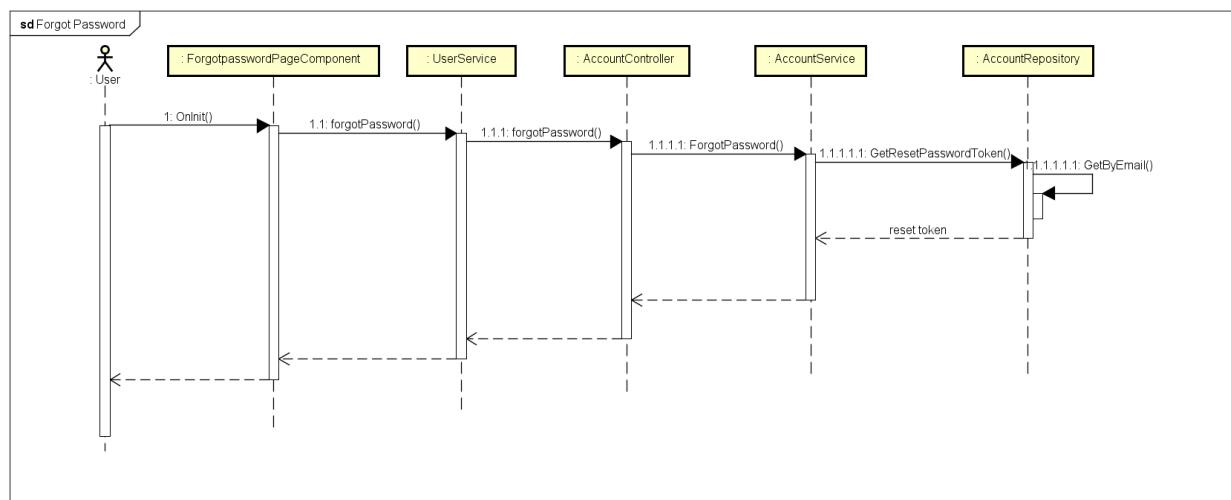


Figure 4-34: Forgot Password Sequence Diagram

4.3.4.6 Sign-out

Screen Design



Figure 4-35: Sign-out Screen Design

Class Diagram

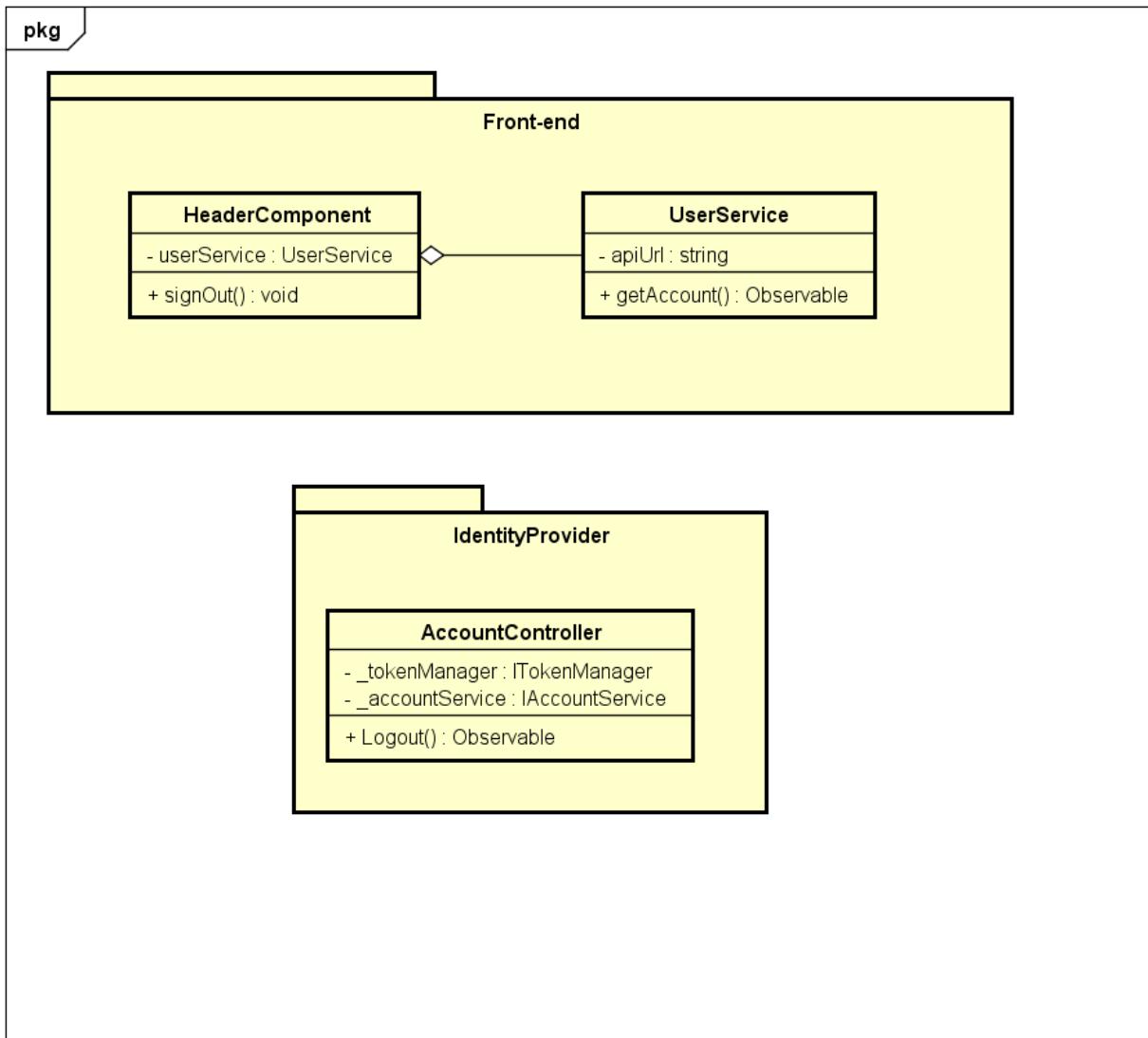


Figure 4-36: Sign-out Class Diagram

Class Specification

HeaderComponent

HeaderComponent			
Physical address	src\app\core\components\header\ header.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	userService	UserService	
Operations			
No	Name	Return Type	Description
1	signOut	Void	

UserService

UserService			
Physical address	src\app\core\services\user-service\user.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	apiUrl	String	
Operations			
No	Name	Return Type	Description
1	getAccount	Observable	

AccountController

AccountController			
Physical address	src\Services\IdentityProvider\IdentityProvider\Controllers\AccountController.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	_accountService	IAccountService	
2	_tokenManager	ITokenManager	
Operations			
No	Name	Return Type	Description
1	Logout	Observable	

Sequence Diagram

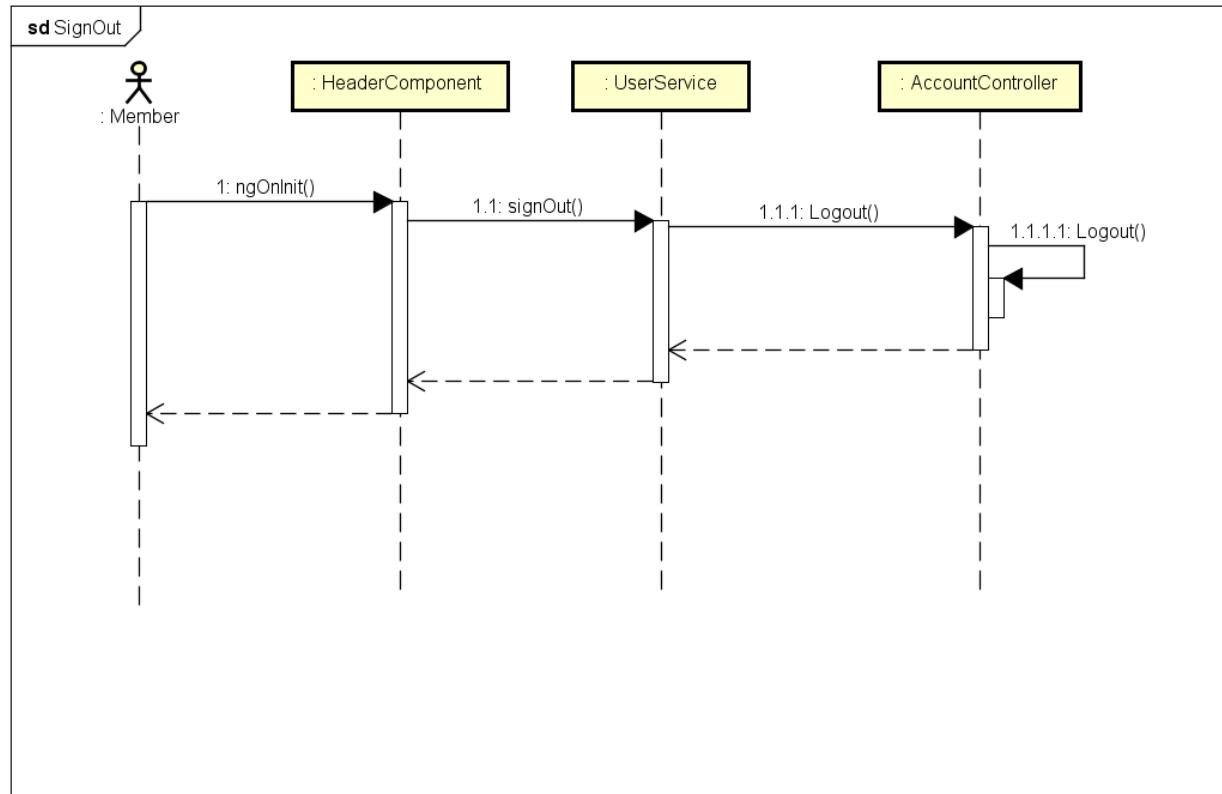


Figure 4-37: Sign-out Sequence Diagram

4.3.4.7 Change Password

Screen Design

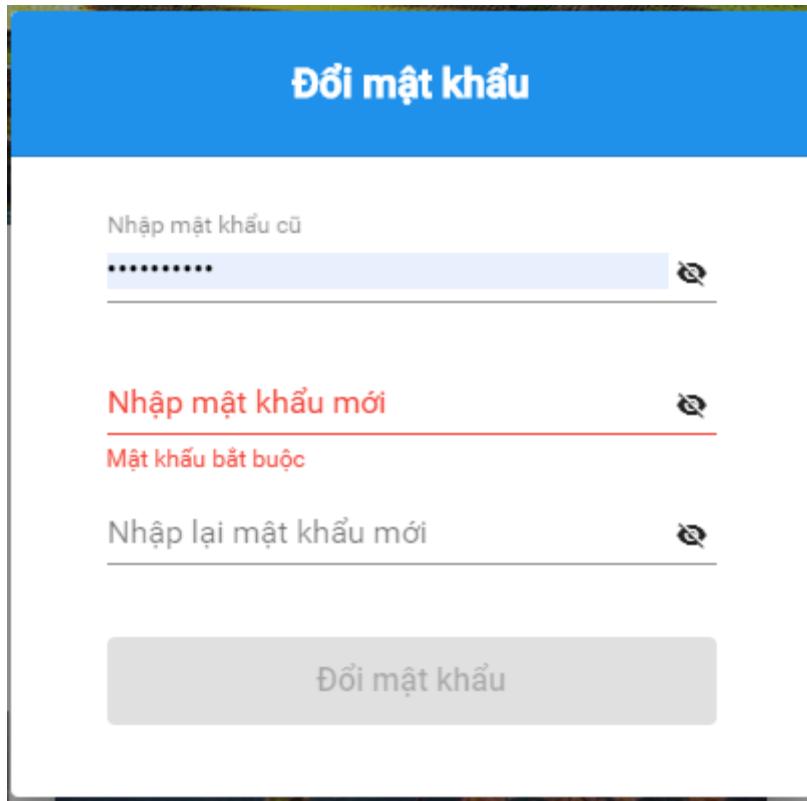


Figure 4-38: Change Password Screen Design

Class Diagram

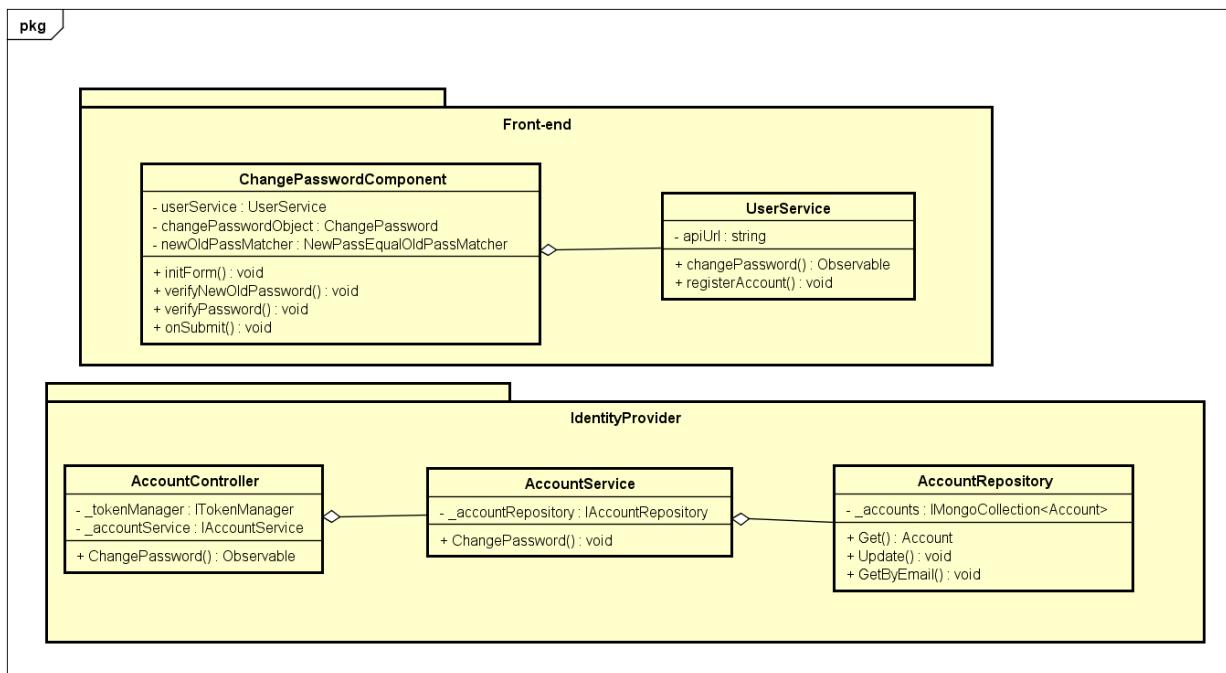


Figure 4-39: Change Password Class Diagram

Class Specification

ChangePasswordComponent

ChangePasswordComponent			
Physical address	\src\app\shared\components\change-password\change-password.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	userService	UserService	
2	changePasswordObject	ChangePassword	
3	newOldPassMatcher	NewPassEqualOldPassMatcher	
Operations			
No	Name	Return Type	Description
1	initForm	Void	
2	verifyNewOldPassword	Void	
3	verifyPassword	Void	
4	onSubmit	Void	

UserService

UserService			
Physical address	src\app\core\services\user-service\user.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	apiUrl	String	
2	apiUserService	String	
Operations			
No	Name	Return Type	Description
1	changePassword	Observable	
2	registerAccount	Observable	

AccountController

AccountController			
Physical address	src\Services\IdentityProvider\IdentityProvider\Controllers\AccountController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_accountService	IAccountService	
2	_tokenManager	ITokenManager	

Operations			
No	Name	Return Type	Description
1	ChangePassword	Observable	

AccountService

AccountService			
Physical address	\src\Services\IdentityProvider\IdentityProvider\Services\AccountService.cs		
Base Class	IAccountService		
Attributes			
No	Name	Type	Description
1	_accountRepository	IAccountRepository	
Operations			
No	Name	Return Type	Description
1	ChangePassword	Void	

AccountRepository

AccountRepository			
Physical address	\src\Services\IdentityProvider\IdentityProvider\Repositories\AccountRepository.cs		
Base Class	IAccountRepository		
Attributes			
No	Name	Type	Description
1	_accounts	IMongoCollection<?>	
Operations			
No	Name	Return Type	Description
1	GetByEmail	Account	
2	Update	Void	
3	Get	Account	

Sequence Diagram

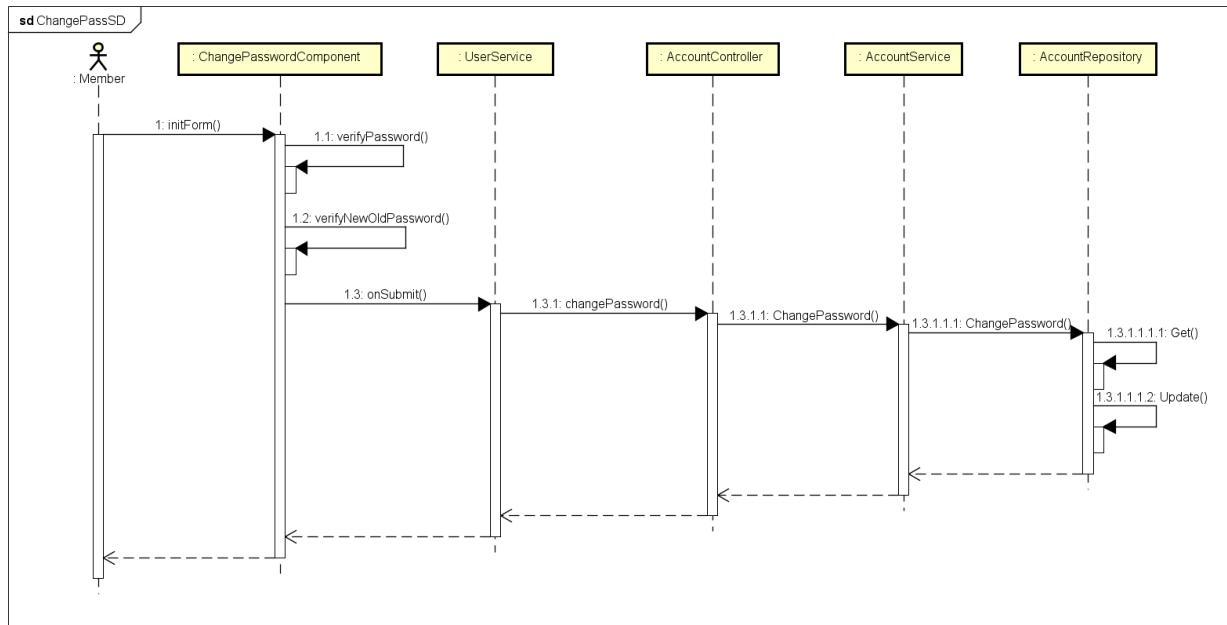


Figure 4-40: Change Password Sequence Diagram

4.3.4.8 Update Profile

Screen Design

Thông tin

Cập nhật thông tin người dùng

Username *

coldraindocs

Duy nhất cho mỗi tài khoản

Họ *

Le

Tên hiển thị *

Le Xuan Truong

Ngày Sinh

1/24/1997

Địa chỉ

Hà Nội

Nam Nữ

Tiếp tục

Figure 4-41: Update Profile Screen Design

Class Diagram

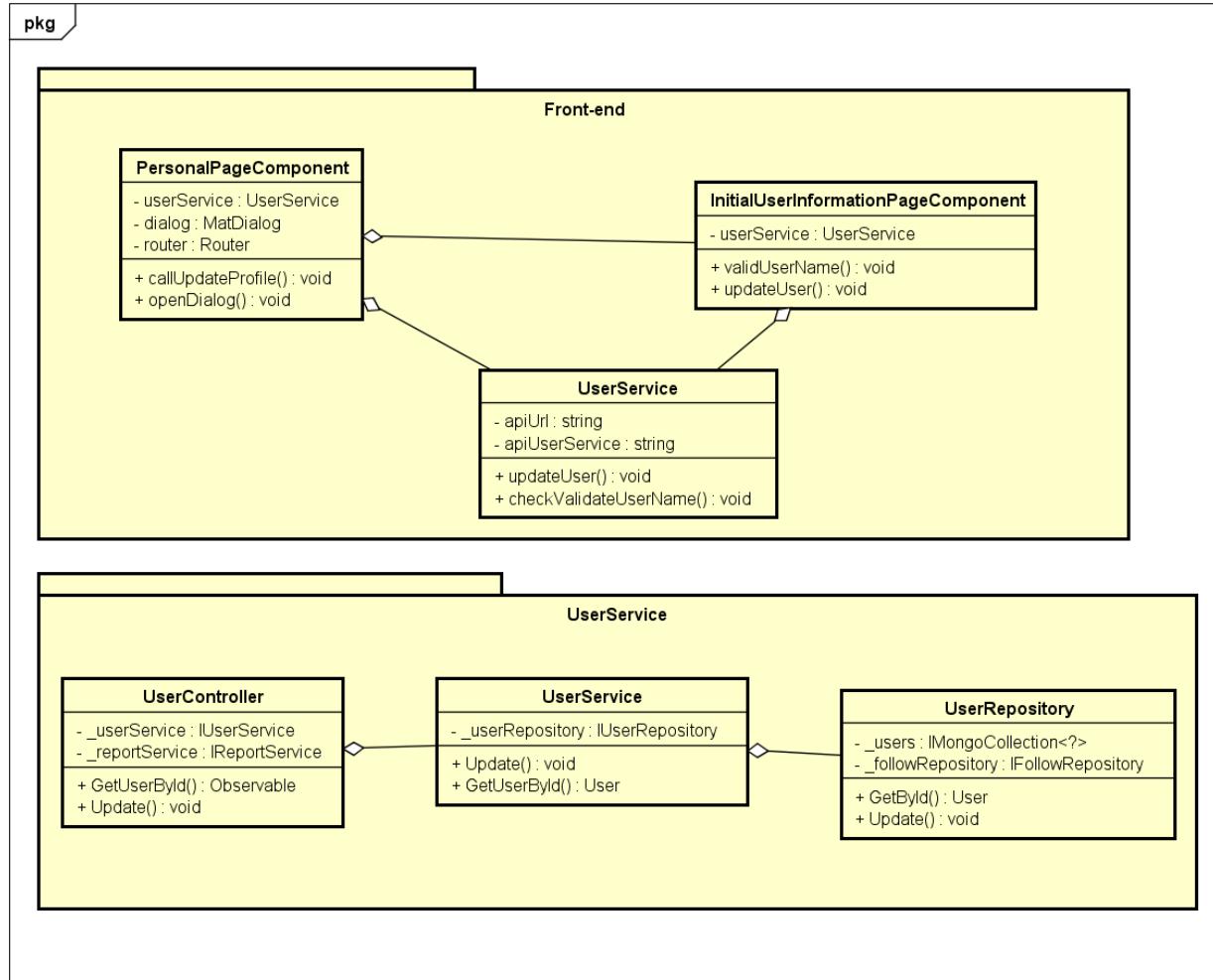


Figure 4-42: Update Profile Class Diagram

Class Specification

PersonalPageComponent

PersonalPageComponent			
Physical address	src\app\pages\personal-page\ personal-page.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	userService	UserService	
2	dialog	MatDialog	
3	router	Router	
Operations			
No	Name	Return Type	Description
1	callUpdateProfile	Void	
2	openDialog	Void	

InitialUserInformationPageComponent

InitialUserInformationPageComponent			
Physical address	src\app\pages\initial-user-information-page\initial-user-information-page.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	userService	UserService	
Operations			
No	Name	Return Type	Description
1	validUserName	Void	
2	updateUser	Void	

UserService

UserService			
Physical address	src\app\core\services\user-service\user.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	apiUrl	String	
Operations			
No	Name	Return Type	Description
1	checkValidateUserName	Observable	
2	updateUser	Observable	

UserController

UserController			
Physical address	\src\Services\UserService\UserService\Controllers\UserController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_userService	IUserService	
2	_reportService	IReportService	
Operations			
No	Name	Return Type	Description
1	GetUserById	Observable	
2	Update	Observable	

UserService (back-end)

UserService			
Physical address	\src\Services\UserService\UserService\Services\UserService.cs		
Base Class	IUserService		
Attributes			
No	Name	Type	Description
1	_userRepository	IUserRepository	
Operations			
No	Name	Return Type	Description
1	GetUserById	User	
2	Update	Void	

UserRepository

UserRepository			
Physical address	\src\Services\UserService\UserService\Repositories\UserRepository.cs		
Base Class	IUserRepository		
Attributes			
No	Name	Type	Description
1	_users	IMongoCollection<?>	
2	_followRepository	IFollowRepository	
Operations			
No	Name	Return Type	Description
1	GetById	User	
2	Update	Void	

Sequence Diagram

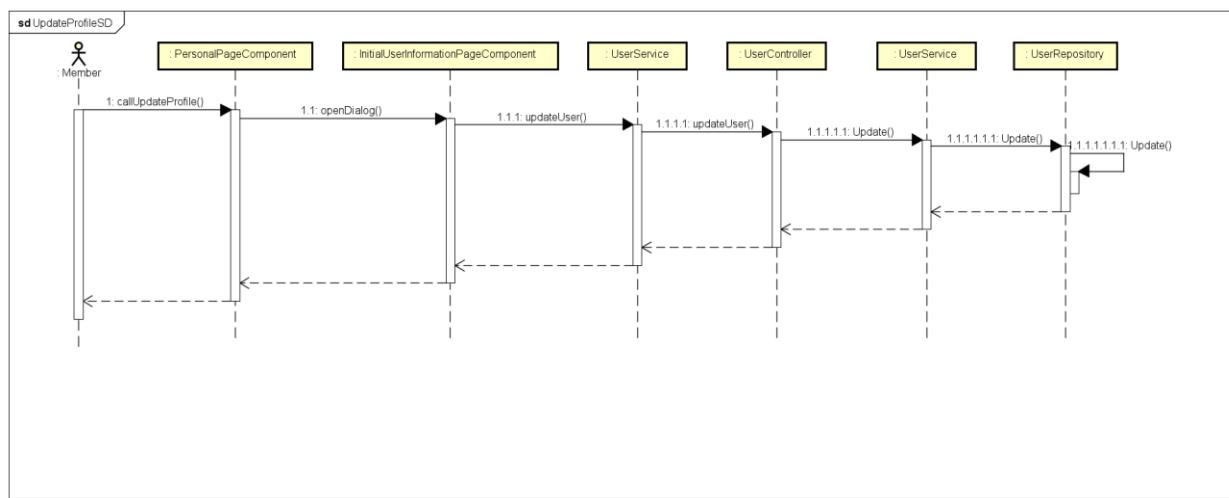


Figure 4-43: Update Profile Sequence Diagram

4.3.4.9 Choose Interested Topic

Screen Design



Figure 4-44: Choose Interested Topic

Class Diagram

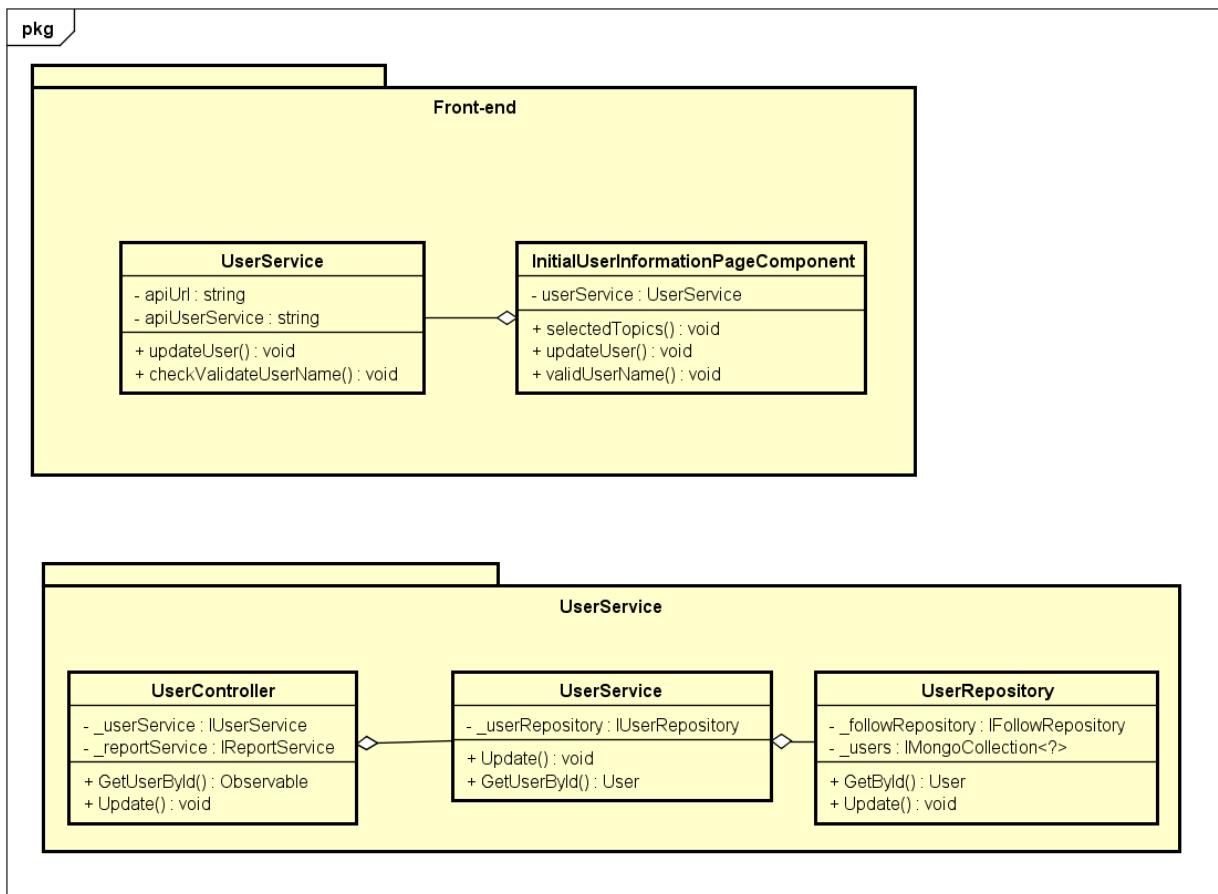


Figure 4-45: Choose Interested Topic Class Diagram

Class Specification

InitialUserInformationPageComponent

InitialUserInformationPageComponent			
Physical address	src\app\pages\initial-user-information-page\initial-user-information-page.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	userService	UserService	
Operations			
No	Name	Return Type	Description
1	validUserName	Void	
2	updateUser	Void	
3	selectedTopics	Void	

UserService

UserService			
Physical address	src\app\core\services\user-service\user.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	apiUrl	String	
Operations			
No	Name	Return Type	Description
1	checkValidateUserName	Observable	
2	updateUser	Observable	

UserController

UserController			
Physical address	\src\Services\UserService\UserService\Controllers\UserController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_userService	IUserService	
2	_reportService	IReportService	
Operations			
No	Name	Return Type	Description
1	GetUserById	Observable	

2	Update	Observable	
---	--------	------------	--

UserService (back-end)

UserService			
Physical address			\src\Services\UserService\UserService\Services\UserService.cs
Base Class			IUserService
Attributes			
No	Name	Type	Description
1	_userRepository	IUserRepository	
Operations			
No	Name	Return Type	Description
1	GetUserById	User	
2	Update	Void	

UserRepository

UserRepository			
Physical address			\src\Services\UserService\UserService\Repositories\UserRepository.cs
Base Class			IUserRepository
Attributes			
No	Name	Type	Description
1	_users	IMongoCollection<?>	
2	_followRepository	IFollowRepository	
Operations			
No	Name	Return Type	Description
1	GetById	User	
2	Update	Void	

Sequence Diagram

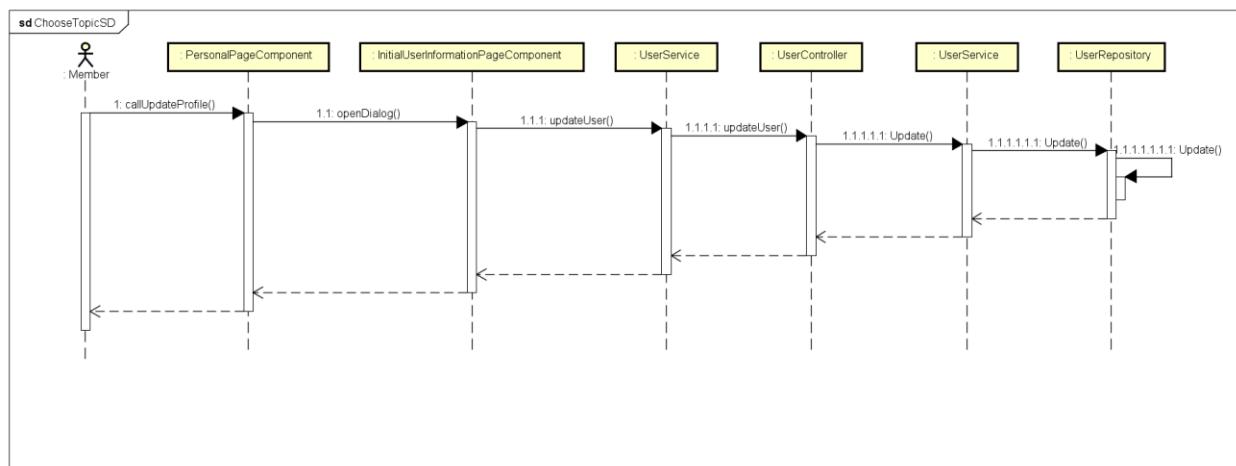


Figure 4-46: Choose Interested Topic Sequence Diagram

4.3.4.10 Create A Post

Screen Design

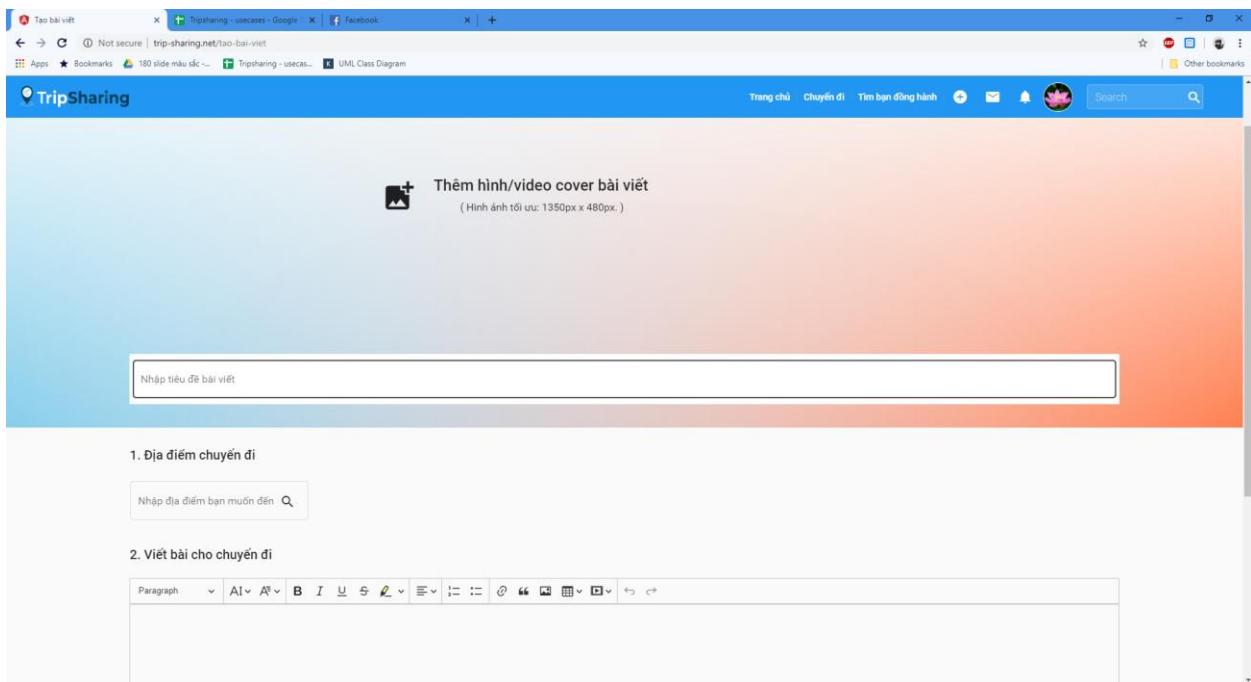


Figure 4-47: Create A Post Screen Design

Class Diagram

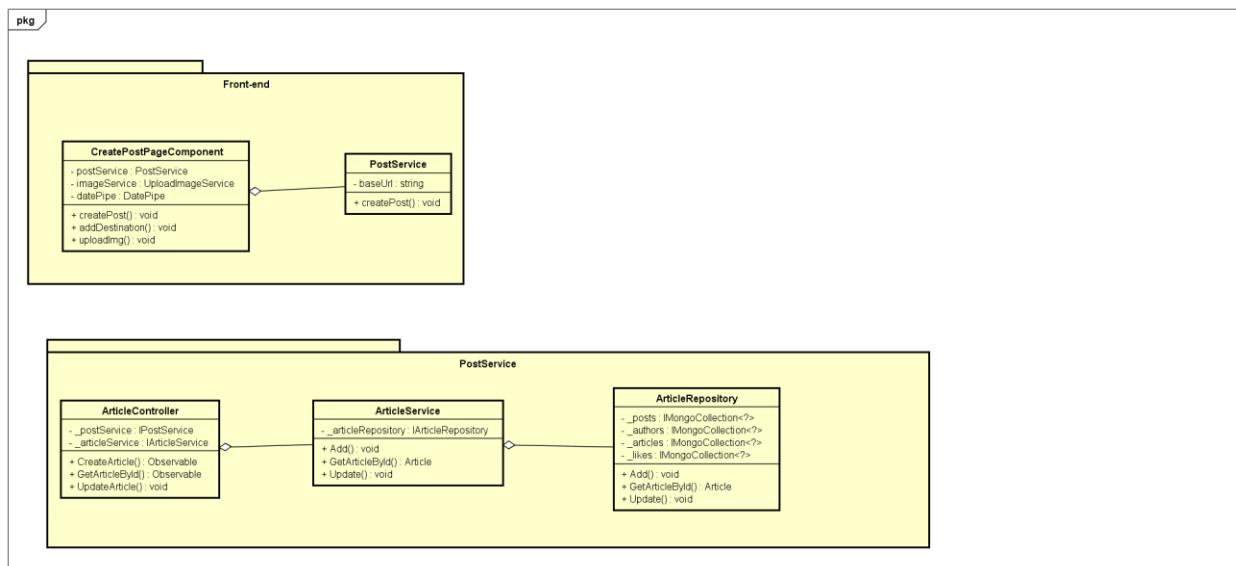


Figure 4-48: Create A Post Class Diagram

Class Specification

CreatePostPageComponent

CreatePostPageComponent			
Physical address	src\app\pages\create-post-page\ create-post-page.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	postService	PostService	
2	imageService	UploadImageService	
3	datePipe	DatePipe	
Operations			
No	Name	Return Type	Description
1	createPost	Void	
2	addDestination	Void	User must choose a destination
3	uploadImg	Void	Upload image

PostService

PostService			
Physical address	src\app\core\services\post-service\post.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	baseUrl	String	
Operations			
No	Name	Return Type	Description
1	createPost	Void	

ArticleController

ArticleController			
Physical address	src\Services\PostService\PostService\Controllers\ArticleController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_articleService	IArticleService	
2	_postService	IPostService	
Operations			
No	Name	Return Type	Description

1	GetArticleById	Observable	
2	UpdateArticle	Void	
3	CreateArticle	Void	

ArticleService

ArticleService			
Physical address	src\Services\PostService\PostService\Services\ArticleService.cs		
Base Class	IArticleService		
Attributes			
No	Name	Type	Description
1	_articleService	IArticleService	
2	_postService	IPostService	
Operations			
No	Name	Return Type	Description
1	GetArticleById	Article	
2	Update	Void	
3	Add	Void	

ArticleRepository

ArticleRepository			
Physical address	src\Services\PostService\PostService\Repositories\ArticleRepository.cs		
Base Class	IArticleRepository		
Attributes			
No	Name	Type	Description
1	_articles	IMongoCollection<?>	
2	_posts	IMongoCollection<?>	
3	_authors	IMongoCollection<?>	
4	_likes	IMongoCollection<?>	
Operations			
No	Name	Return Type	Description
1	GetArticleById	Article	
2	Update	Void	
3	Add	Void	

Sequence Diagram

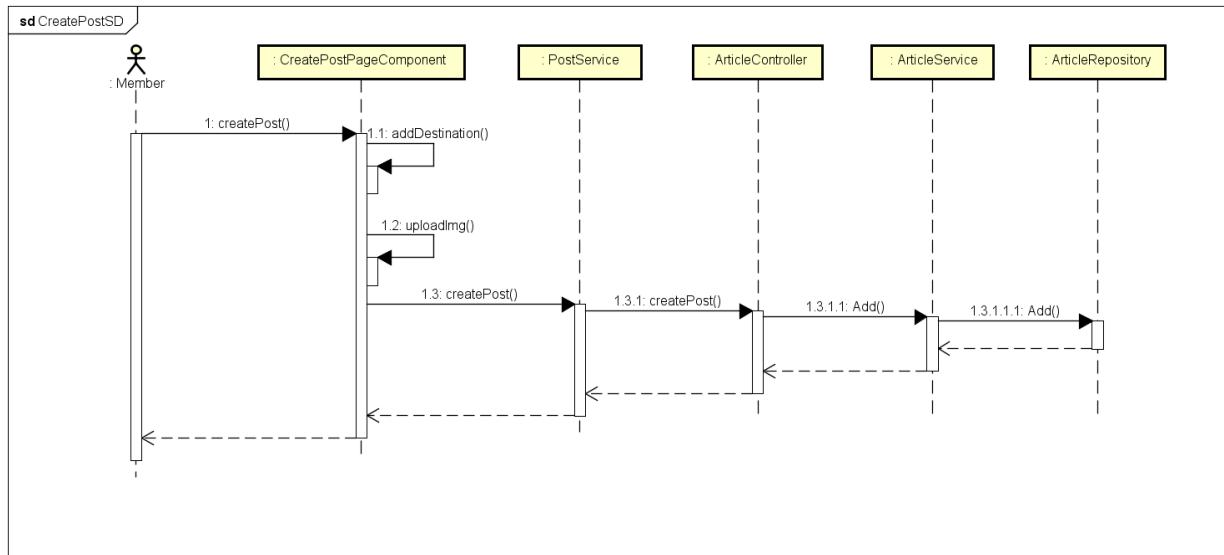


Figure 4-49: Create A Post Sequence Diagram

4.3.4.11 Edit A Post

Screen Design

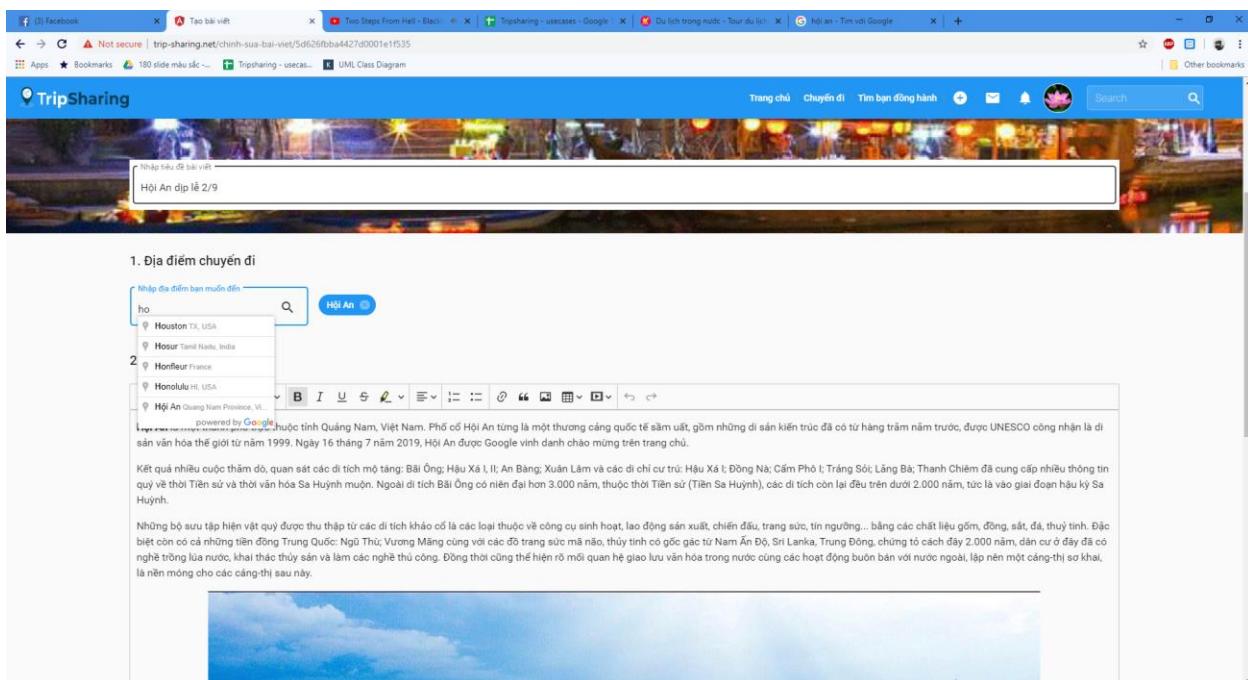


Figure 4-50: Edit A Post Sequence Diagram

Class Diagram

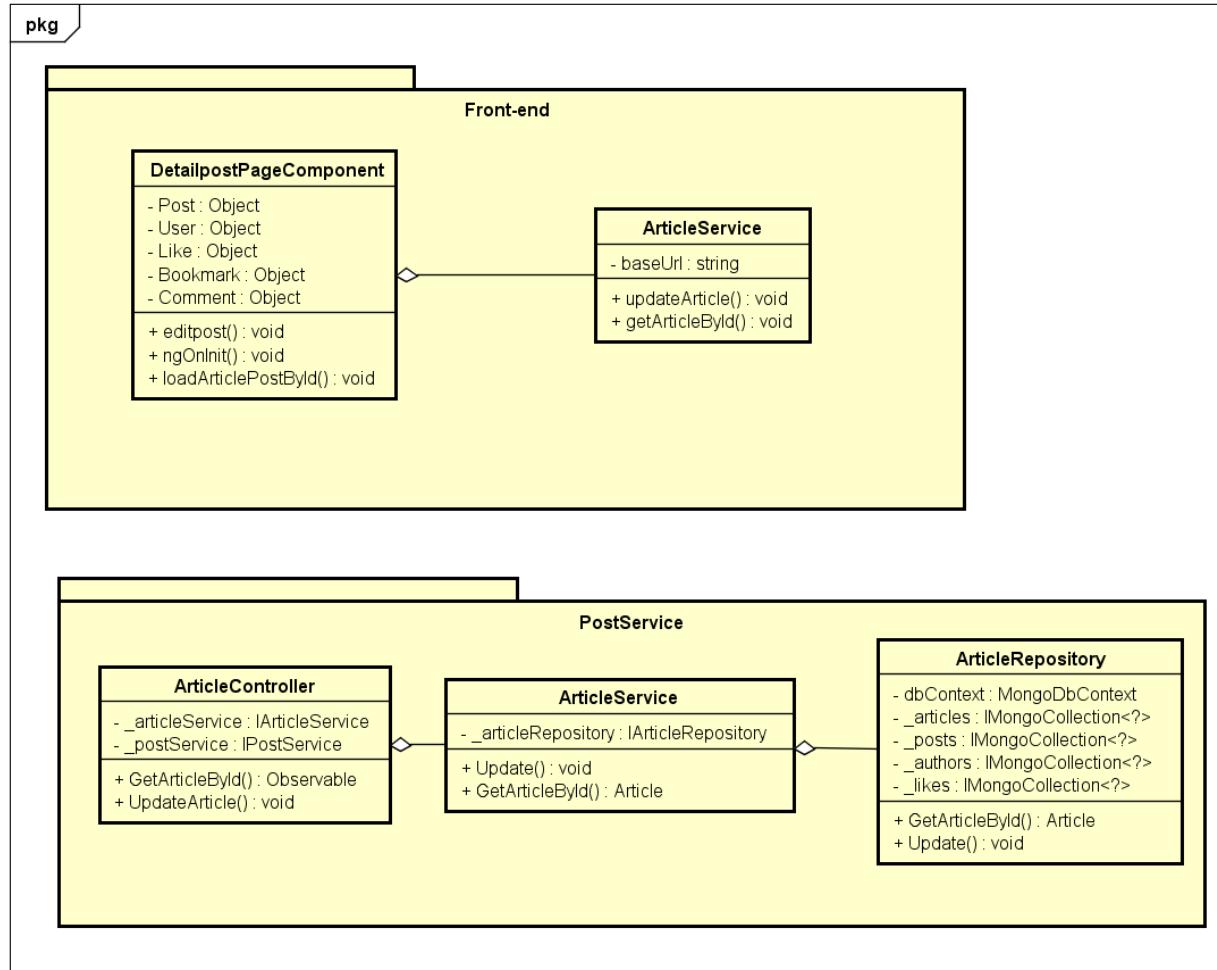


Figure 4-51: Edit A Post Class Diagram

Class Specification

DetailpostPageComponent

DetailpostPageComponent			
Physical address	src\app\pages\detailpost-page\detailpost-page.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	Post	Object	Instance of Post object
2	User	Object	Instance of User object
3	Like	Object	Instance of Like object
4	Bookmark	Object	Instance of Bookmark object
5	Comment	Object	Instance of Comment object
Operations			

No	Name	Return Type	Description
1	editpost	Void	
2	ngOnInit	Void	
3	loadArticlePostById	Void	

PostService

PostService			
Physical address	src\app\core\services\post-service\post.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	baseUrl	String	
Operations			
No	Name	Return Type	Description
1	getArticleById	Article	
2	updateArticle	Void	

ArticleController

ArticleController			
Physical address	src\Services\PostService\PostService\Controllers\ArticleController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_articleService	IArticleService	
2	_postService	IPostService	
Operations			
No	Name	Return Type	Description
1	GetArticleById	Observable	
2	UpdateArticle	Void	

ArticleService

ArticleService			
Physical address	src\Services\PostService\PostService\Articles\ArticleService.cs		
Base Class	IArticleService		
Attributes			
No	Name	Type	Description
1	_articleService	IArticleService	
2	_postService	IPostService	
Operations			

No	Name	Return Type	Description
1	GetArticleById	Article	
2	Update	Void	

ArticleRepository

ArticleRepository			
Physical address	src\Services\PostService\PostService\Repositories\ArticleRepository.cs		
Base Class	IArticleRepository		
Attributes			
No	Name	Type	Description
1	dbContext	MongoDbContext	
2	_articles	IMongoCollection<?>	
3	_posts	IMongoCollection<?>	
4	_authors	IMongoCollection<?>	
5	_likes	IMongoCollection<?>	
Operations			
No	Name	Return Type	Description
1	GetArticleById	Article	
2	Update	Void	

Sequence Diagram

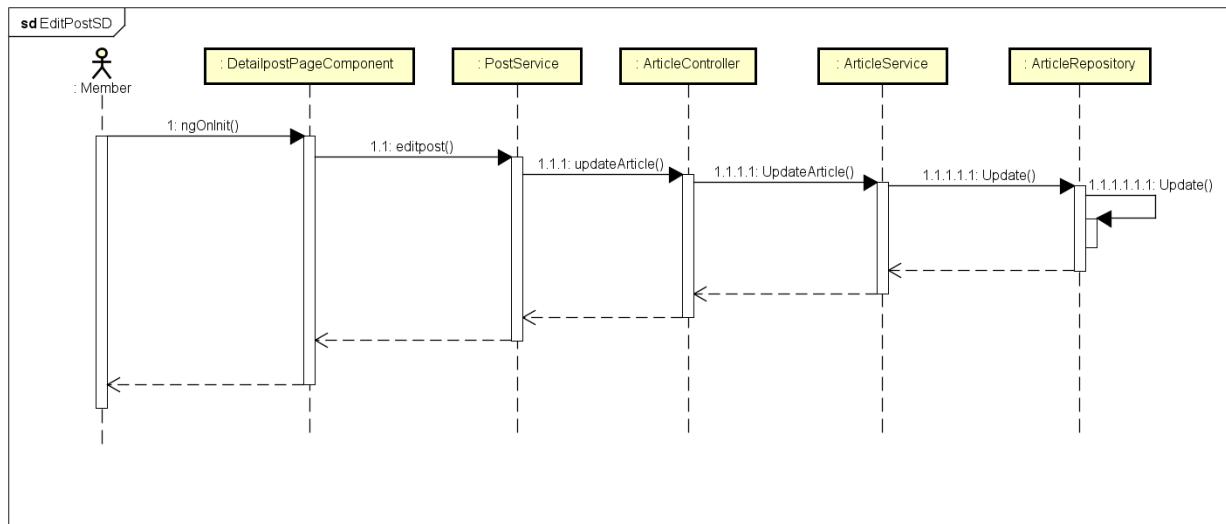


Figure 4-52: Edit A Post Sequence Diagram

4.3.4.12 Delete A Post

Screen Design

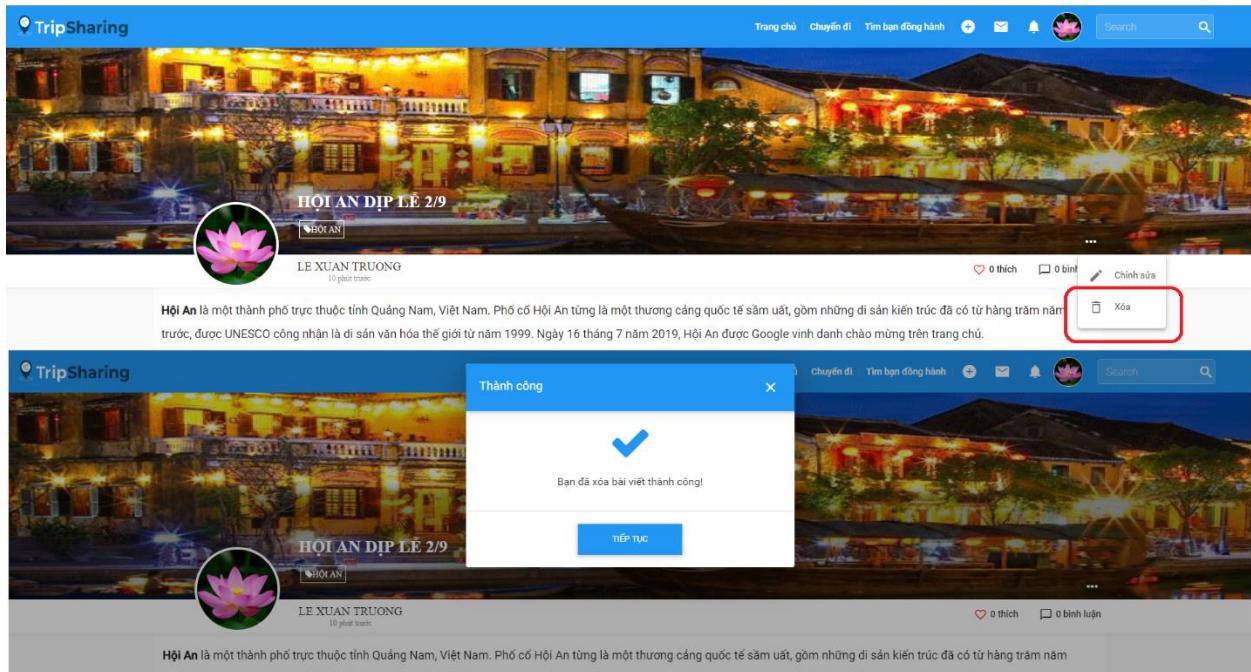


Figure 4-53: Delete A Post Screen Design

Class Diagram

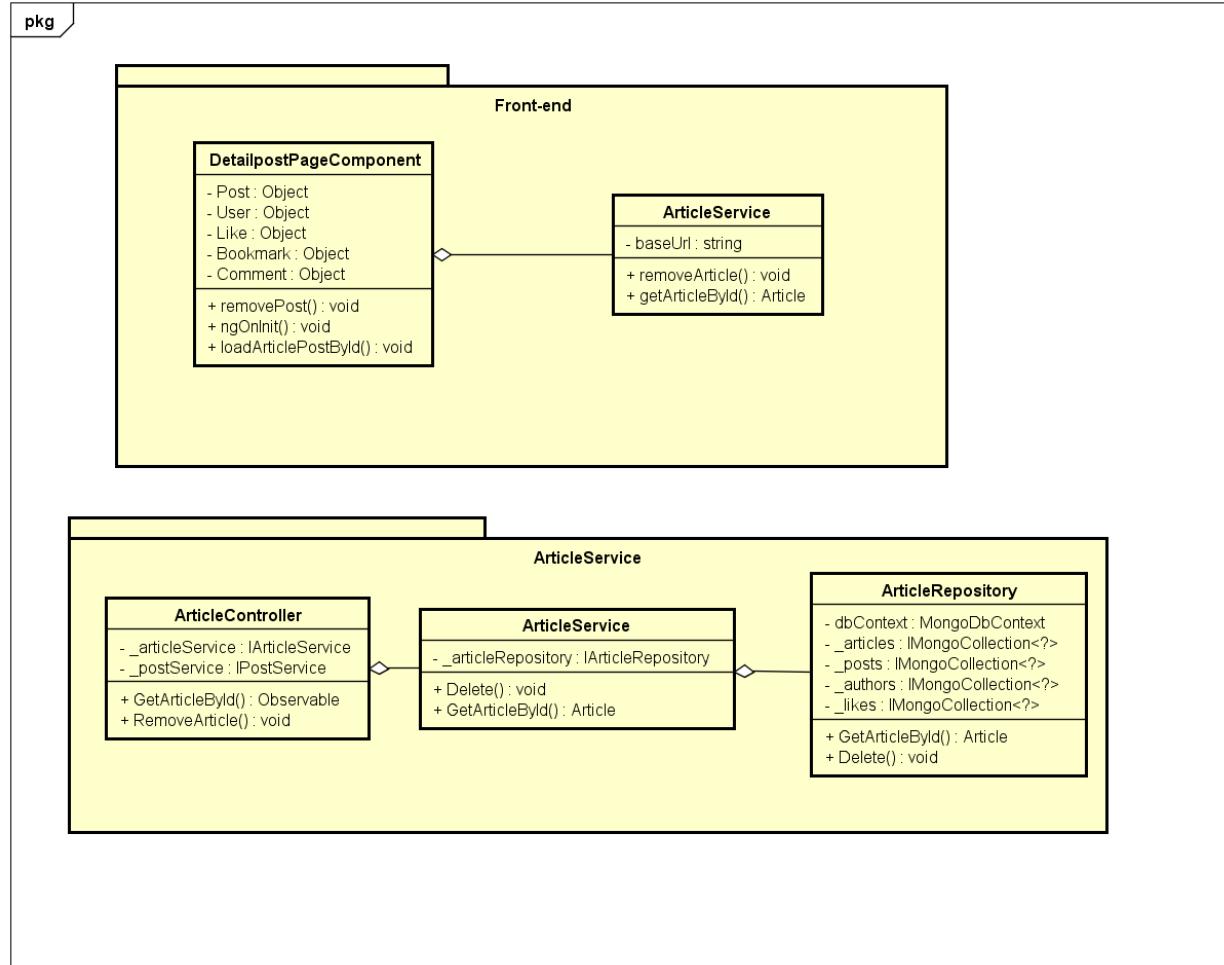


Figure 4-53: Delete A Post Class Diagram

Class Specification

DetailpostPageComponent

DetailpostPageComponent			
Physical address	src\app\pages\detailpost-page\detailpost-page.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	Post	Object	Instance of Post object
2	User	Object	Instance of User object
3	Like	Object	Instance of Like object
4	Bookmark	Object	Instance of Bookmark object
5	Comment	Object	Instance of Comment object
Operations			
No	Name	Return Type	Description

1	removePost	Void	
2	ngOnInit	Void	
3	loadArticlePostById	Void	

PostService

PostService			
Physical address	src\app\core\services\post-service\post.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	baseUrl	String	
Operations			
No	Name	Return Type	Description
1	getArticleById	Article	
2	removeArticle	Void	

ArticleController

ArticleController			
Physical address	src\Services\PostService\PostService\Controllers\ArticleController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_articleService	IArticleService	
2	_postService	IPostService	
Operations			
No	Name	Return Type	Description
1	GetArticleById	Observable	
2	RemoveArticle	Void	

ArticleService

ArticleService			
Physical address	src\Services\PostService\PostService\Services\ArticleService.cs		
Base Class	IArticleService		
Attributes			
No	Name	Type	Description
1	_articleService	IArticleService	
2	_postService	IPostService	
Operations			
No	Name	Return Type	Description

1	GetArticleById	Article	
2	Delete	Void	

ArticleRepository

ArticleRepository			
Physical address	src\Services\PostService\PostService\Repositories\ArticleRepository.cs		
Base Class	IArticleRepository		
Attributes	No	Name	Type
	1	dbContext	MongoDbContext
	2	_articles	IMongoCollection<?>
	3	_posts	IMongoCollection<?>
	4	_authors	IMongoCollection<?>
	5	_likes	IMongoCollection<?>
Operations	No	Name	Return Type
	1	GetArticleById	Article
	2	Delete	Void

Sequence Diagram

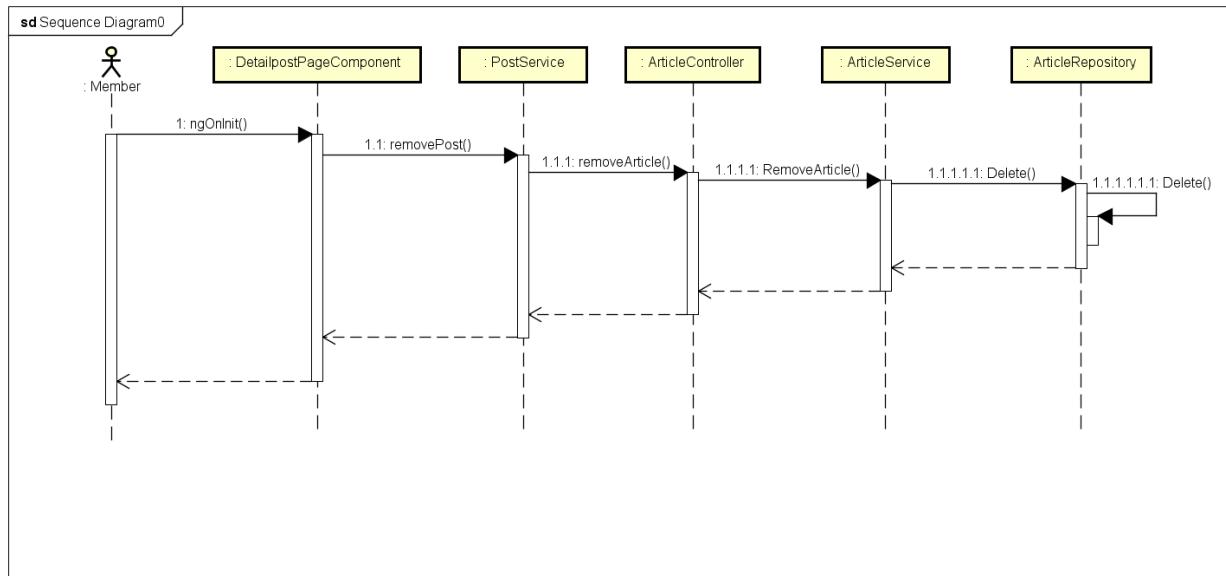


Figure 4-54: Delete A Post Sequence Diagram

4.3.4.13 Like/Unlike A Post

Screen Design

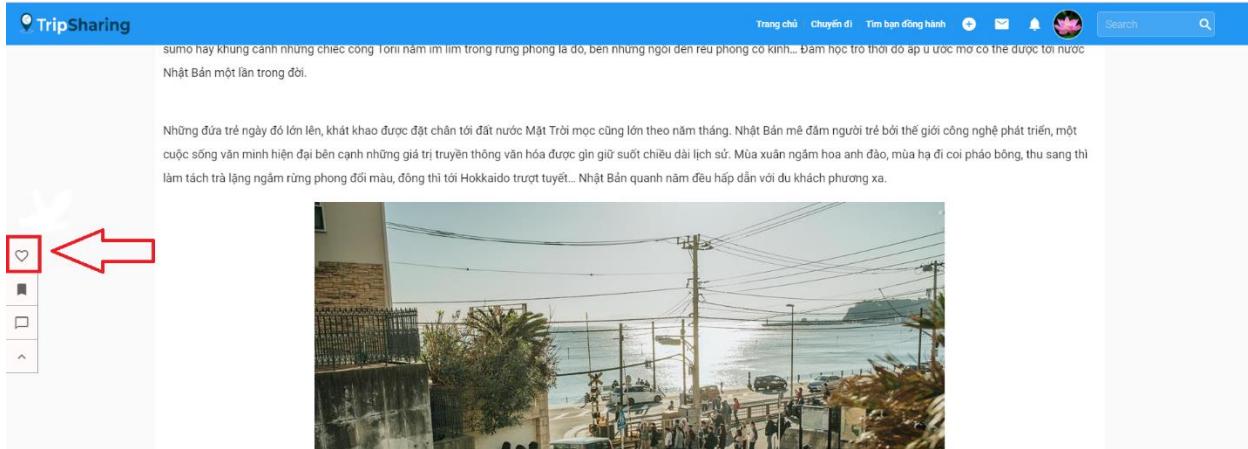


Figure 4-55: Like/Unlike A Post Screen Design

Class Diagram

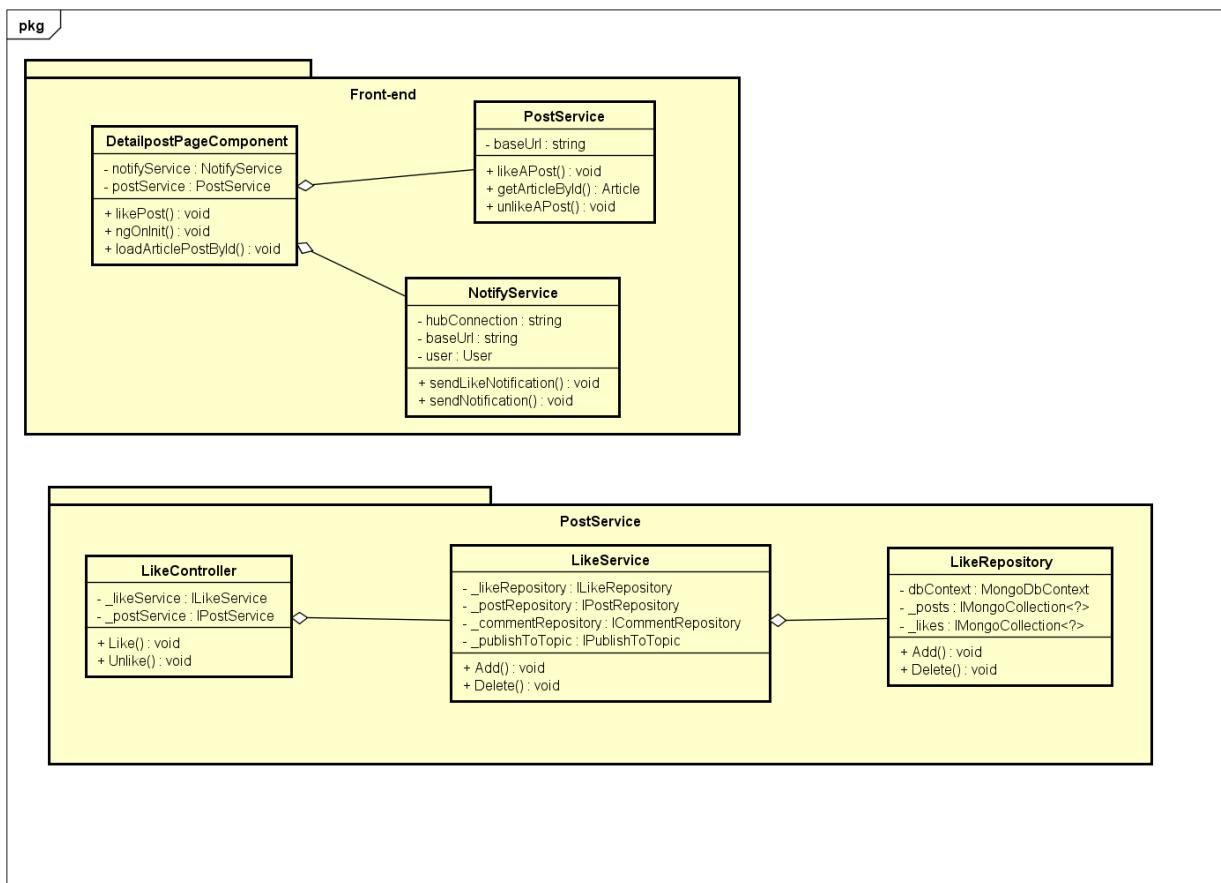


Figure 4-55: Like/Unlike A Post Class Diagram

Class Specification

DetailpostPageComponent

DetailpostPageComponent			
Physical address	src\app\pages\detailpost-page\detailpost-page.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	notifyService	NotifyService	
2	postService	PostService	
Operations			
No	Name	Return Type	Description
1	likePost	Void	
2	ngOnInit	Void	
3	loadArticlePostById	Void	

NotifyService

NotifyService			
Physical address	src\app\core\services\post-service\post.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	hubConnection	String	
2	baseUrl	String	
3	user	User	
Operations			
No	Name	Return Type	Description
1	sendLikeNotification	Void	
2	sendNotification	Void	

PostService

PostService			
Physical address	src\app\core\services\post-service\post.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	baseUrl	String	
Operations			
No	Name	Return Type	Description
1	getArticleById	Article	
2	likeAPost	Void	
3	unlikeAPost	Void	

LikeController

LikeController			
Physical address	src\Services\PostService\PostService\Controllers\LikeController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_likeService	ILikeService	
2	_postService	IPostService	
Operations			
No	Name	Return Type	Description
1	Like	Void	
2	Unlike	Void	

LikeService

LikeService			
Physical address	src\Services\PostService\PostService\services\LikeService.cs		
Base Class	ILikeService		
Attributes			
No	Name	Type	Description
1	_likeRepository	ILikeRepository	
2	_postService	IPostService	
3	_commentRepository	ICommentRepository	
4	_publishToTopic	IPublishToTopic	
Operations			
No	Name	Return Type	Description
1	Add	Void	
2	Delete	Void	

LikeRepository

LikeRepository			
Physical address	src\Services\PostService\PostService\Repositories\LikeRepository.cs		
Base Class	ILikeRepository		
Attributes			
No	Name	Type	Description
1	dbContext	MongoDbContext	
2	_likes	IMongoCollection<?>	
3	_posts	IMongoCollection<?>	
Operations			
No	Name	Return Type	Description
1	Add	Void	
2	Delete	Void	

Sequence Diagram

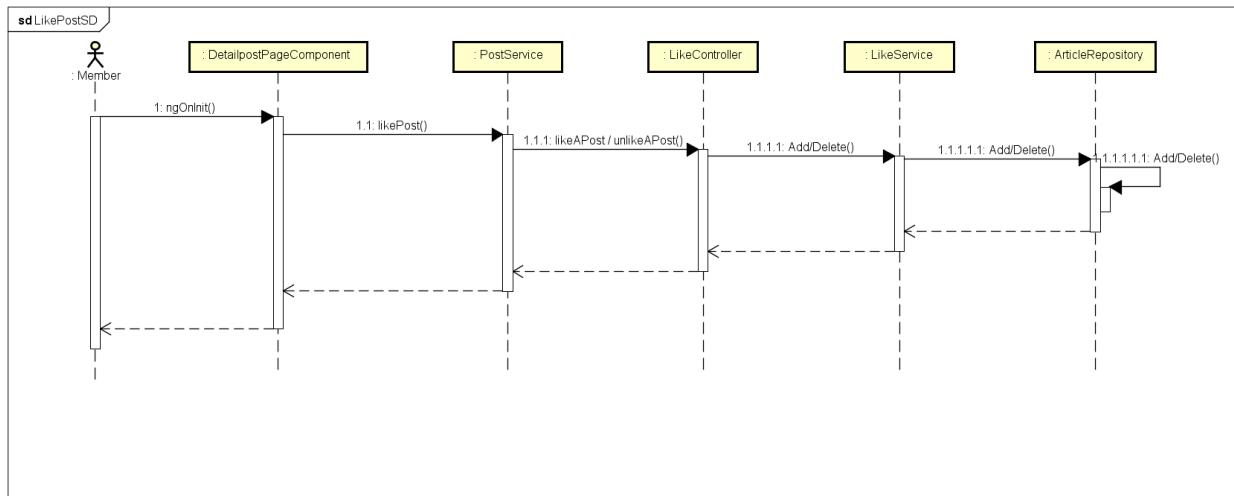


Figure 4-56: Like/Unlike A Post Sequence Diagram

4.3.4.14 Share A Post

Screen Design

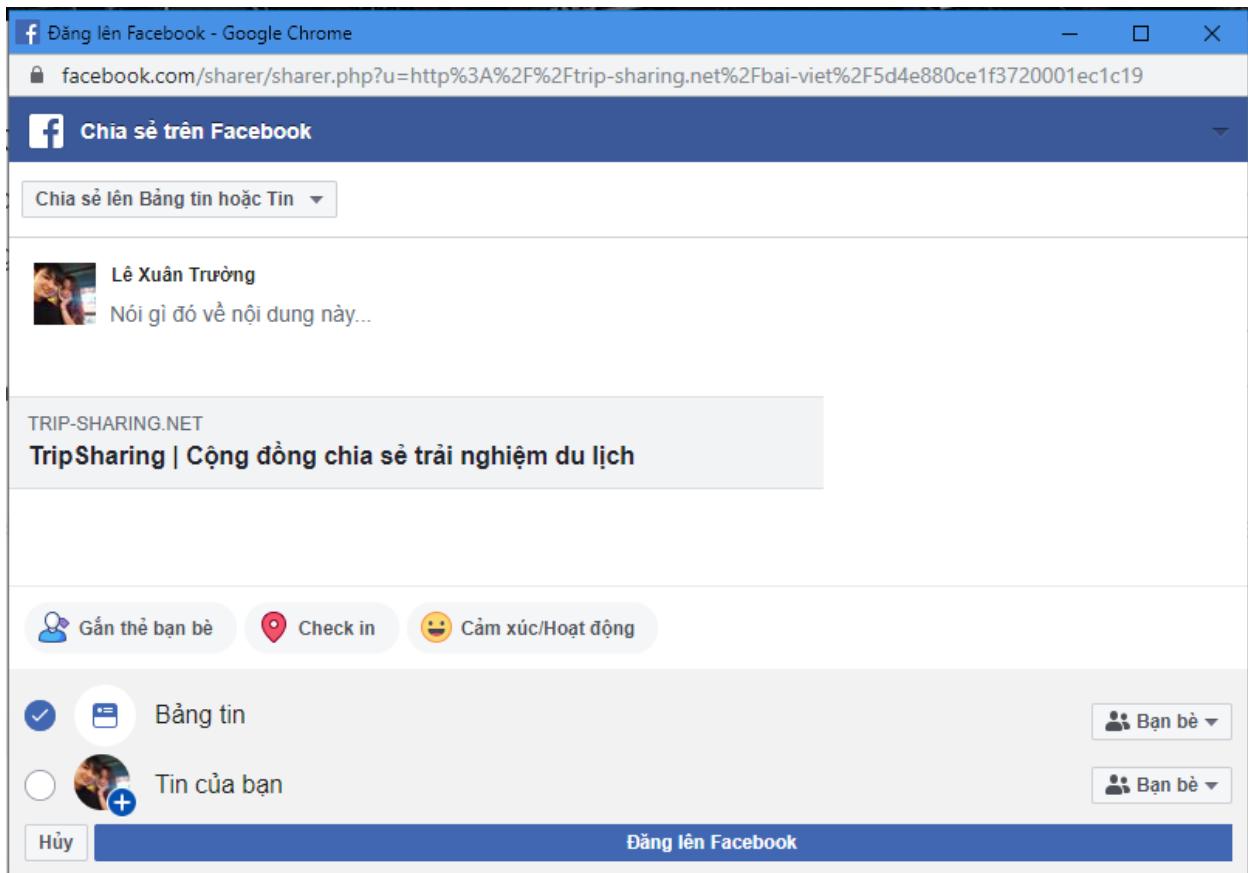


Figure 4-57: Share A Post Screen Design

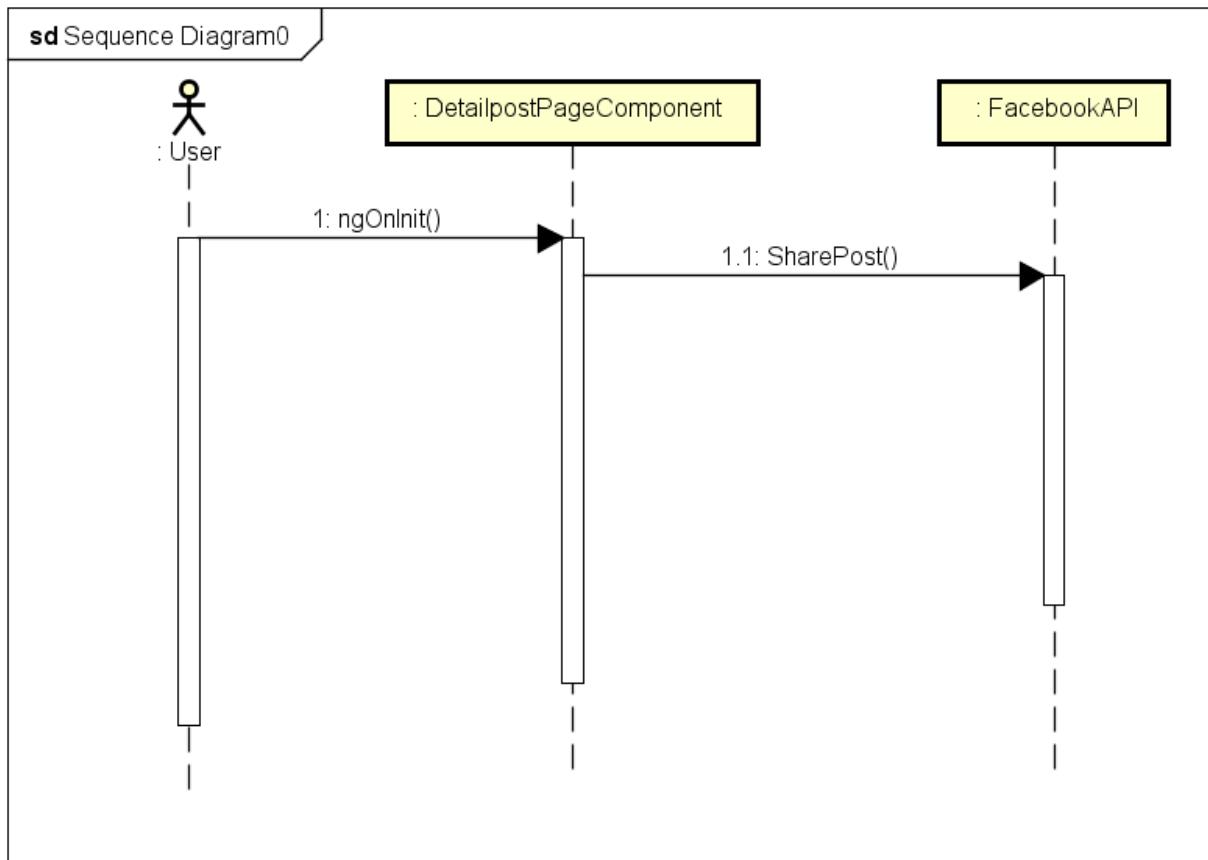


Figure 4-58: Share A Post Sequence Diagram

4.3.4.15 Report A Post

Screen Design

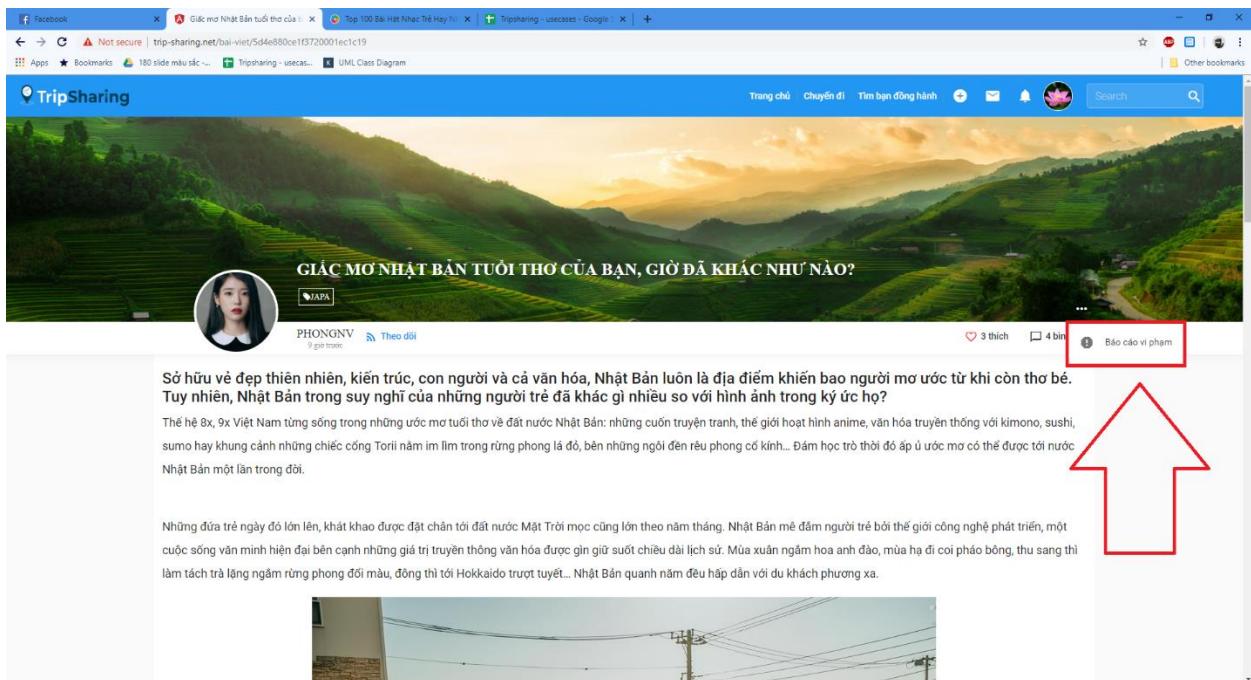


Figure 4-59: Report A Post Screen Design

Class Diagram

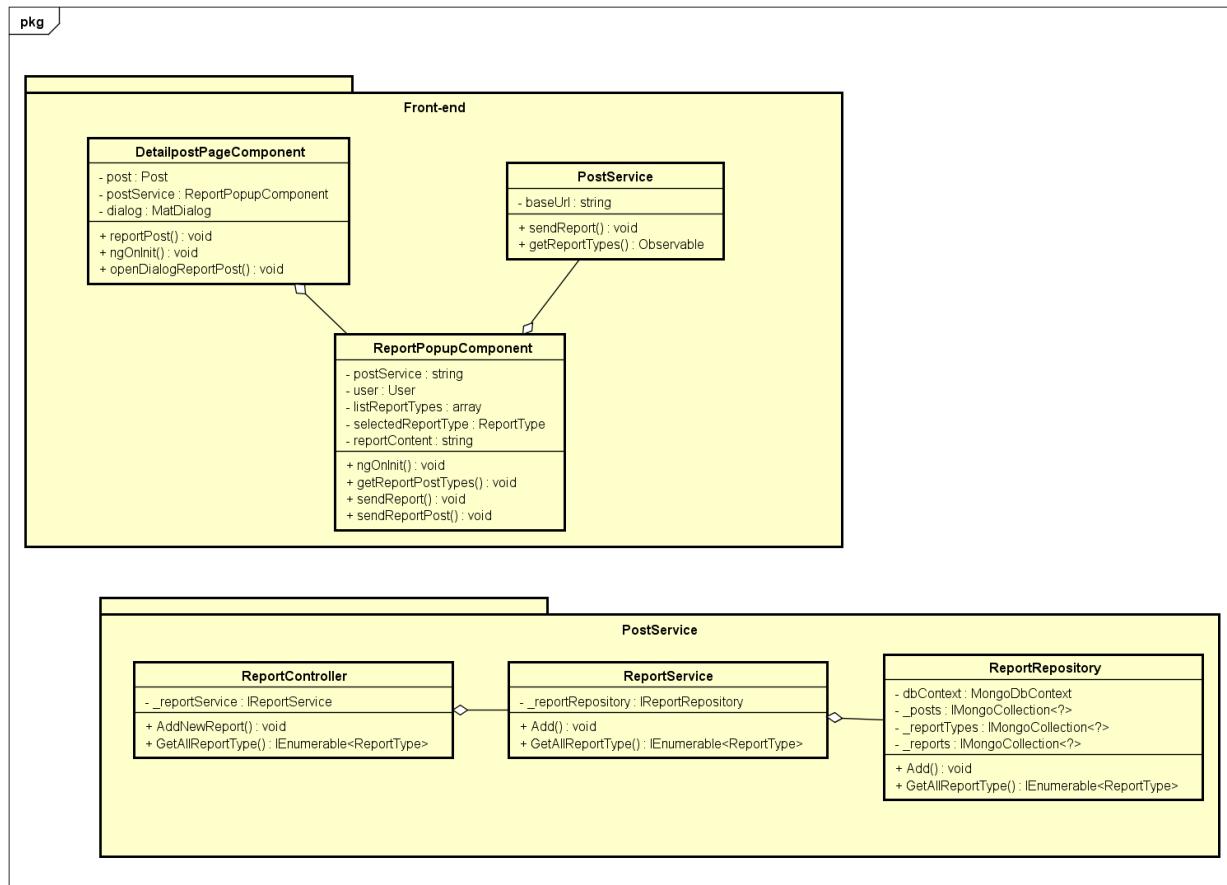


Figure 4-60: Report A Post Class Diagram

Class Specification

DetailpostPageComponent

DetailpostPageComponent			
Physical address	src\app\pages\detailpost-page\detailpost-page.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	post	Post	
2	postService	PostService	
3	dialog	MatDialog	
Operations			
No	Name	Return Type	Description
1	reportPost	Void	
2	ngOnInit	Void	
3	openDialogReportPost	Void	

ReportPopupComponent

ReportPopupComponent			
Physical address	src\app\shared\components\report-popup\report-popup.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	postService	String	
2	reportContent	String	
3	user	User	
4	listReportTypes	Array	
5	selectedReportType	ReportType	
Operations			
No	Name	Return Type	Description
1	ngOnInit	Void	
2	getReportPostTypes	Void	
3	sendReportPost	Void	
4	sendReport	Void	

PostService

PostService			
Physical address	src\app\core\services\post-service\post.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	baseUrl	String	
Operations			
No	Name	Return Type	Description
1	sendReport	Void	
2	getReportTypes	Observable	

ReportController

ReportController			
Physical address	src\Services\PostService\PostService\Controllers\ReportController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_reportService	IReportService	
Operations			
No	Name	Return Type	Description
1	AddNewReport	Void	
2	GetAllReportType	IEnumerable<ReportType>	

ReportService

ReportService			
Physical address	src\Services\PostService\PostService\Services\ReportService.cs		
Base Class	IReportService		
Attributes			
No	Name	Type	Description
1	_reportRepository	IReportRepository	
Operations			
No	Name	Return Type	Description
1	Add	Void	
2	GetAllReportType	IEnumerable<ReportType>	

ReportRepository

ReportRepository			
Physical address	src\Services\PostService\PostService\Repositories\ReportRepository.cs		
Base Class	IReportRepository		
Attributes			
No	Name	Type	Description
1	dbContext	MongoDbContext	
2	_reportTypes	IMongoCollection<?>	
3	_posts	IMongoCollection<?>	
4	_reports	IMongoCollection<?>	
Operations			
No	Name	Return Type	Description
1	Add	Void	
2	GetAllReportType	IEnumerable<ReportType>	

Sequence Diagram

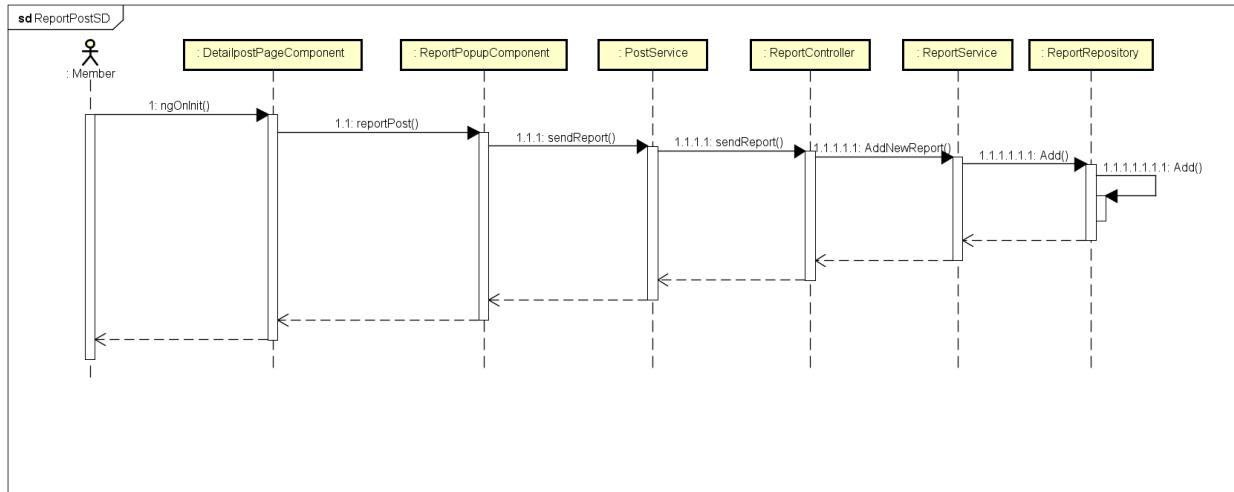


Figure 4-61: Report A Post Sequence Diagram

4.3.4.16 Bookmark a post

Screen Design

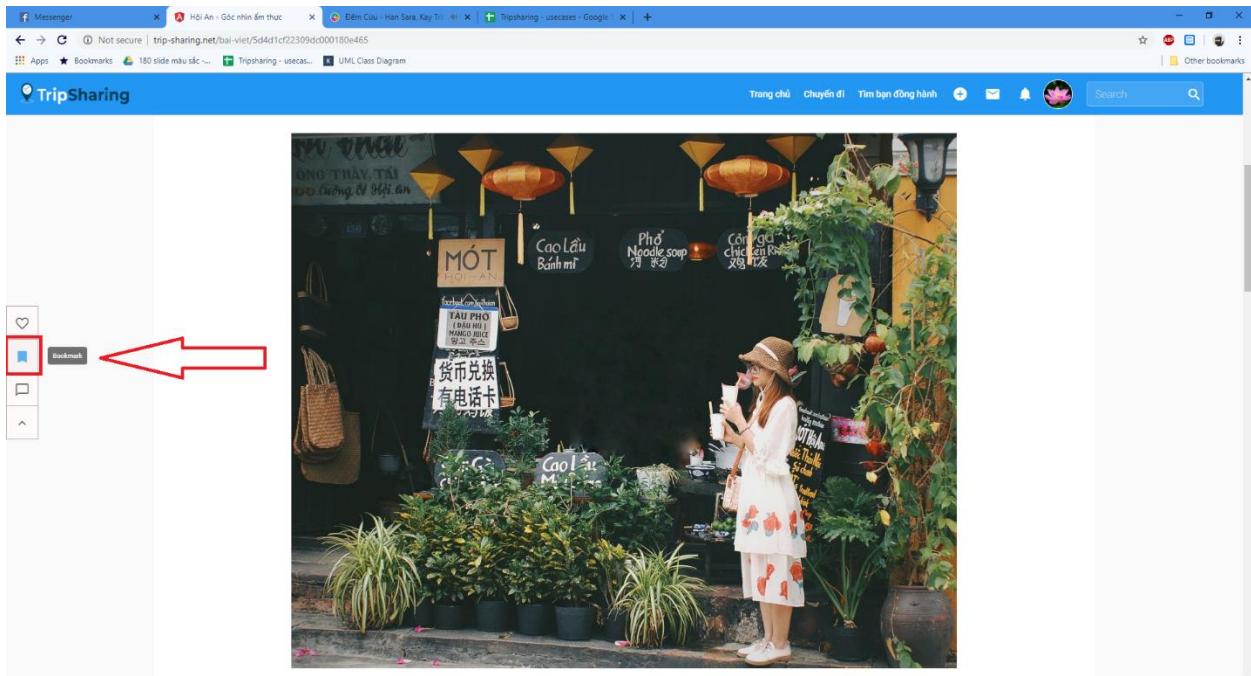


Figure 4-62: Bookmark A Post Screen Design

Class Diagram

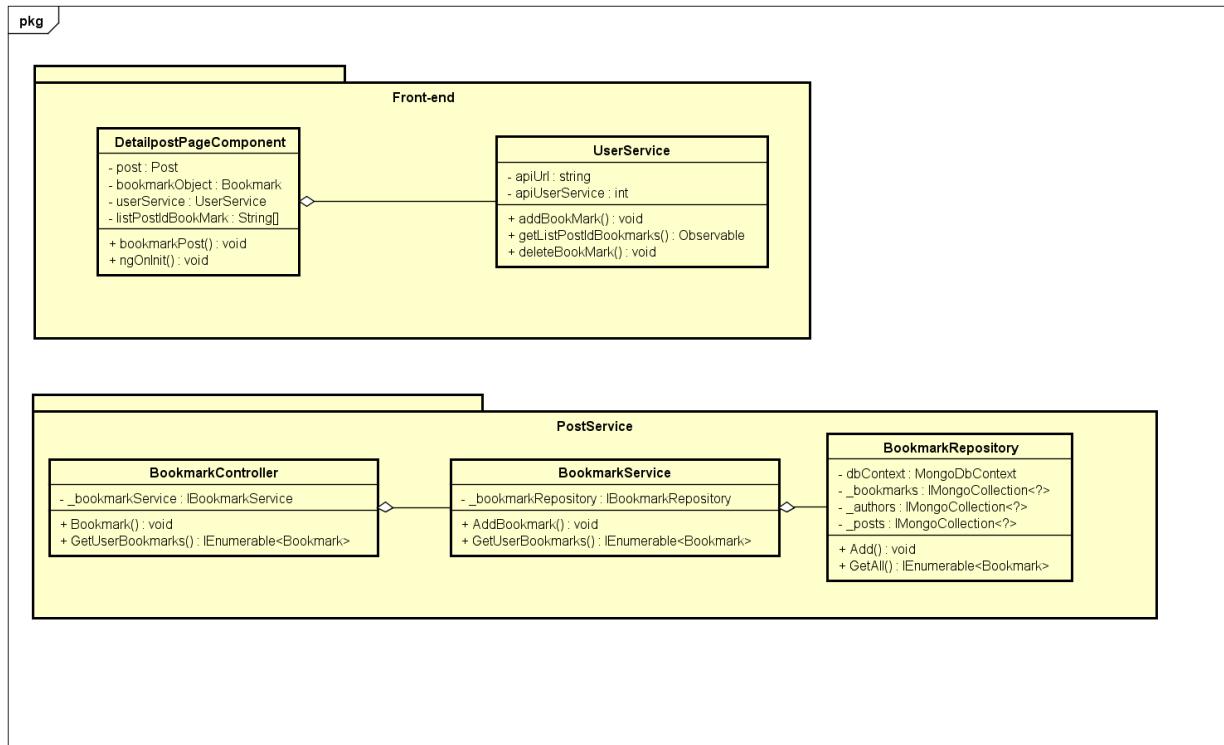


Figure 4-63: Bookmark A Post Class Diagram

Class Specification

DetailpostPageComponent

DetailpostPageComponent			
Physical address	src\app\pages\detailpost-page\detailpost-page.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	post	Post	
2	bookmarkObject	Bookmark	
3	userService	UserService	
4	listPostIdBookMark	String[]	
Operations			
No	Name	Return Type	Description
1	bookmarkPost	Void	
2	ngOnInit	Void	

UserService

UserService			
Physical address	src\app\core\services\user-service\user.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	apiUrl	string	
2	apiUserService	int	
3	addBookMark	void	
4	getListPostIdBookmarks	Observable	
5	deleteBookMark	void	

Attributes			
No	Name	Type	Description
1	apiUrl	String	
2	apiUserService	String	
Operations			
No	Name	Return Type	Description
1	addBookMark	Void	
2	getListPostIdBookmarks	Observable	
3	deleteBookMark	Void	

BookmarkController

BookmarkController			
Physical address	src\Services\PostService\PostService\Controllers\BookmarkController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_bookmarkService	IBookmarkService	
Operations			
No	Name	Return Type	Description
1	Bookmark	Void	
2	GetUserBookmarks	IEnumerable<Bookmark>	

BookmarkService

BookmarkService			
Physical address	src\Services\PostService\PostService\services\BookmarkService.cs		
Base Class	IBitmapService		
Attributes			
No	Name	Type	Description
1	_bookmarkRepository	IBookmarkRepository	
Operations			
No	Name	Return Type	Description
1	AddBookmark	Void	
2	GetUserBookmarks	IEnumerable<Bookmark>	

BookmarkRepository

BookmarkRepository			
Physical address	src\Services\PostService\PostService\Repositories\BookmarkRepository.cs		
Base Class	IBitmapRepository		
Attributes			
No	Name	Type	Description

No	Name	Type	Description
1	dbContext	MongoDbContext	
2	_bookmarks	IMongoCollection<?>	
3	_posts	IMongoCollection<?>	
4	_authors	IMongoCollection<?>	
Operations			
No	Name	Return Type	Description
1	Add	Void	
2	GetAll	IEnumerable<Bookmark>	

Sequence Diagram

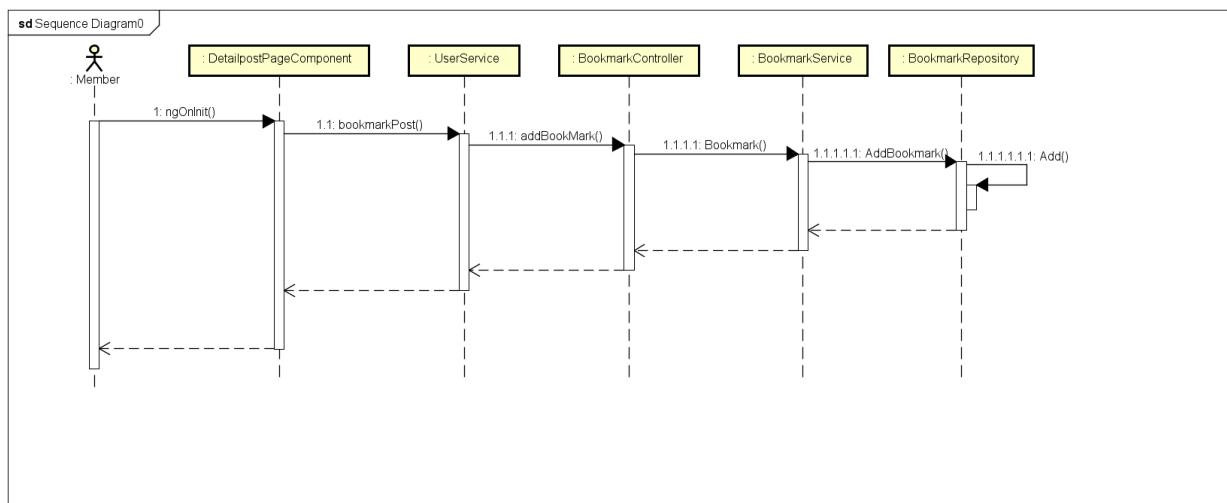


Figure 4-64: Bookmark A Post Sequence Diagram

4.3.4.17 View All Bookmark

Screen Design

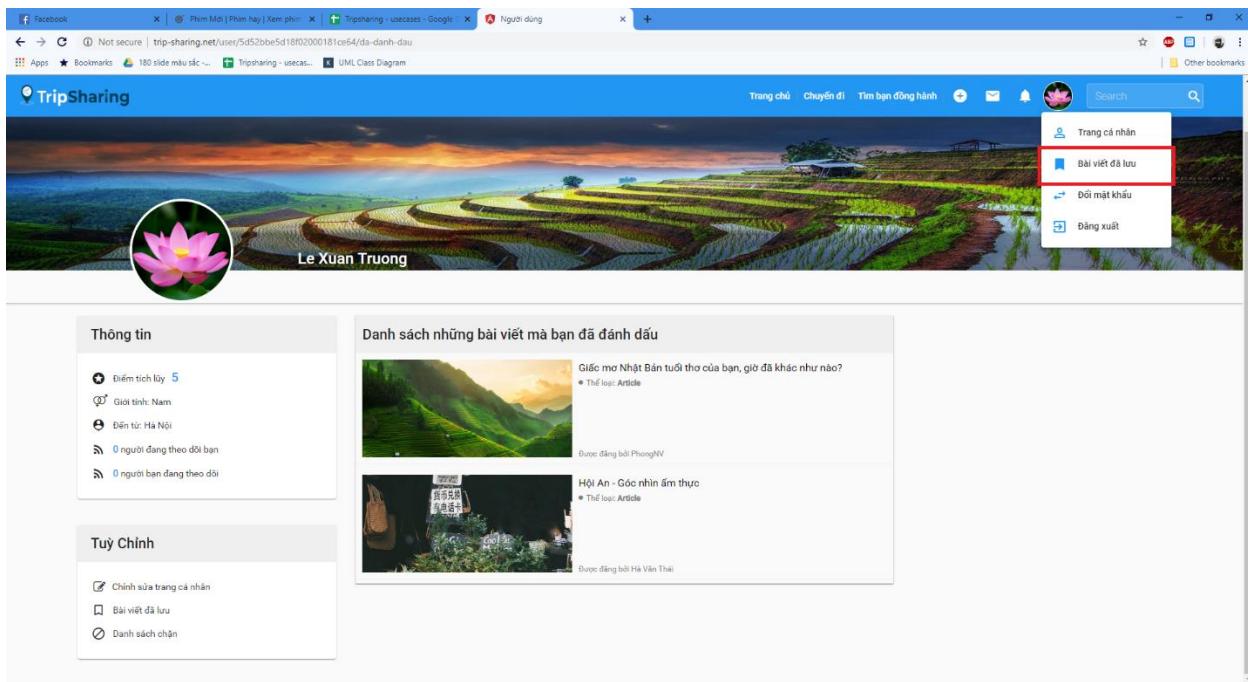


Figure 4-65: View All Bookmark Screen Design

Class Diagram

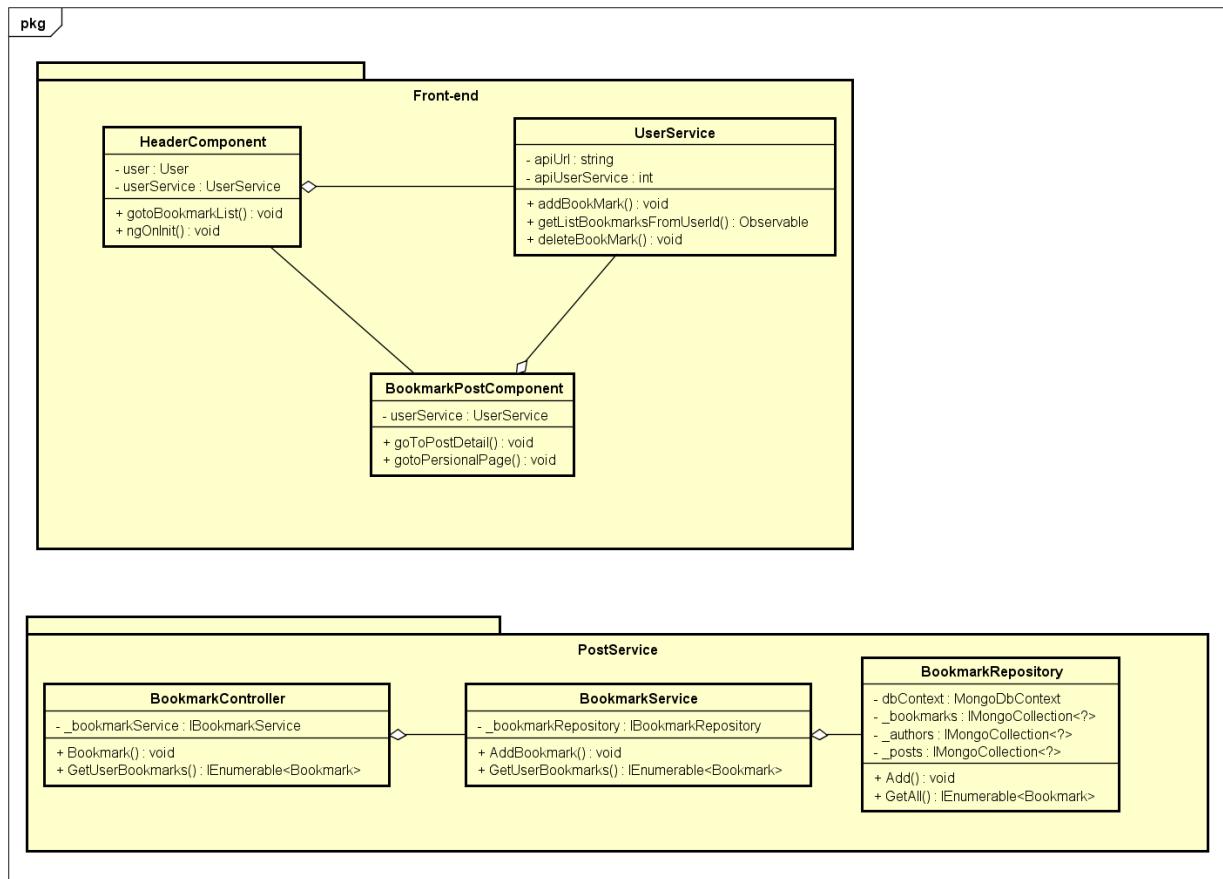


Figure 4-66: View All Bookmark Class Diagram

Class Specification

HeaderComponent

HeaderComponent			
Physical address	src\app\core\components\header\header.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	user	Post	
2	userService	UserService	
Operations			
No	Name	Return Type	Description
1	bookmarkPost	Void	
2	ngOnInit	Void	

BookmarkPostComponent

BookmarkPostComponent			
Physical address	src\app\shared\components\bookmark-post\bookmark-post.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
2	userService	UserService	
Operations			
No	Name	Return Type	Description
1	goToPostDetail	Void	
2	gotoPersionalPage	Void	

UserService

UserService			
Physical address	src\app\core\services\user-service\user.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	apiUrl	String	
2	apiUserService	String	
Operations			
No	Name	Return Type	Description
1	addBookMark	Void	
2	getListBookmarksFromUserId	Observable	
3	deleteBookMark	Void	

BookmarkController

BookmarkController			
Physical address	src\Services\PostService\PostService\Controllers\BookmarkController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_bookmarkService	IBookmarkService	
Operations			
No	Name	Return Type	Description
1	Bookmark	Void	
2	GetUserBookmarks	IEnumerable<Bookmark>	

BookmarkService

BookmarkService			
Physical address	src\Services\PostService\PostService\Services\BookmarkService.cs		
Base Class	IBookmarkService		
Attributes			
No	Name	Type	Description
1	_bookmarkRepository	IBookmarkRepository	
Operations			
No	Name	Return Type	Description
1	AddBookmark	Void	
2	GetUserBookmarks	IEnumerable<Bookmark>	

BookmarkRepository

BookmarkRepository			
Physical address	src\Services\PostService\PostService\Repositories\BookmarkRepository.cs		
Base Class	IBookmarkRepository		
Attributes			
No	Name	Type	Description
1	dbContext	MongoDbContext	
2	_bookmarks	IMongoCollection<?>	
3	_posts	IMongoCollection<?>	
4	_authors	IMongoCollection<?>	
Operations			
No	Name	Return Type	Description
1	Add	Void	
2	GetAll	IEnumerable<Bookmark>	

Sequence Diagram

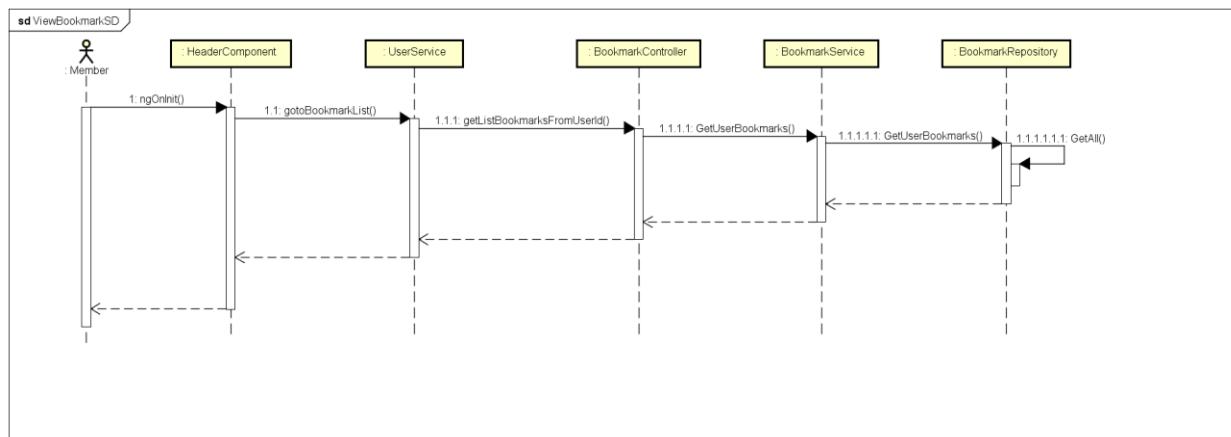


Figure 4-66: View All Bookmark Sequence Diagram

4.3.4.18 Remove A Bookmark

Screen Design

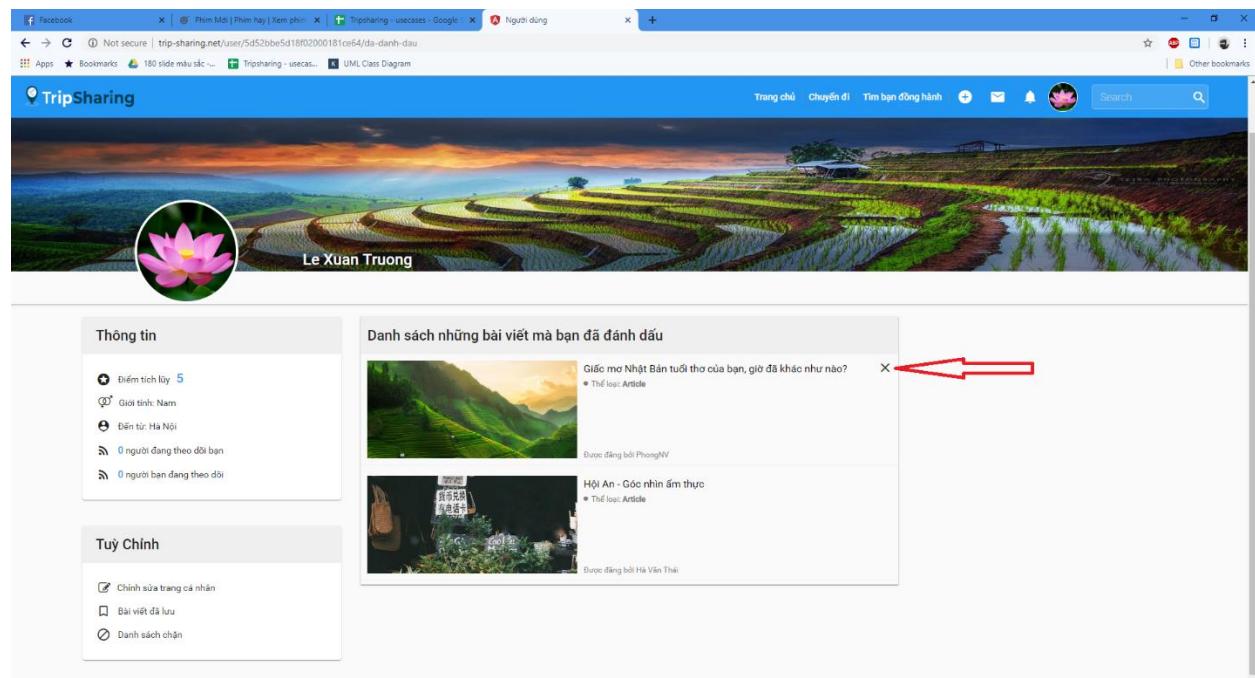


Figure 4-67: Remove A Bookmark Screen Design

Class Diagram

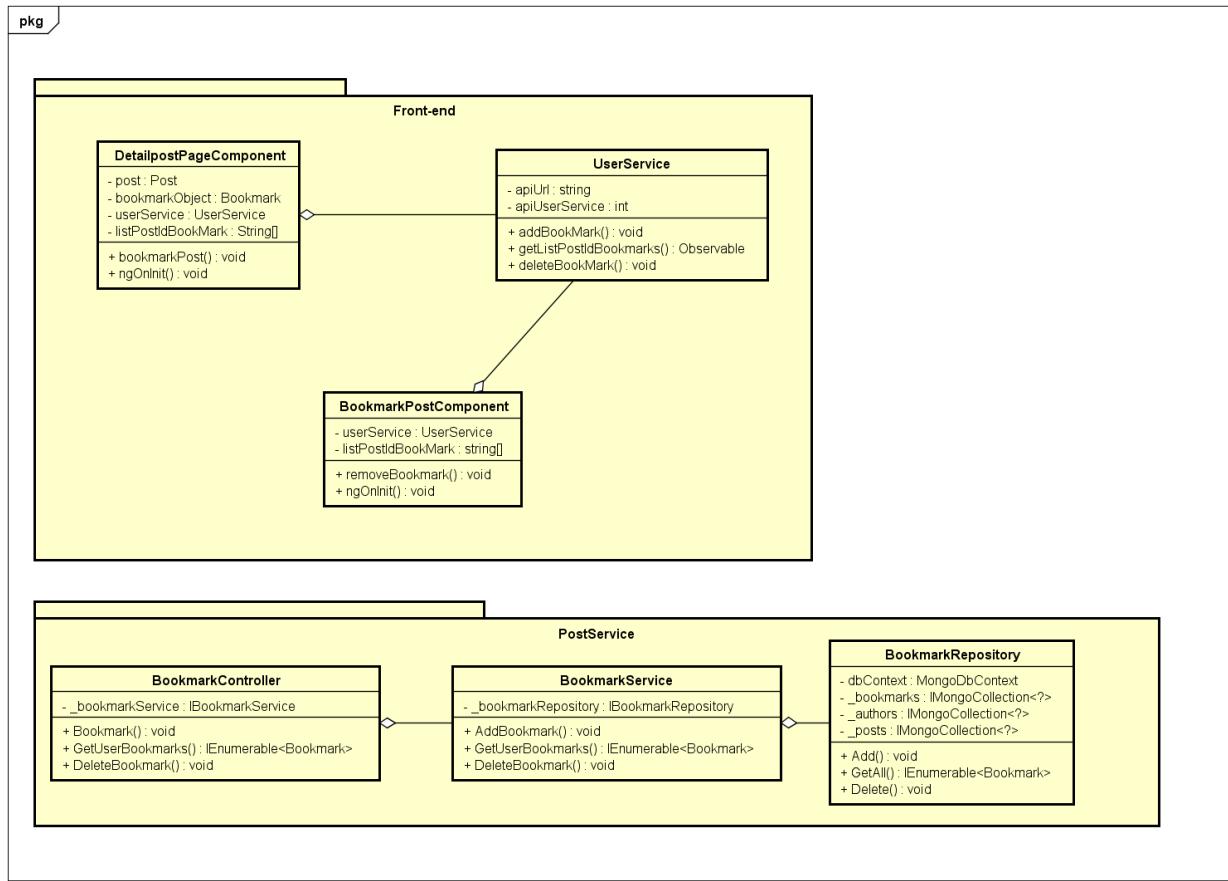


Figure 4-67: Remove A Bookmark Class Diagram

Class Specification

DetailpostIdPageComponent

DetailpostIdPageComponent			
Physical address	src\app\pages\detailpost-page\detailpost-page.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	post	Post	
2	bookmarkObject	Bookmark	
3	userService	UserService	
4	listPostIdBookMark	String[]	
Operations			
No	Name	Return Type	Description
1	bookmarkPost	Void	
2	ngOnInit	Void	
3	DeleteBookmark	Void	

BookmarkPostComponent

BookmarkPostComponent			
Physical address	src\app\shared\components\bookmark-post\bookmark-post.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	post	Post	
2	bookmarkObject	Bookmark	
Operations			
No	Name	Return Type	Description
1	removeBookmark	Void	
2	ngOnInit	Void	

UserService

UserService			
Physical address	src\app\core\services\user-service\user.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	apiUrl	String	
2	apiUserService	String	
Operations			
No	Name	Return Type	Description
1	addBookMark	Void	
2	getListPostIdBookmarks	Observable	
3	deleteBookMark	Void	

BookmarkController

BookmarkController			
Physical address	src\Services\PostService\PostService\Controllers\BookmarkController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_bookmarkService	IBookmarkService	
Operations			
No	Name	Return Type	Description
1	Bookmark	Void	
2	GetUserBookmarks	IEnumerable<Bookmark>	
3	DeleteBookmark	Void	

BookmarkService

BookmarkService			
Physical address	src\Services\PostService\PostService\Services\BookmarkService.cs		
Base Class	IBookmarkService		
Attributes			
No	Name	Type	Description
1	_bookmarkRepository	IBookmarkRepository	
Operations			
No	Name	Return Type	Description
1	AddBookmark	Void	
2	GetUserBookmarks	IEnumerable<Bookmark>	
3	DeleteBookmark	Void	

BookmarkRepository

BookmarkRepository			
Physical address	src\Services\PostService\PostService\Repositories\BookmarkRepository.cs		
Base Class	IBitmapRepository		
Attributes			
No	Name	Type	Description
1	dbContext	MongoDbContext	
2	_bookmarks	IMongoCollection<?>	
3	_posts	IMongoCollection<?>	
4	_authors	IMongoCollection<?>	
Operations			
No	Name	Return Type	Description
1	Add	Void	
2	GetAll	IEnumerable<Bookmark>	
3	Delete	Void	

Sequence Diagram

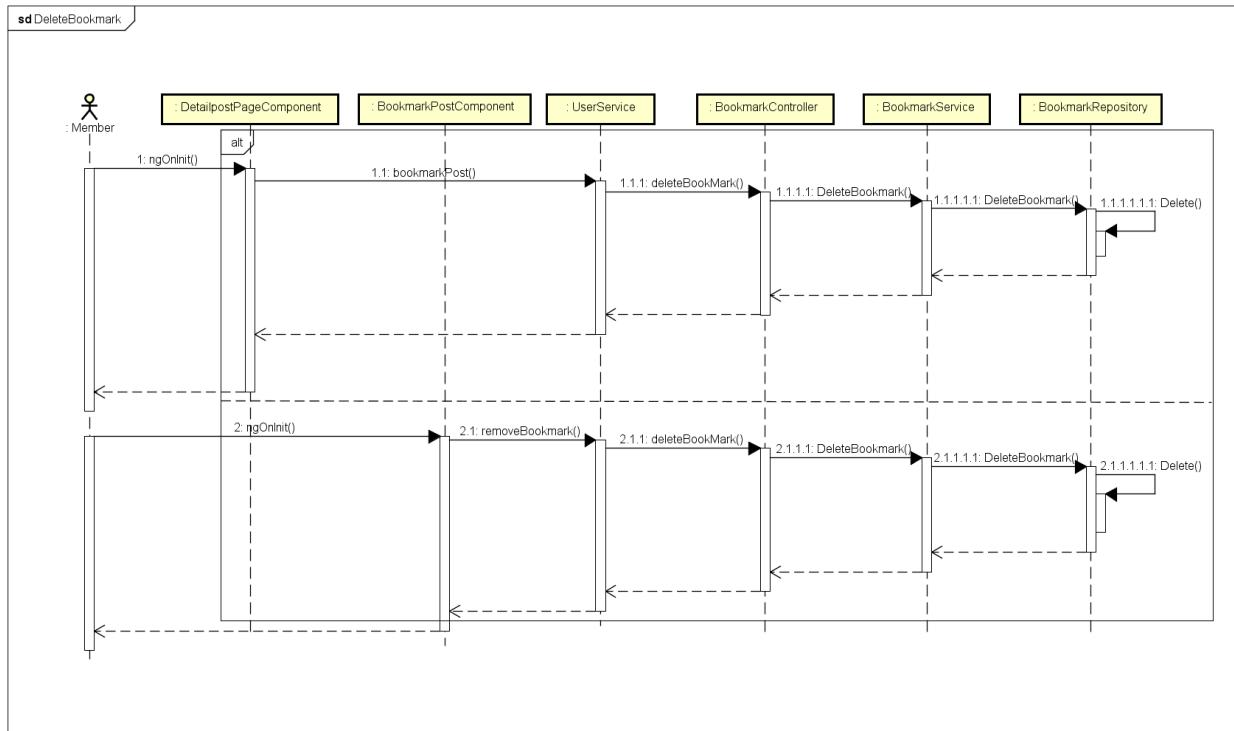


Figure 4-68: Remove A Bookmark Sequence Diagram

4.3.4.19 Comment to A Post

Screen Design

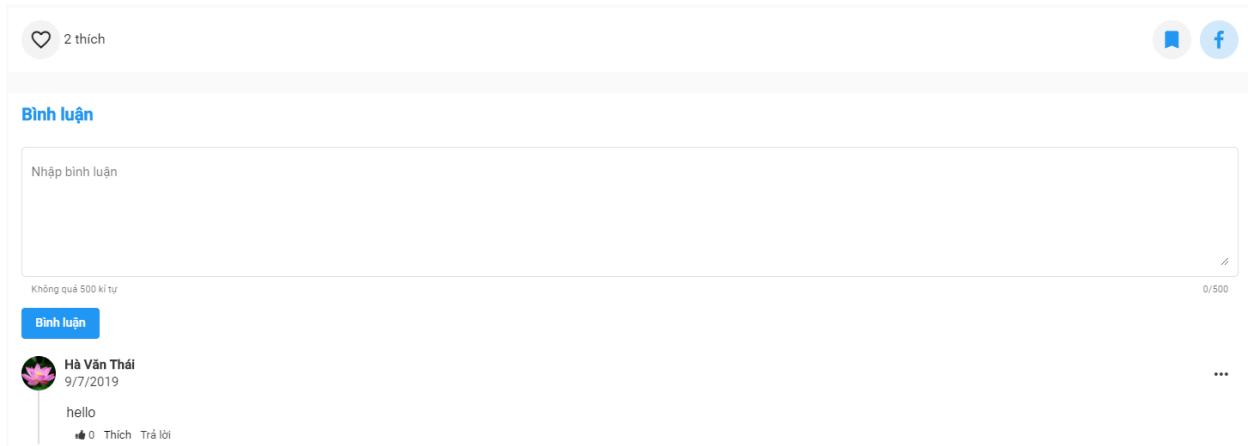


Figure 4-69: Comment to A Post Screen Design

Class Diagram

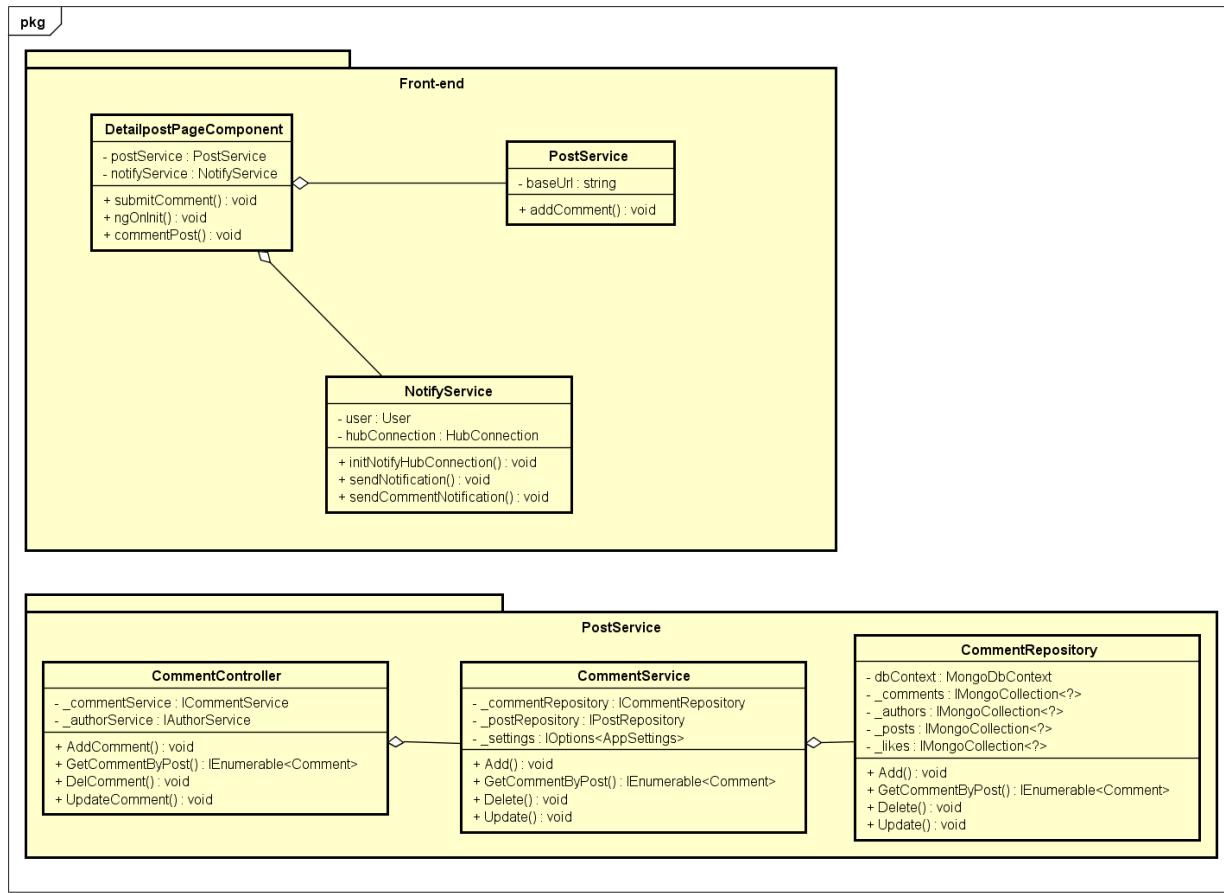


Figure 4-70: Comment to A Post Class Diagram

Class Specification

DetailpostPageComponent

DetailpostPageComponent			
Physical address	src\app\pages\detailpost-page\detailpost-page.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	postService	PostService	
2	notifyService	NotifyService	
Operations			
No	Name	Return Type	Description
1	submitComment	Void	
2	ngOnInit	Void	
3	commentPost	Void	

PostService

PostService			
Physical address	src\app\core\services\post-service\post.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	baseUrl	String	
Operations			
No	Name	Return Type	Description
1	addComment	Void	

NotifyService

NotifyService			
Physical address	src\app\core\services\notify-service\notify.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	user	User	
2	hubConnection	HubConnection	
Operations			
No	Name	Return Type	Description
1	initNotifyHubConnection	Void	
2	sendNotification	Void	
3	sendCommentNotification	Void	

CommentController

CommentController			
Physical address	src\Services\PostService\PostService\Controllers\CommentController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_commentService	ICommentService	
2	_authorService	IAuthorService	
Operations			
No	Name	Return Type	Description
1	AddComment	Void	
2	GetCommentByPost	IEnumerable<Comment>	
3	DelComment	Void	
4	UpdateComment	Void	

CommentService

CommentService			
Physical address	src\Services\PostService\PostService\CommentService.cs		
Base Class	ICommentService		
Attributes			
No	Name	Type	Description
1	_commentRepository	ICommentRepository	
2	_postRepository	IPostRepository	
3	_settings	IOptions<AppSettings>	
Operations			
No	Name	Return Type	Description
1	Add	Void	
2	GetCommentByPost	IEnumerable<Comment>	
3	Delete	Void	
4	Update	Void	

CommentRepository

CommentRepository			
Physical address	src\Services\PostService\PostService\Repositories\CommentRepository.cs		
Base Class	IBookmarkRepository		
Attributes			
No	Name	Type	Description
1	dbContext	MongoDbContext	
2	_comments	IMongoCollection<?>	
3	_posts	IMongoCollection<?>	
4	_authors	IMongoCollection<?>	
5	_likes	IMongoCollection<?>	
Operations			
No	Name	Return Type	Description
1	Add	Void	
2	GetCommentByPost	IEnumerable<Comment>	
3	Delete	Void	
4	Update	Void	

Sequence Diagram

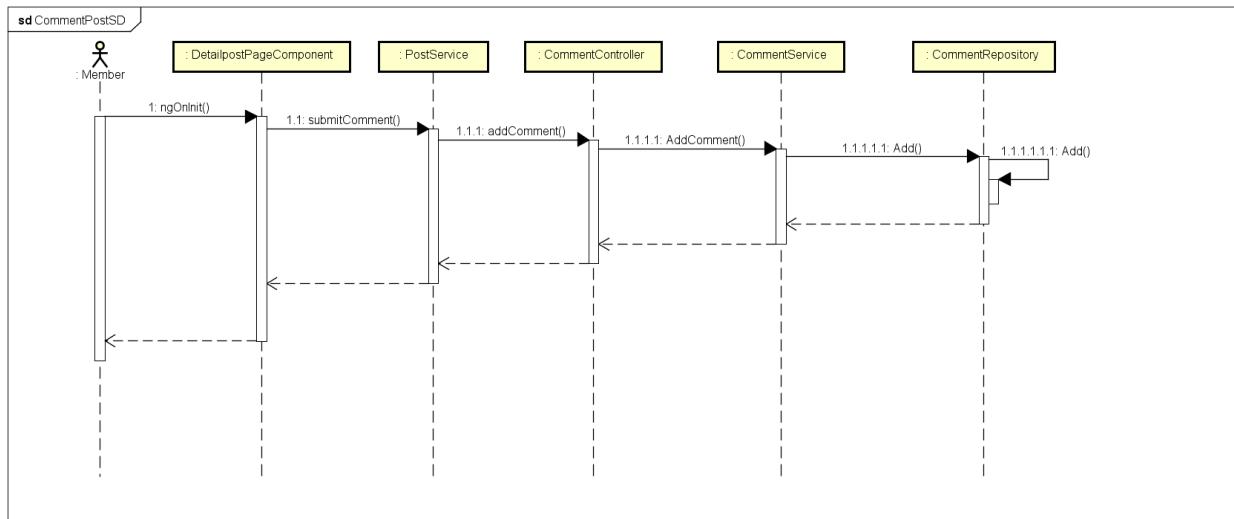


Figure 4-71: Comment to A Post Sequence Diagram

4.3.4.20 Edit A Comment

Screen Design

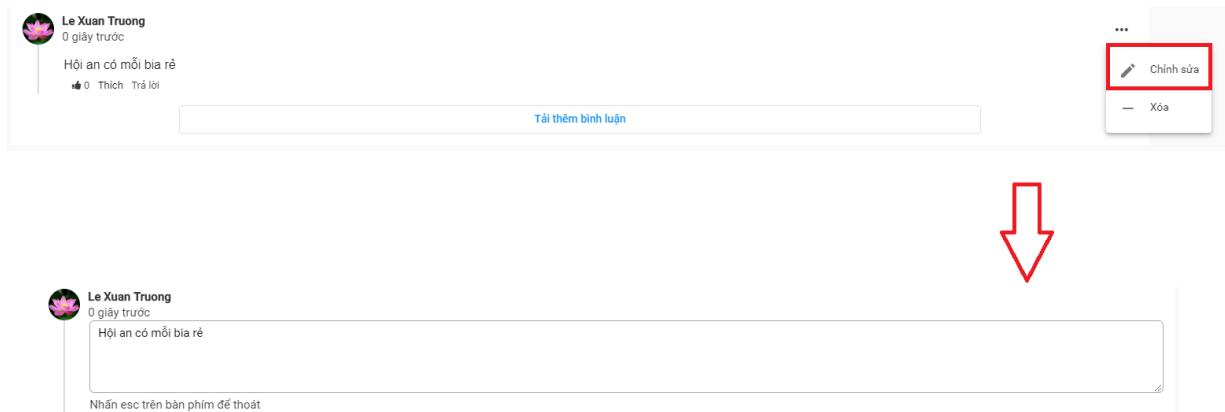


Figure 4-72: Edit A Comment Screen Design

Class Diagram

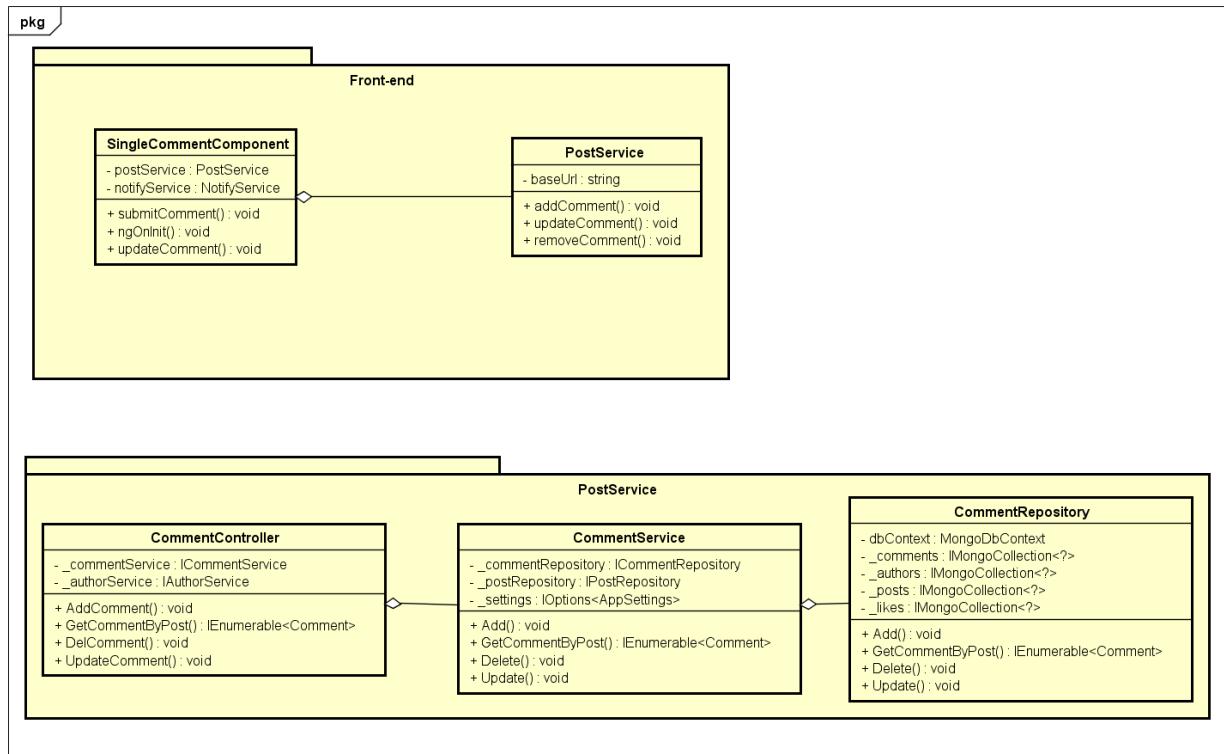


Figure 4-73: Edit A Comment Class Diagram

Class Specification

SingleCommentComponent

SingleCommentComponent			
Physical address	src\app\shared\components\single-comment\single-comment.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	postService	PostService	
2	notifyService	NotifyService	
Operations			
No	Name	Return Type	Description
1	submitComment	Void	
2	ngOnInit	Void	
3	updateComment	Void	

PostService

PostService			
Physical address	src\app\core\services\post-service\post.service.ts		
Base Class	Class		
Attributes			

No	Name	Type	Description
1	baseUrl	String	
Operations			
No	Name	Return Type	Description
1	addComment	Void	
2	updateComment	Void	
3	removeComment	Void	

CommentController

CommentController			
Physical address	src\Services\PostService\PostService\Controllers\CommentController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_commentService	ICommentService	
2	_authorService	IAuthorService	
Operations			
No	Name	Return Type	Description
1	AddComment	Void	
2	GetCommentByPost	IEnumerable<Comment>	
3	DelComment	Void	
4	UpdateComment	Void	

CommentService

CommentService			
Physical address	src\Services\PostService\PostService\CommentService.cs		
Base Class	ICommentService		
Attributes			
No	Name	Type	Description
1	_commentRepository	ICommentRepository	
2	_postRepository	IPostRepository	
3	_settings	IOptions<AppSettings>	
Operations			
No	Name	Return Type	Description
1	Add	Void	
2	GetCommentByPost	IEnumerable<Comment>	
3	Delete	Void	
4	Update	Void	

CommentRepository

CommentRepository			
Physical address	src\Services\PostService\PostService\Repositories\CommentRepository.cs		
Base Class	IBookmarkRepository		
Attributes			
No	Name	Type	Description
1	dbContext	MongoDbContext	
2	_comments	IMongoCollection<?>	
3	_posts	IMongoCollection<?>	
4	_authors	IMongoCollection<?>	
5	_likes	IMongoCollection<?>	
Operations			
No	Name	Return Type	Description
1	Add	Void	
2	GetCommentByPost	IEnumerable<Comment>	
3	Delete	Void	
4	Update	Void	

Sequence Diagram

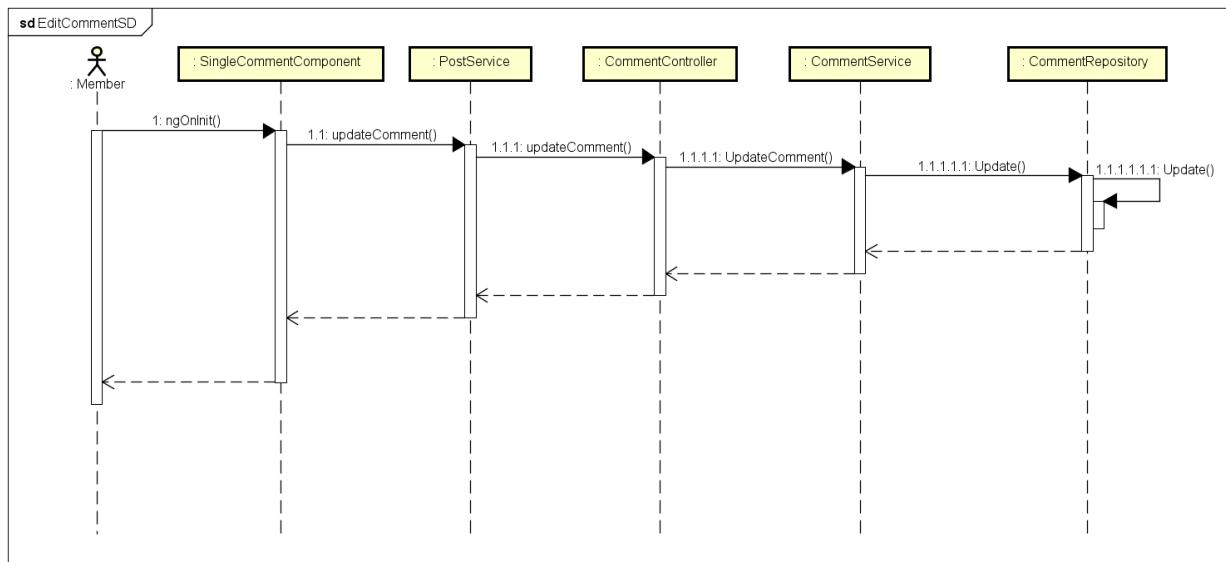


Figure 4-74: Edit A Comment Class Diagram

4.3.4.21 Delete a comment

Screen design

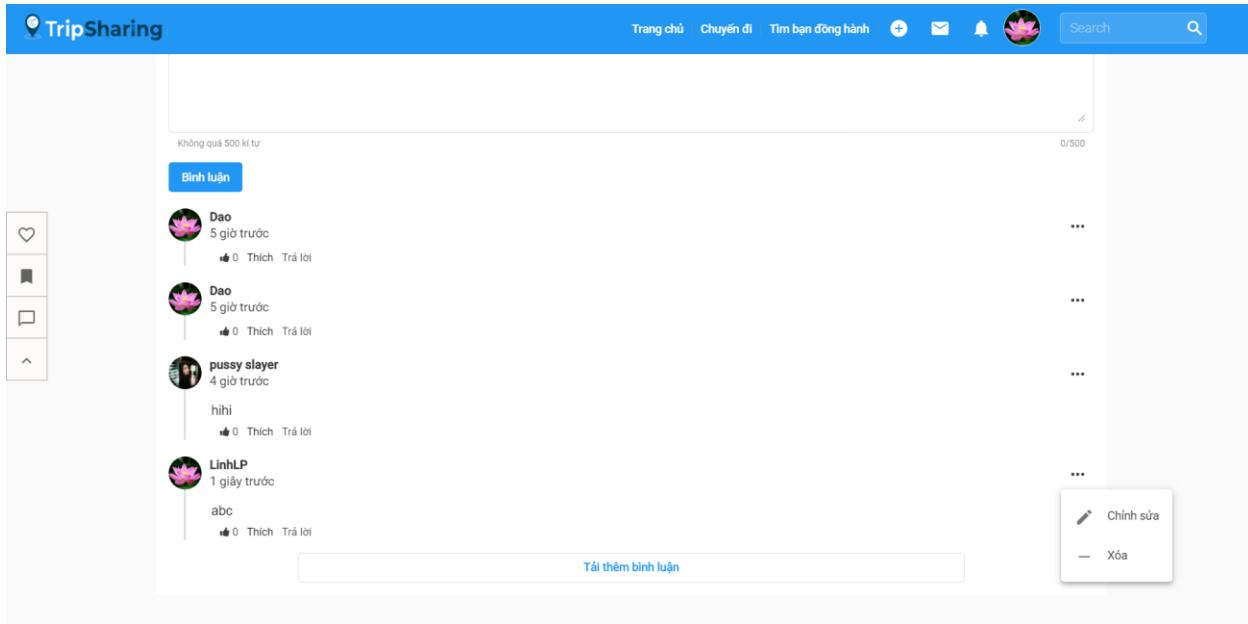


Figure 4-75: Delete A Comment Screen Design

Class Diagram

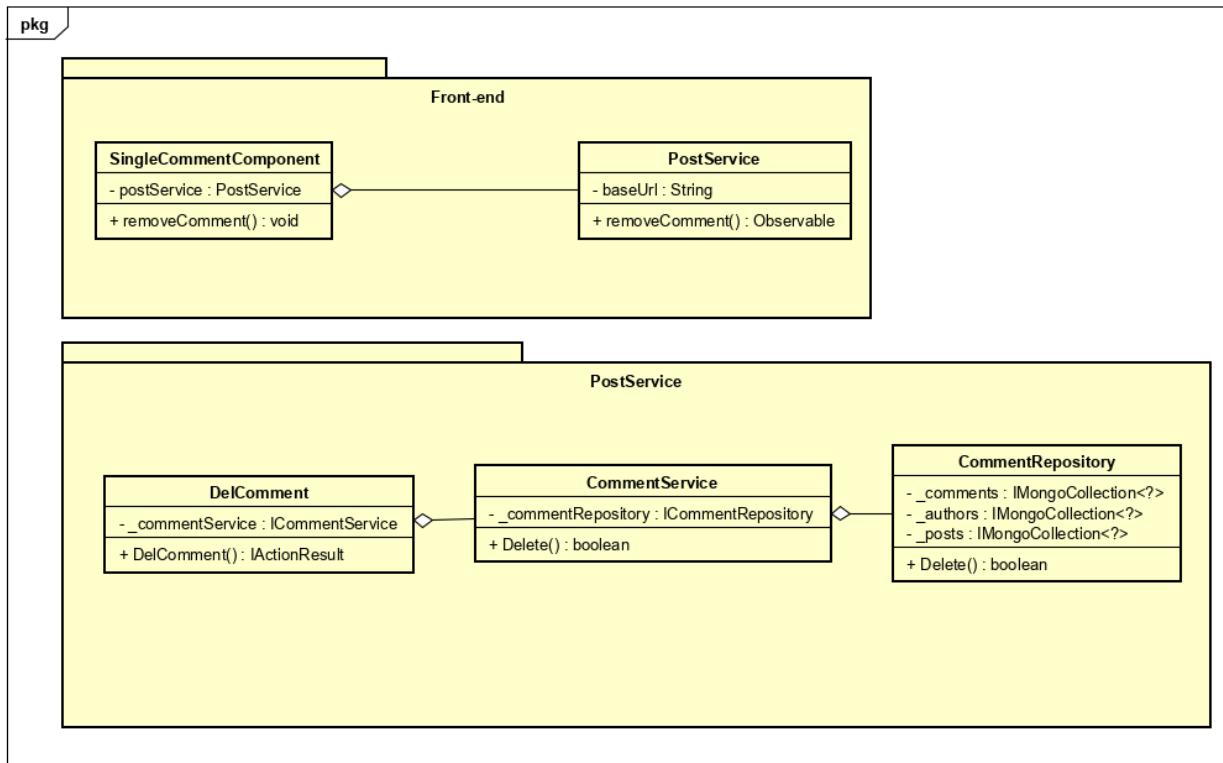


Figure 4-76: Delete A Comment Class Diagram

Class Specification

SingleCommentComponent

SingleCommentComponent			
Physical address	src\app\shared\components\single-comment\single-comment.component.spec.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	postService	PostService	
Operations			
No	Name	Return Type	Description
1	removeComment	void	

PostService

PostService			
Physical address	src\app\core\services\post-service\virtual-trip.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	baseUrl	String	
Operations			
No	Name	Return Type	Description
1	removeComment	Observable	

Comment

CommentController			
Physical address	src\Services\PostService\PostService\Controllers\CommentController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_commentService	ICommentService	
Operations			
No	Name	Return Type	Description
1	DelComment	IActionResult	

CommentService

CommentService			
Physical address	src\Services\PostService\PostService\Services\CommentService.cs		
Base Class	ICommentService		
Attributes			
No	Name	Type	Description

1	_commentRepository	ICommentRepository	
Operations			
No	Name	Return Type	Description
1	Delete	Bool	

CommentRepository

CommentRepository			
Physical address	src\Services\PostService\PostService \\Repositories\CommentRepository.cs		
Base Class	ICommentRepository		
Attributes			
No	Name	Type	Description
1	_comments	IMongoCollection<?>	
2	_posts	IMongoCollection<?>	
3	_authors	IMongoCollection<?>	
Operations			
No	Name	Return Type	Description
1	Delete	Bool	

Sequence Diagram

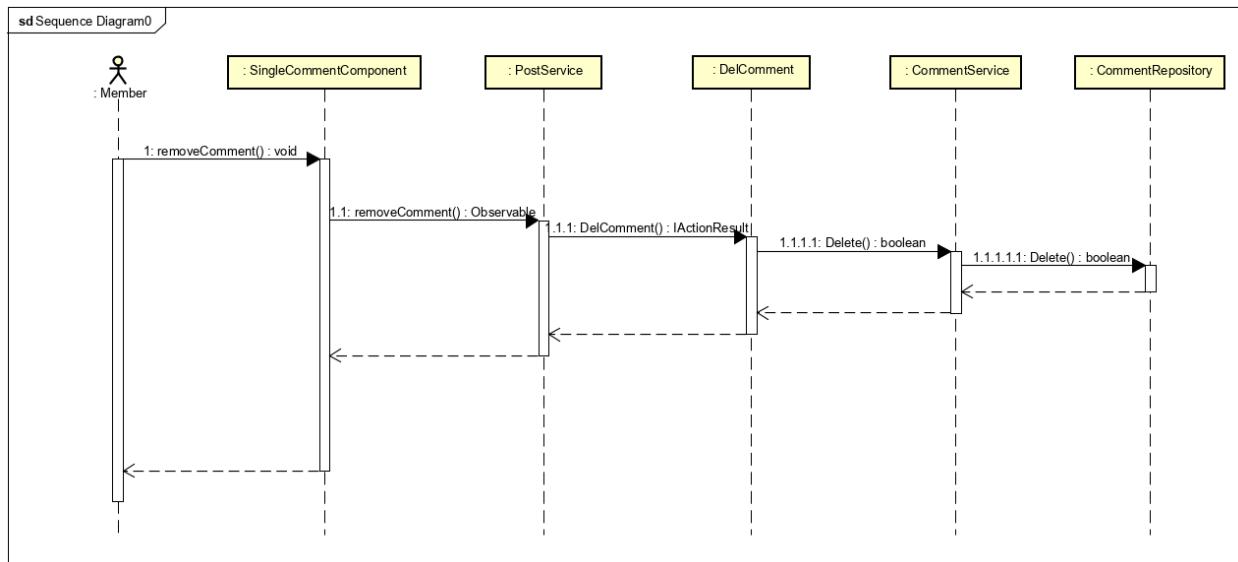


Figure 4-77: Delete A Comment Screen Design

4.3.4.22 Like/Unlike a comment

Screen design

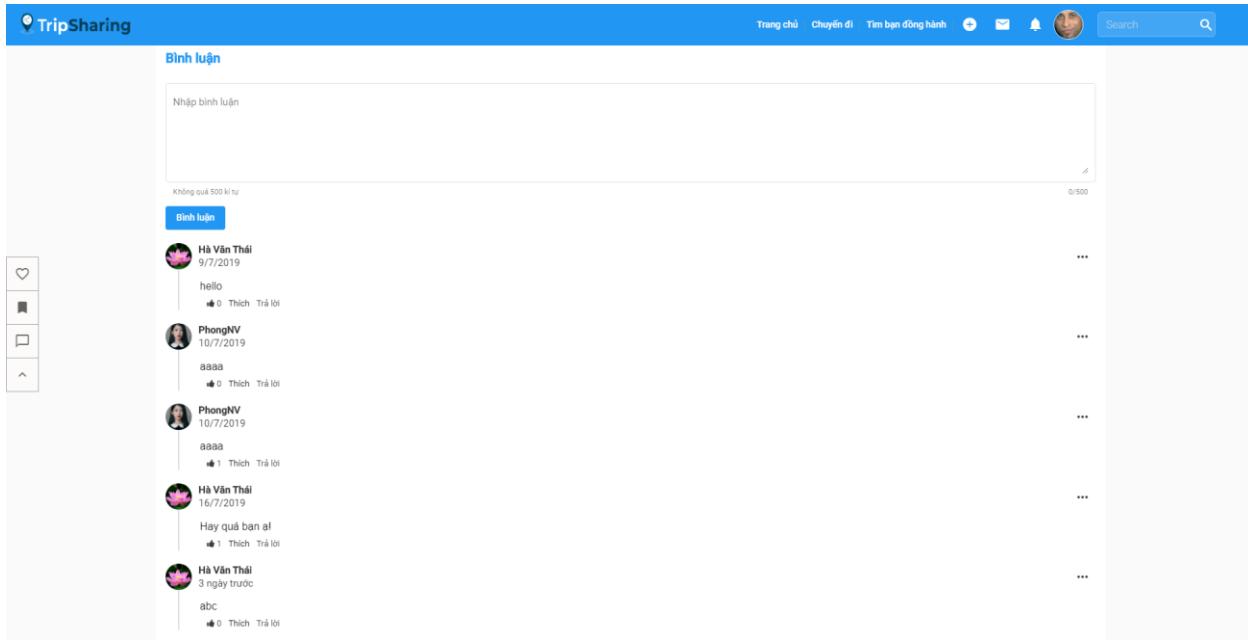


Figure 4-78: Like/Unlike a comment Screen Design

Class Diagram

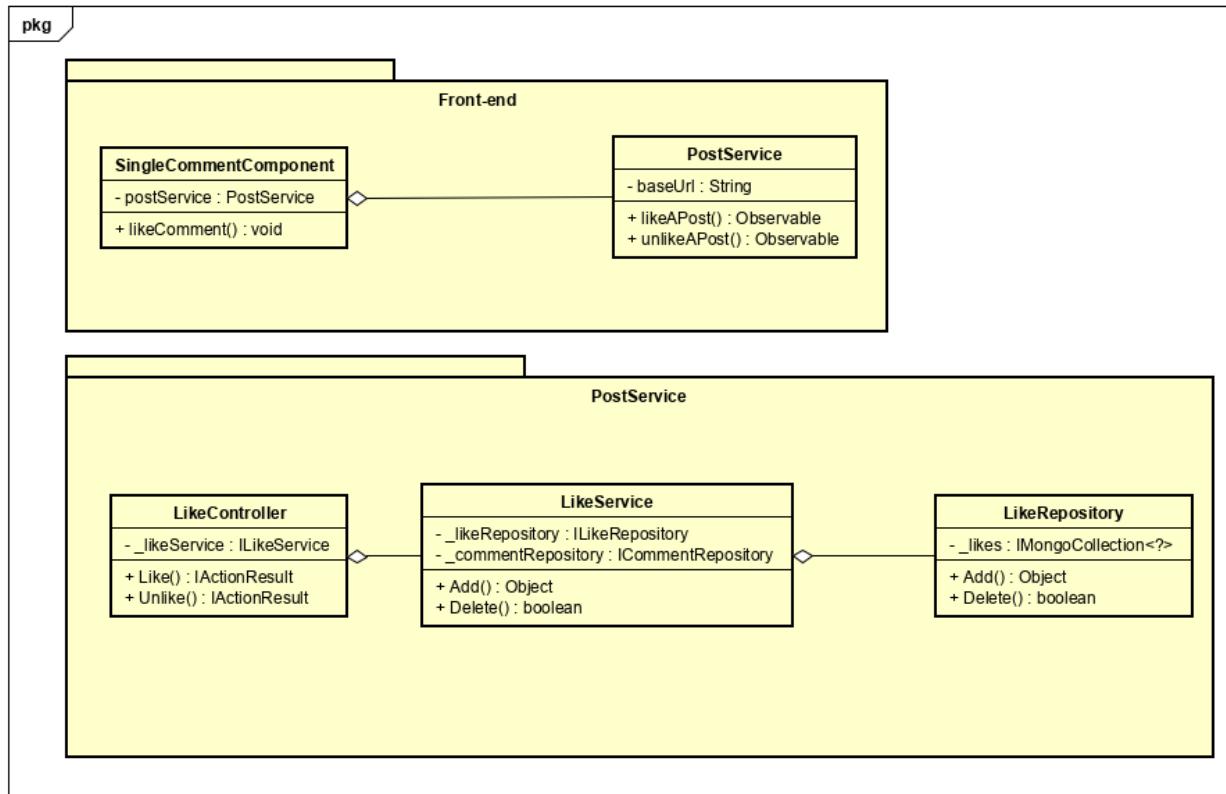


Figure 4-79: Like/Unlike a comment Class Diagram

Class Specification

SingleCommentComponent

SingleCommentComponent			
Physical address	src\app\shared\components\single-comment\single-comment.component.spec.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	postService	PostService	
Operations			
No	Name	Return Type	Description
1	likeComment	void	

PostService

PostService			
Physical address	src\app\core\services\post-service\virtual-trip.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	baseUrl	String	
Operations			
No	Name	Return Type	Description
1	likeAPost	Observable	
2	unlikeAPost	Observable	

LikeController

LikeController			
Physical address	src\Services\PostService\PostService\Controllers\LikeController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_likeService	ILikeService	
Operations			
No	Name	Return Type	Description
1	Like	IActionResult	
2	Unlike	IActionResult	

LikeService

LikeService			
Physical address	src\Services\PostService\PostService\Services\LikeService.cs		
Base Class	ILikeService		
Attributes			
No	Name	Type	Description
1	_likeRepository	ILikeRepository	
2	_commentRepository	ICommentRepository	

Operations			
No	Name	Return Type	Description
1	Add	Object	
2	Delete	Bool	

LikeRepository

LikeRepository			
Physical address	src\Services\PostService\PostService\Repositories\LikeRepository.cs		
Base Class	ILikeRepository		
Attributes			
No	Name	Type	Description
1	_likes	IMongoCollection<?>	
Operations			
No	Name	Return Type	Description
1	Add	Object	
2	Delete	Bool	

Sequence Diagram

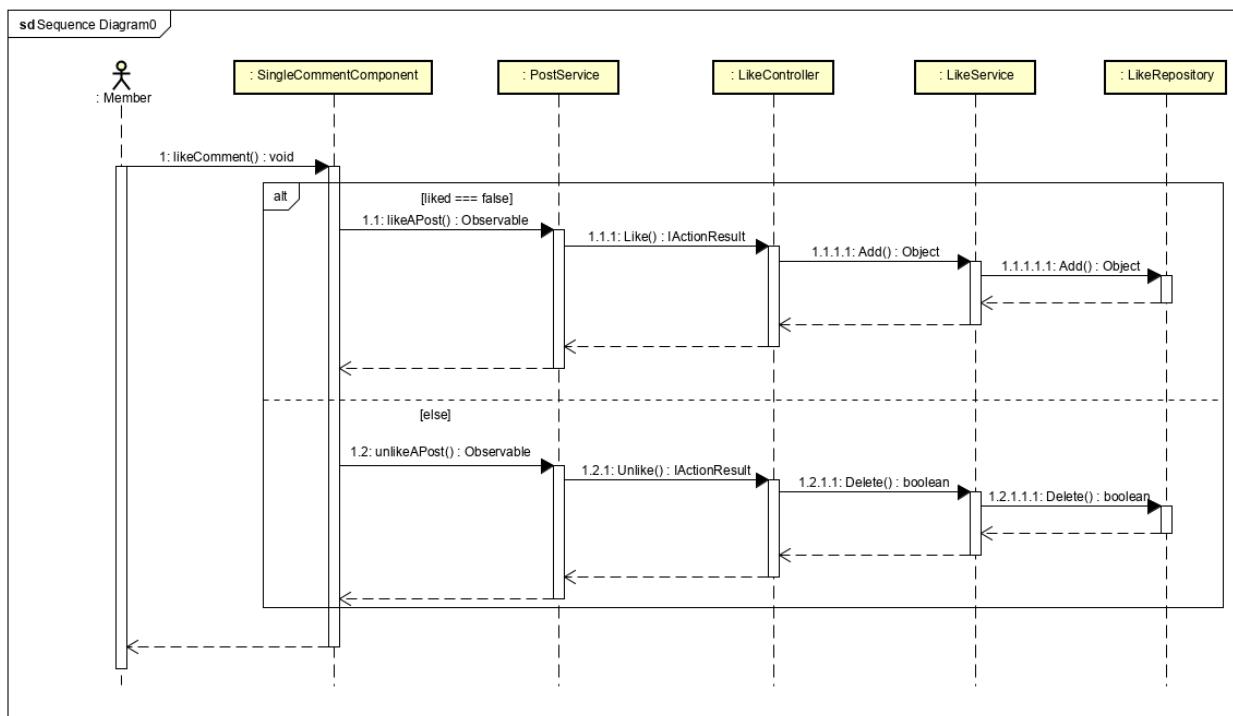


Figure 4-80: Like/Unlike a comment Sequence Diagram

4.3.4.23 Report a comment

Screen design

Báo cáo bình luận



Bạn có thể báo cáo nội dung sau khi chọn vấn đề.

Nội dung không phù hợp, phản cảm

Sao chép, đạo nhái của người khác

Nội dung sai sự thật

Lí do khác

Lý do khác:

Gửi

Figure 4-81: Report a comment Screen Design

Class Diagram

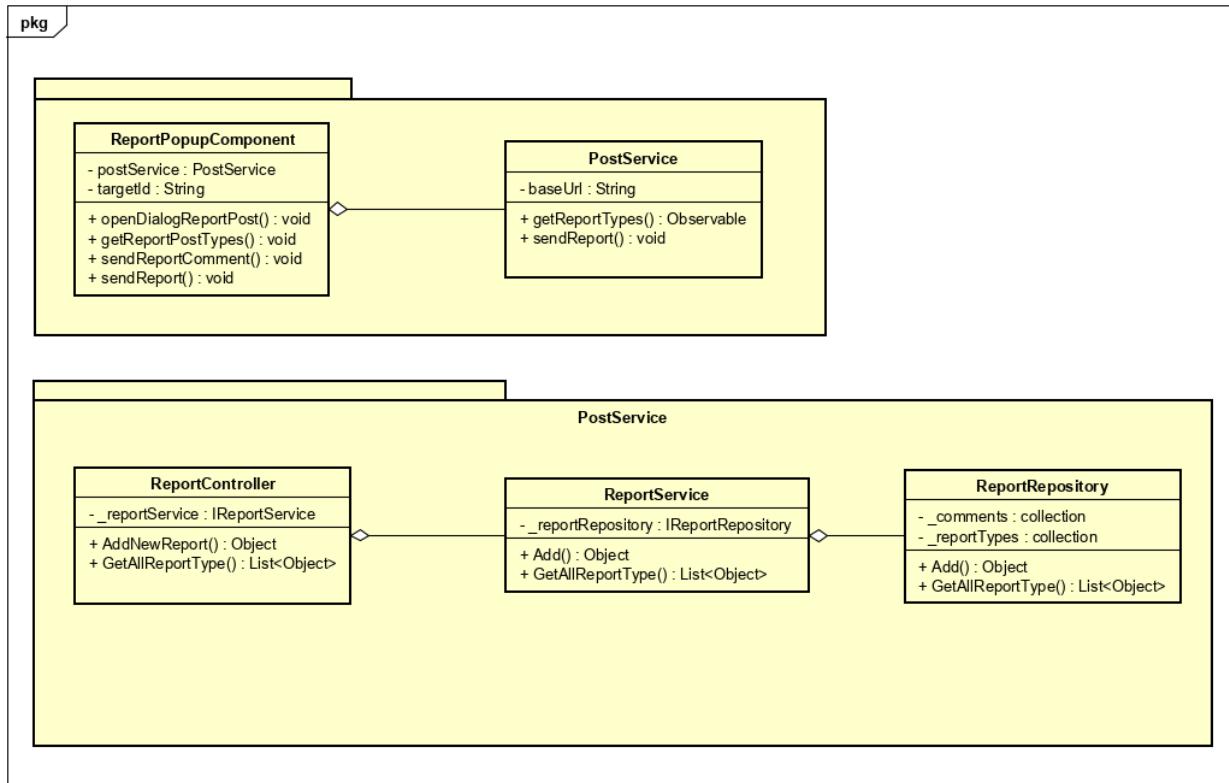


Figure 4-82: Report a comment Class Diagram

Class Specification

ReportPopupComponent

ReportPopupComponent			
Physical address	src\app\shared\components\report-popup\ report-popup.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	postService	PostService	
2	targetId	String	Specify object id of reported comment
Operations			
No	Name	Return Type	Description
1	openDialogReportPost	void	Show the report dialog
2	getReportPostTypes	observable	Get the report type from database
13	sendReportComment	void	Send report with type = comment
4	sendReport	void	Send report

PostService

PostService			
Physical address	src\app\core\services\post-service\virtual-trip.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	baseUrl	String	
Operations			
No	Name	Return Type	Description
1	sendReport	observable	
2	getReportTypes	observable	

ReportController

ReportController			
Physical address	src\Services\PostService\PostService\Controllers\ReportController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_reportService	IReportService	
Operations			
No	Name	Return Type	Description
1	AddNewReport	object	
2	GetAllReportType	List<object>	

ReportService

ReportService			
Physical address	src\Services\PostService\PostService\Services\ReportService.cs		
Base Class	IReportService		
Attributes			
No	Name	Type	Description
1	_reportRepository	IReportRepository	
Operations			
No	Name	Return Type	Description
1	Add	Object	
2	GetAllReportType	List<Object>	

ReportRepository

ReportRepository			
Physical address	src\Services\PostService\PostService\Repositories\ReportRepository.cs		
Base Class	IReportRepository		
Attributes			

No	Name	Type	Description
1	_comments		
2	_reportTypes		
Operations			
No	Name	Return Type	Description
1	Add	Object	
2	GetAllReportType	List<Object>	

Sequence Diagram

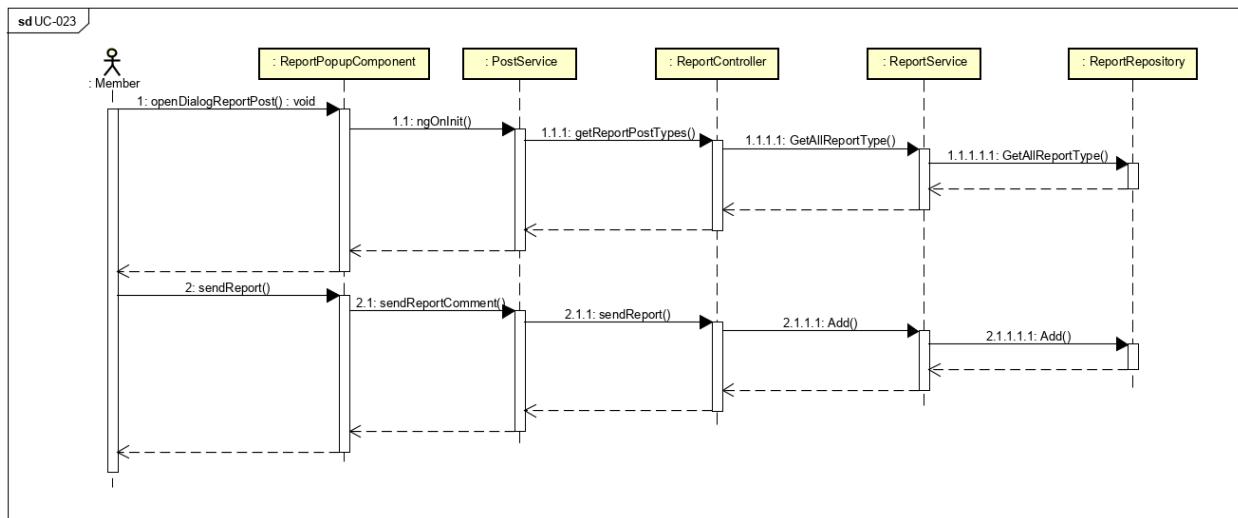


Figure 4-83: Report a comment Sequence Diagram

4.3.4.24 Follow a User

Screen design

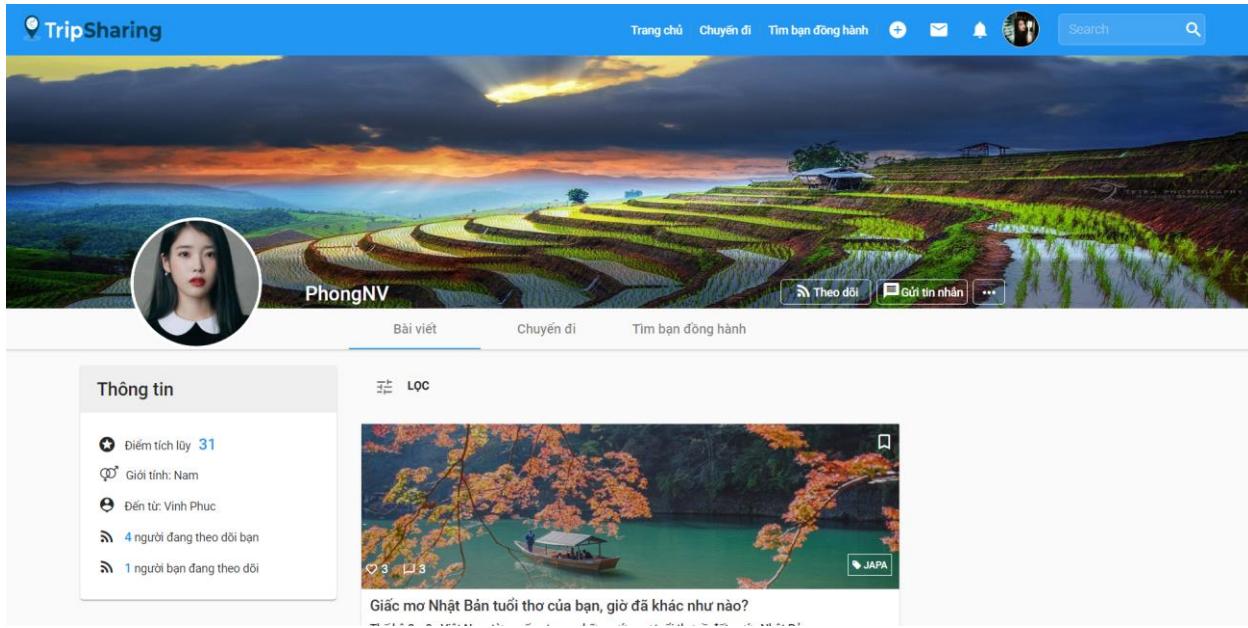


Figure 4-84: Follow a User Screen Design

Class Diagram

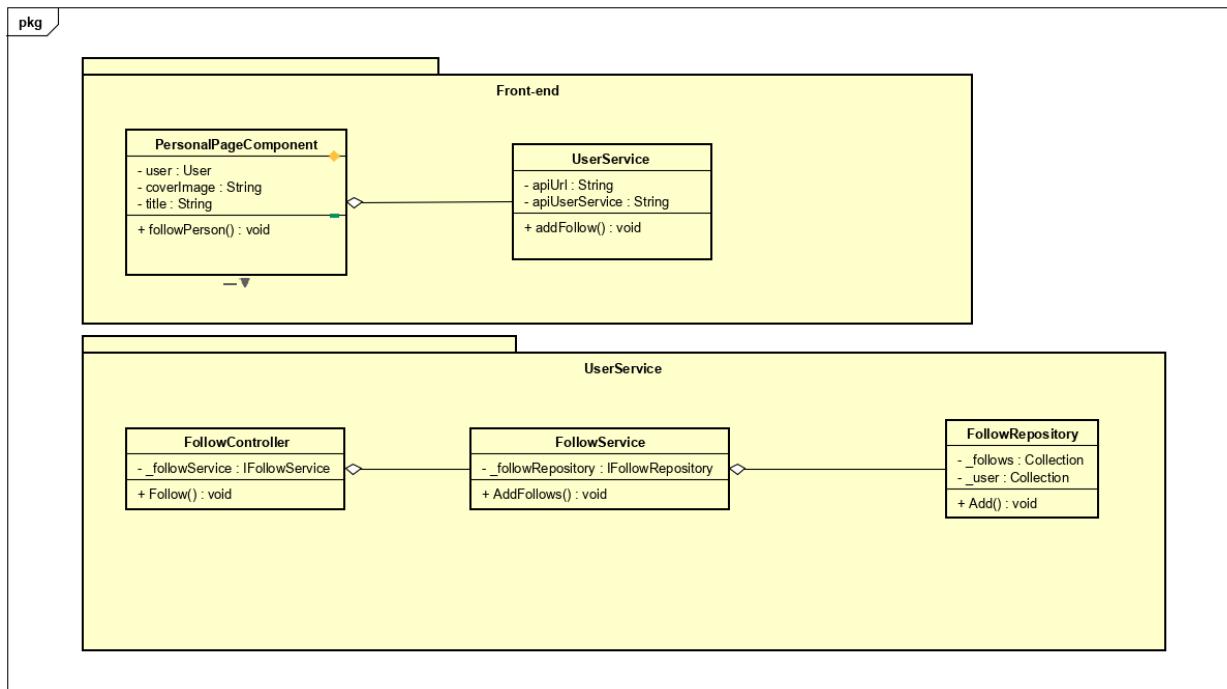


Figure 4-85: Follow a User Class Diagram

Class Specification

PersonalPageComponent

PersonalPageComponent

Physical address	src\app\pages\personal-page\personal-page.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	user	User	
2	coverImage	String	
3	title	string	
Operations			
No	Name	Return Type	Description
1	followPerson	void	

UserService

UserService			
Physical address	src\app\core\services\user-service\user.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	apiUrl	String	
2	apiUserService	String	
Operations			
No	Name	Return Type	Description
1	addFollow	observable	

FollowController

FollowController			
Physical address	src\Services\UserService\UserService\Controllers\FollowController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_followService	IFollowService	
Operations			
No	Name	Return Type	Description
1	Follow	object	

FollowService

FollowService			
Physical address	src\Services\UserService\UserService\Services\FollowService.cs		
Base Class	IFollowService		
Attributes			
No	Name	Type	Description

1	_followRepository	IFollowRepository	
Operations			
No	Name	Return Type	Description
1	AddFollows	Object	

FollowRepository

FollowRepository						
Physical address	src\Services\UserService\UserService\Repositories\ReportRepository.cs					
Base Class						
Operations						
No	Name	Return Type	Description			
1	Add	Object				

Sequence Diagram

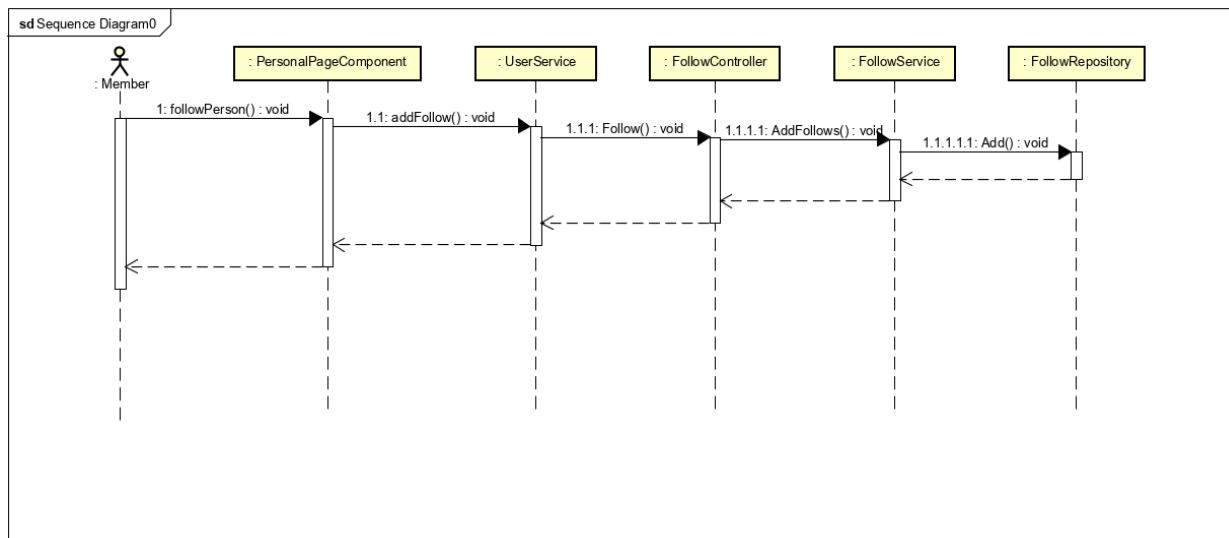


Figure 4-86: Follow a User Sequence Diagram

4.3.4.25 Unfollow a user

Screen design

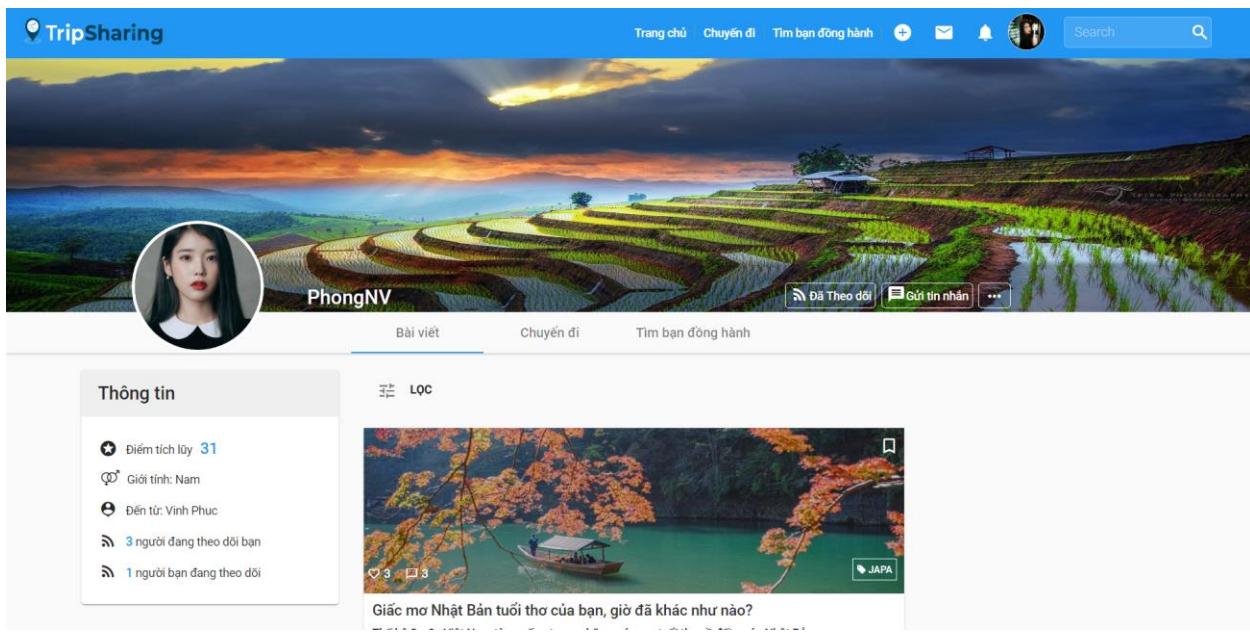


Figure 4-83: Unfollow a user Screen Design

Class Diagram

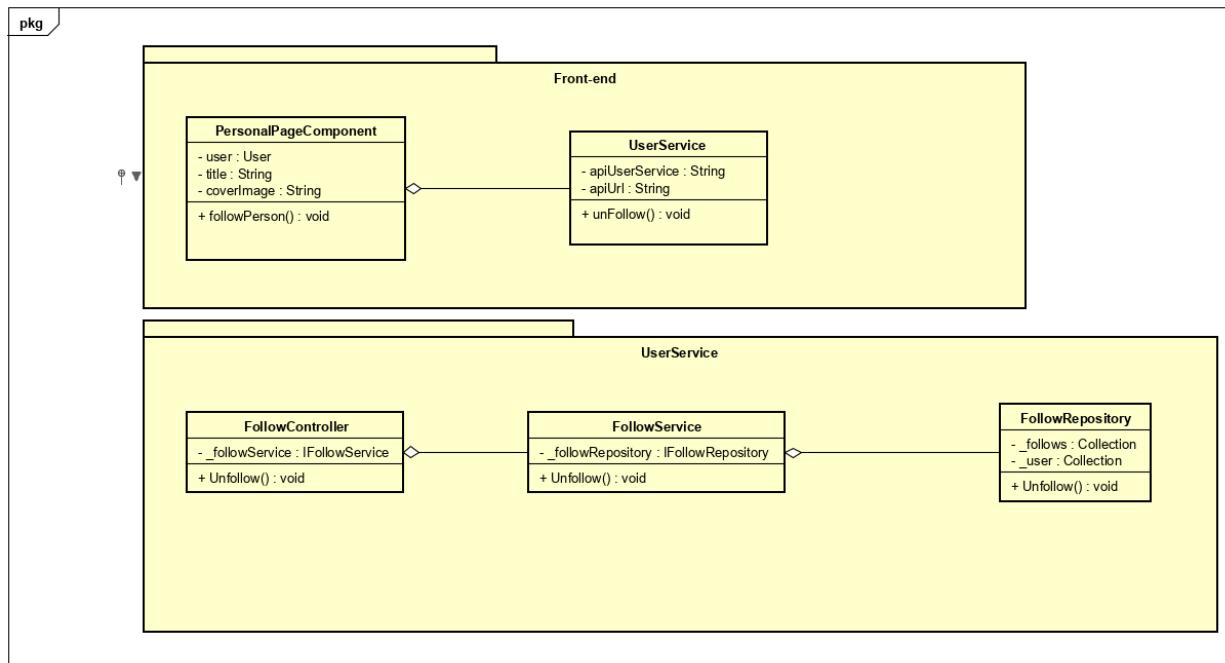


Figure 4-84: Unfollow a user Class Diagram

Class Specification

PersonalPageComponent

PersonalPageComponent	
Physical address	src\app\pages\personal-page\personal-page.component.ts

Base Class	Class		
Attributes			
No	Name	Type	Description
1	user	User	
2	coverImage	String	
3	title	string	
Operations			
No	Name	Return Type	Description
1	followPerson	void	

UserService

UserService			
Physical address	src\app\core\services\user-service\user.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	apiUrl	String	
2	apiUserService	String	
Operations			
No	Name	Return Type	Description
1	unFollow	Void	

FollowController

FollowController			
Physical address	src\Services\UserService\UserService\Controllers\FollowController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_followService	IFollowService	
Operations			
No	Name	Return Type	Description
1	UnFollow	void	

FollowService

FollowService			
Physical address	src\Services\UserService\UserService\Services\FollowService.cs		
Base Class	IFollowService		
Attributes			
No	Name	Type	Description
1	_followRepository	IFollowRepository	
Operations			
No	Name	Return Type	Description
1	UnFollow	Void	

FollowRepository

FollowRepository			
Physical address	src\Services\UserService\UserService\Repositories\ReportRepository.cs		
Base Class	IFollowRepository		
Attributes			
No	Name	Type	Description
1	_follows	Collection	
2	_user	Collection	
Operations			
No	Name	Return Type	Description
1	UnFollow	Void	

Sequence Diagram

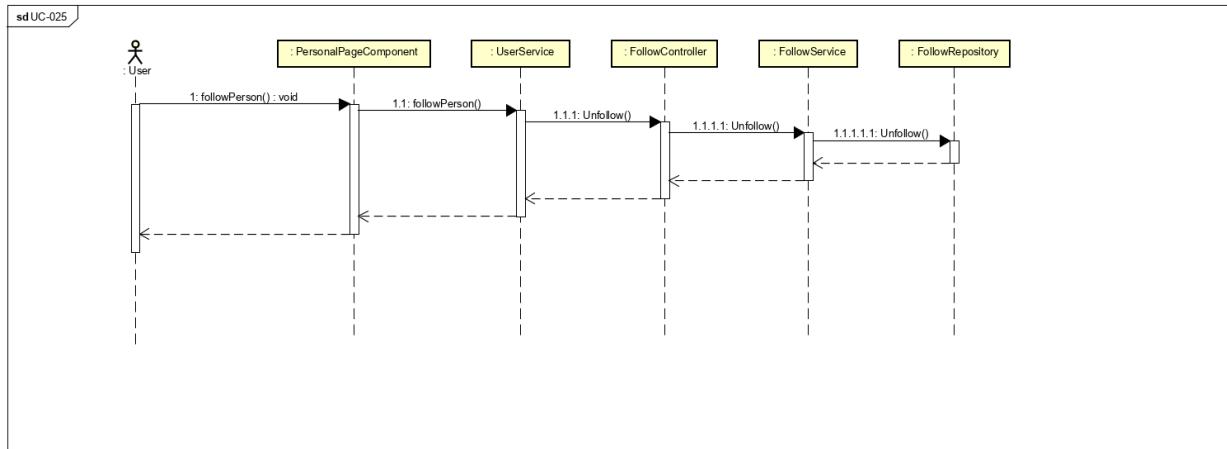


Figure 4-85: Unfollow a user Sequence Diagram

4.3.4.26 Report a user

Screen design

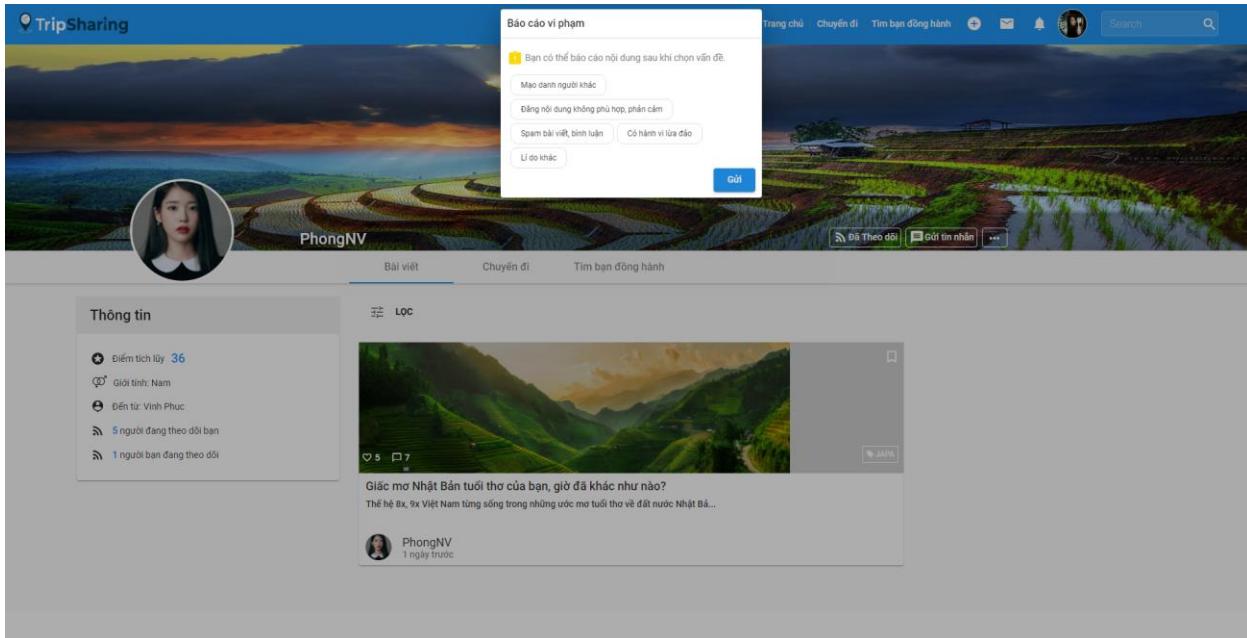


Figure 4-86: Report a user Screen Design

Class Diagram

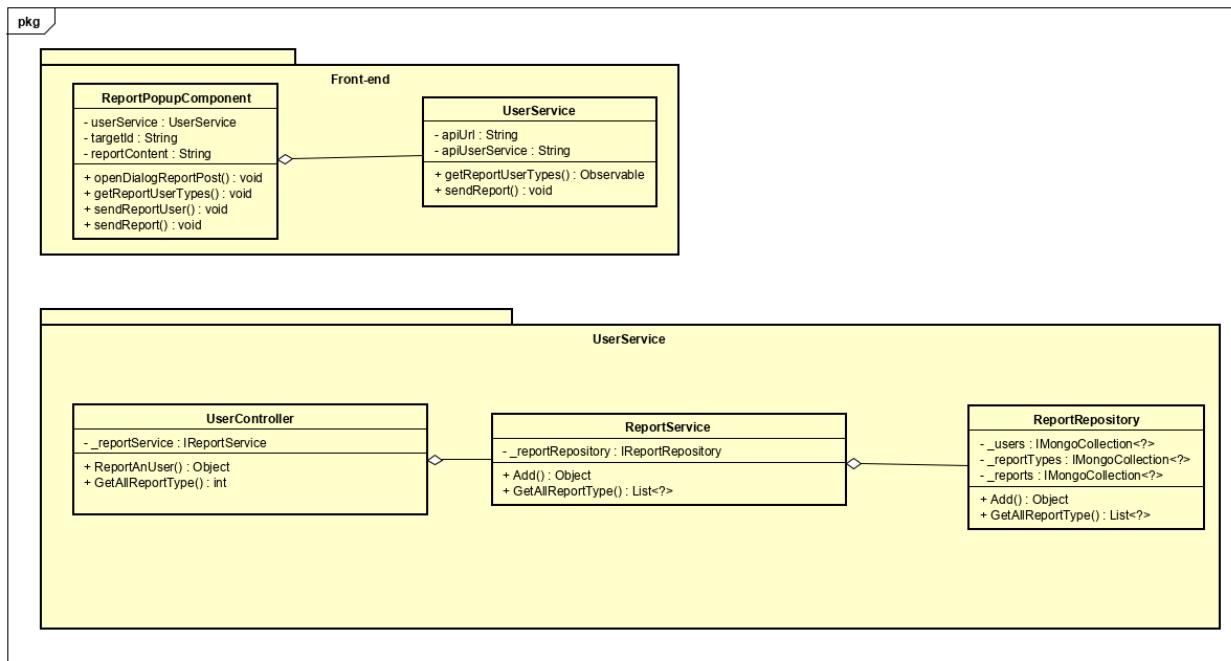


Figure 4-87: Report a user Class Diagram

Class Specification

ReportPopupComponent

ReportPopupComponent

Physical address	src\app\shared\components\report-popup\report-popup.component.ts
------------------	--

Base Class	Class		
Attributes			
No	Name	Type	Description
1	userService	UserService	
2	targetId	String	
3	reportContent	String	
Operations			
No	Name	Return Type	Description
1	openDialogReportPost	Void	
2	getReportUserTypes	Void	
3	sendReportUser	Void	
4	sendReport	Void	

UserService

UserService			
Physical address	src\app\core\services\user-service\user.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	apiUrl	String	
2	apiUserService	String	
Operations			
No	Name	Return Type	Description
1	getReportUserTypes	Observable	
2	sendReport	Void	

UserController

UserController			
Physical address	src\Services\UserService\UserService\Controllers\UserController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_reportService	IReportService	
Operations			
No	Name	Return Type	Description
1	Add	Object	

ReportService

ReportService			
Physical address	src\Services\UserService\UserService\Services\ReportService.cs		

Base Class	IReportService		
Attributes			
No	Name	Type	Description
1	_reportRepository	IReportRepository	
Operations			
No	Name	Return Type	Description
1	Add	Object	
2	GetAllReportType	List<?>	

ReportRepository

ReportRepository			
Physical address	src\Services\UserService\UserService\Repositories\ReportRepository.cs		
Base Class	IReportRepository		
Attributes			
No	Name	Type	Description
1	_reportTypes	IMongoCollection<?>	
2	_users	IMongoCollection<?>	
3	_reports	IMongoCollection<?>	
Operations			
No	Name	Return Type	Description
1	Add	Object	
2	GetAllReportType	List<?>	

Sequence Diagram

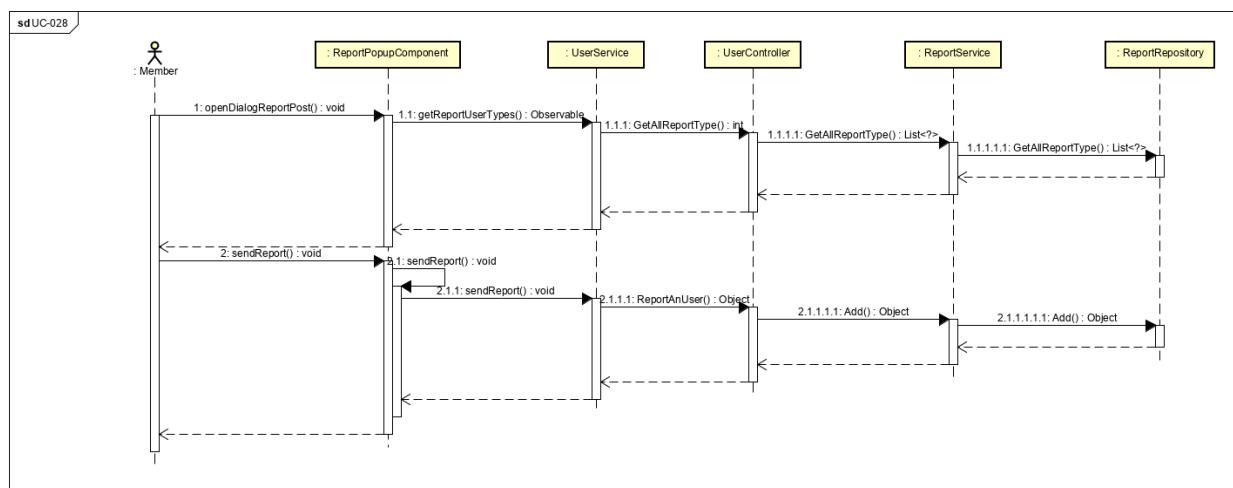


Figure 4-88: Report a user Sequence Diagram

4.3.4.27 Join to a companion group

Screen design

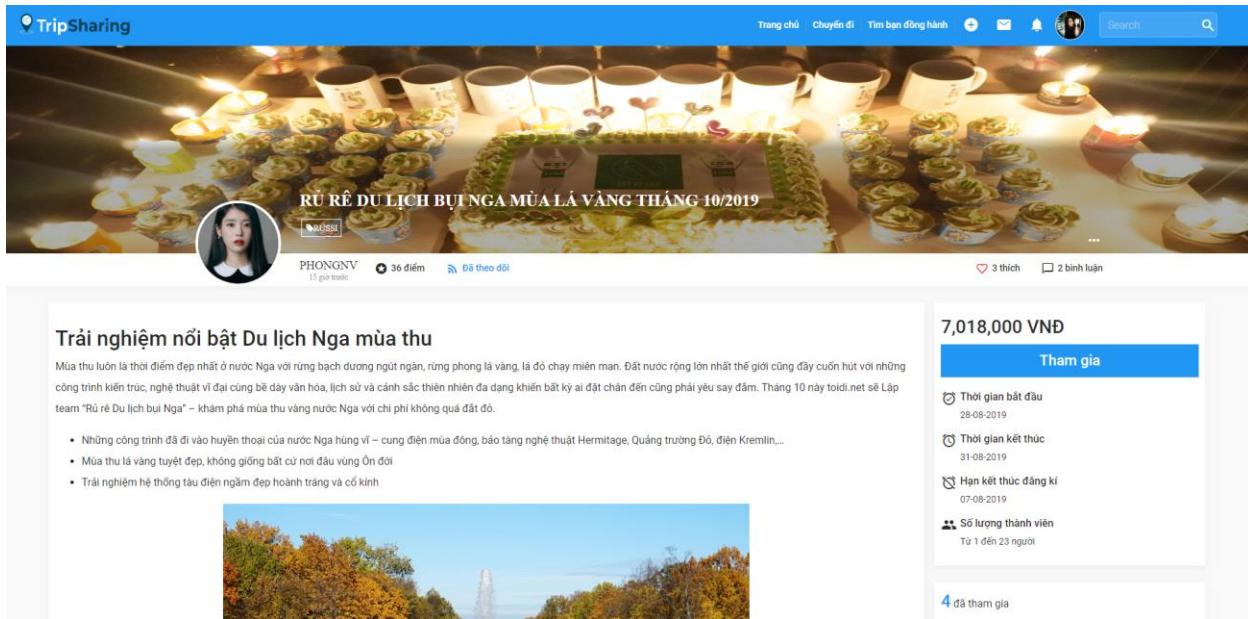


Figure 4-89: Join to a companion group Screen Design

Class Diagram

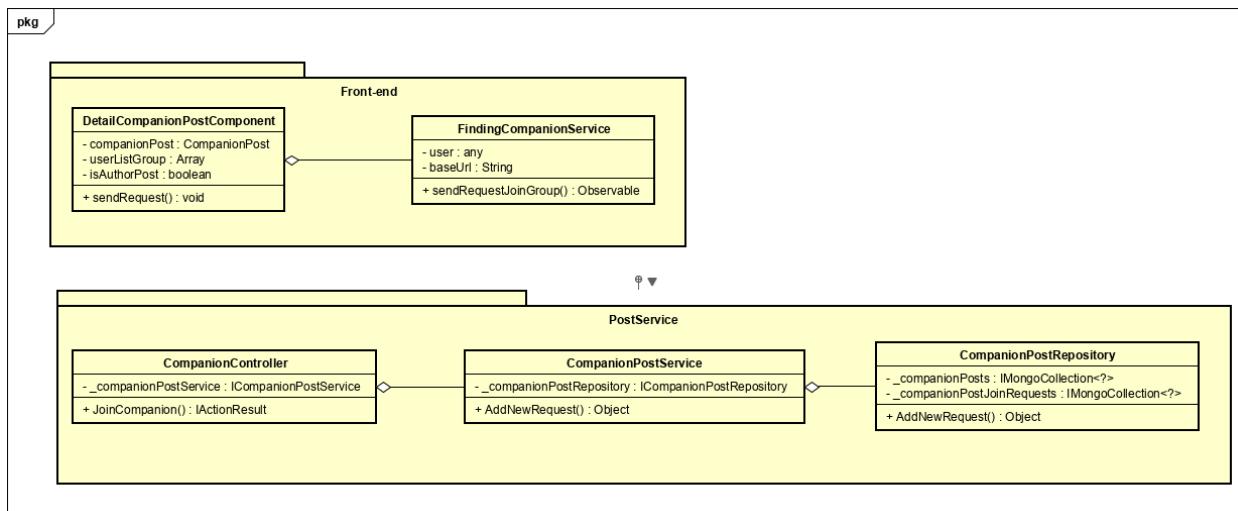


Figure 4-90: Join to a companion group Class Diagram

Class Specification

DetailCompanionPostComponent

DetailCompanionPostComponent			
Attributes	Operations		
Physical address	src\app\shared\components\detail-companion-post\detail-companion-post.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description

1	companionPost	CompanionPost	
2	userListGroup	Array	
3	isAuthorPost	boolean	
Operations			
No	Name	Return Type	Description
1	sendRequest	Void	

FindingCompanionService

FindingCompanionService			
Physical address	src\app\core\services\post-service\finding-companion.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	apiUrl	String	
2	apiUserService	String	
Operations			
No	Name	Return Type	Description
1	getReportUserTypes	Observable	
2	sendReport	Void	

CompanionController

CompanionController			
Physical address	src\Services\PostService\PostService\Controllers\CompanionController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_companionPostService	ICompanionPostService	
Operations			
No	Name	Return Type	Description
1	JoinCompanion	IActionResult	

CompanionPostService

CompanionPostService			
Physical address	src\Services\PostService\PostService\Services\CompanionPostService.cs		
Base Class	ICompanionPostService		
Attributes			
No	Name	Type	Description
1	_companionPostRepository	ICompanionPostRepository	
Operations			
No	Name	Return Type	Description

1	AddNewRequest	Object	
---	---------------	--------	--

CompanionPostRepository

ReportRepository			
Physical address	src\Services\PostService\PostService\Repositories\CompanionPostRepository.cs		
Base Class	ICompanionPostRepository		
Attributes			
No	Name	Type	Description
1	_companionPostJoinRequests	IMongoCollection<?>	
Operations			
No	Name	Return Type	Description
1	AddNewRequest	Object	

Sequence Diagram

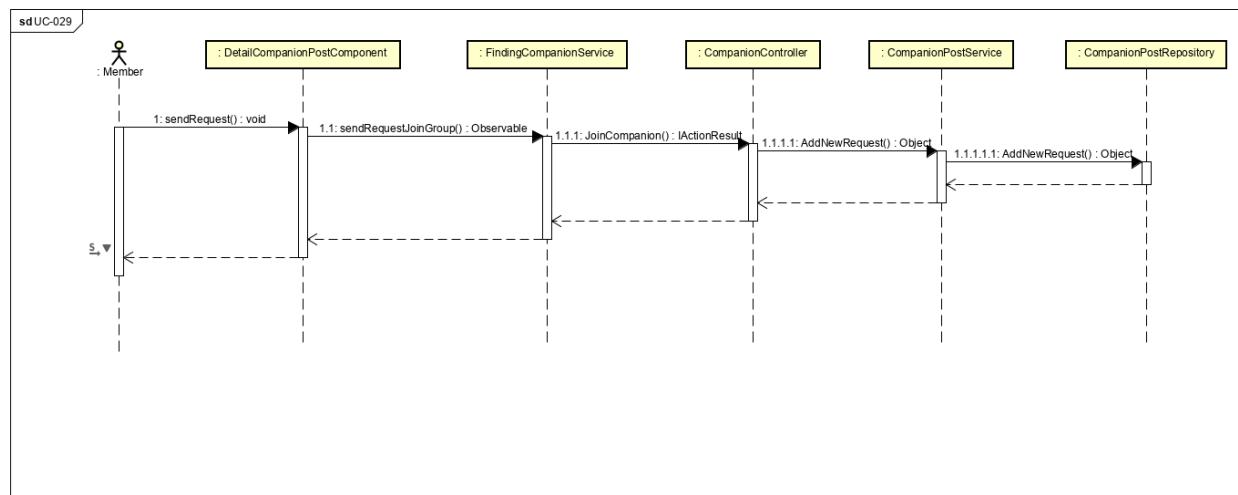


Figure 4-91: Join to a companion group Sequence Diagram

4.3.4.28 Leave a companion group

Screen design

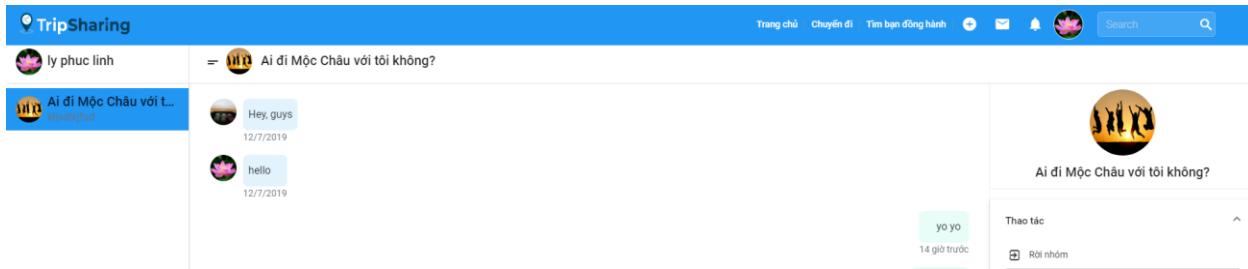


Figure 4-92: Leave a companion group Screen Design

Class Diagram

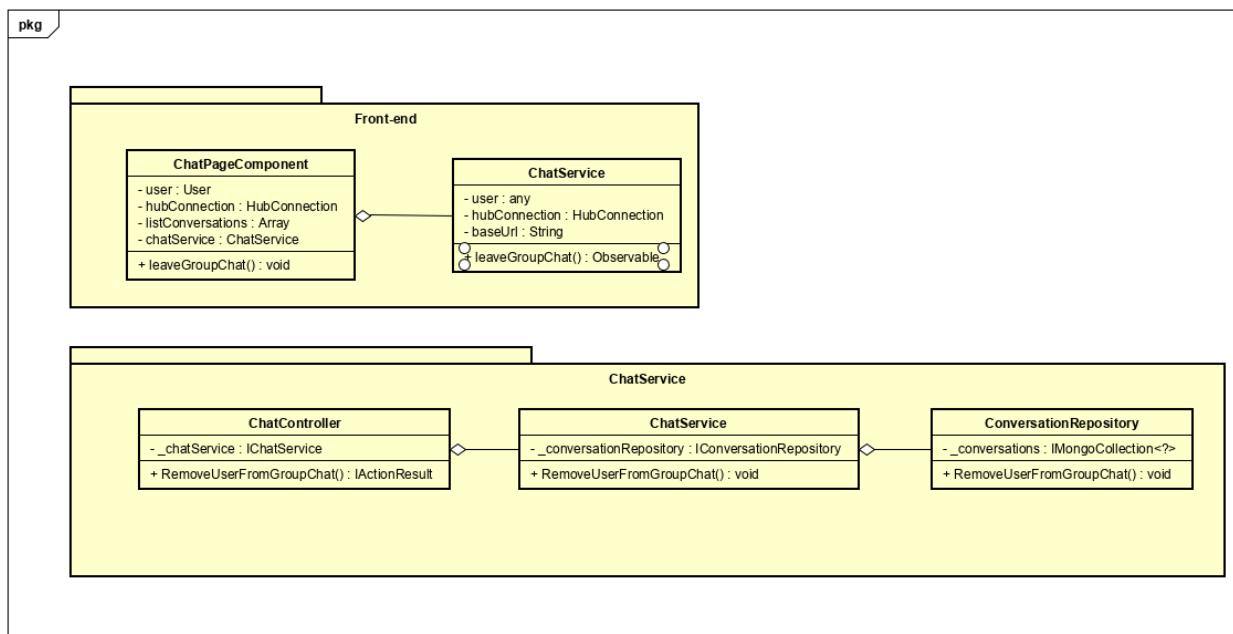


Figure 4-93: Leave a companion group Class Diagram

Class Specification

ChatPageComponent

ChatPageComponent			
Physical address	src\app\pages\chat-page\chat-page.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	user	User	
2	hubConnection	HubConnection	
3	listConversation	Array	
4	chatService	ChatService	
Operations			
No	Name	Return Type	Description

1	leaveGroupChat	Void	
---	----------------	------	--

ChatService

ChatService			
Physical address	src\app\core\services\chat-service\chat.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	user	Any	
2	hubConnection	HubConnection	
3	baseUrl	String	
Operations			
No	Name	Return Type	Description
1	leaveGroupChat	Observable	

ChatController

ChatController			
Physical address	src\Services\ChatService\ChatService\Controllers\ChatController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_chatService	IChatService	
Operations			
No	Name	Return Type	Description
1	RemoveUserFromGroupChat	IActionResult	

ChatService

ChatService			
Physical address	src\Services\PostService\PostService\Services\CompanionPostService.cs		
Base Class	IChatService		
Attributes			
No	Name	Type	Description
1	_conversationRepository	IConversationRepository	
Operations			
No	Name	Return Type	Description
1	RemoveUserFromGroupChat	Bool	

ConversationRepository

ConversationRepository

Physical address	src\Services\ChatService\ChatService\Repositories\ConversationRepository.cs		
Base Class	IConversationRepository		
Attributes			
No	Name	Type	Description
1	_conversations	IMongoCollection<?>	
Operations			
No	Name	Return Type	Description
1	RemoveUserFromGroupChat	Bool	

Sequence Diagram

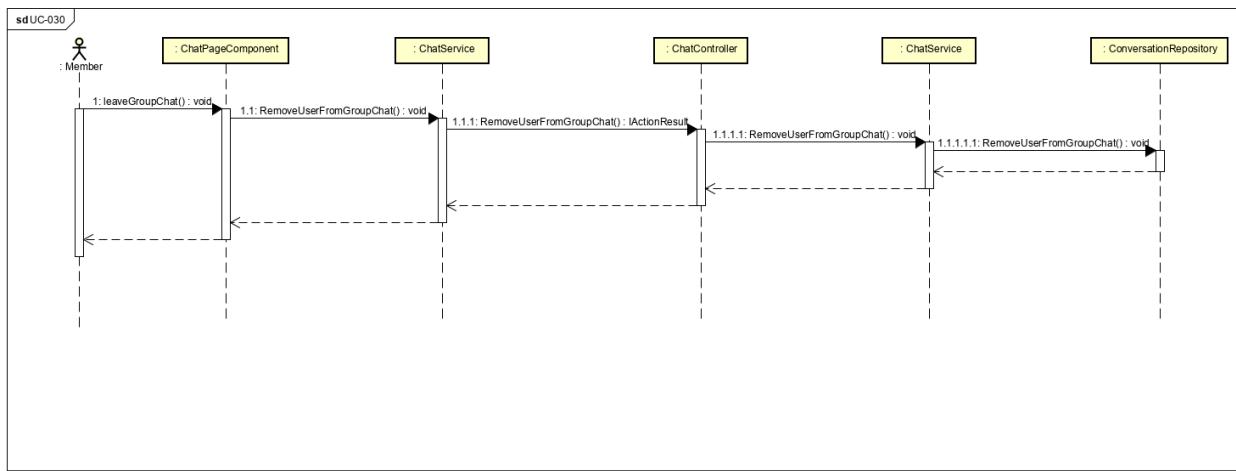


Figure 4-94: Leave a companion group Sequence Diagram

4.3.4.29 Remove a member

Screen design

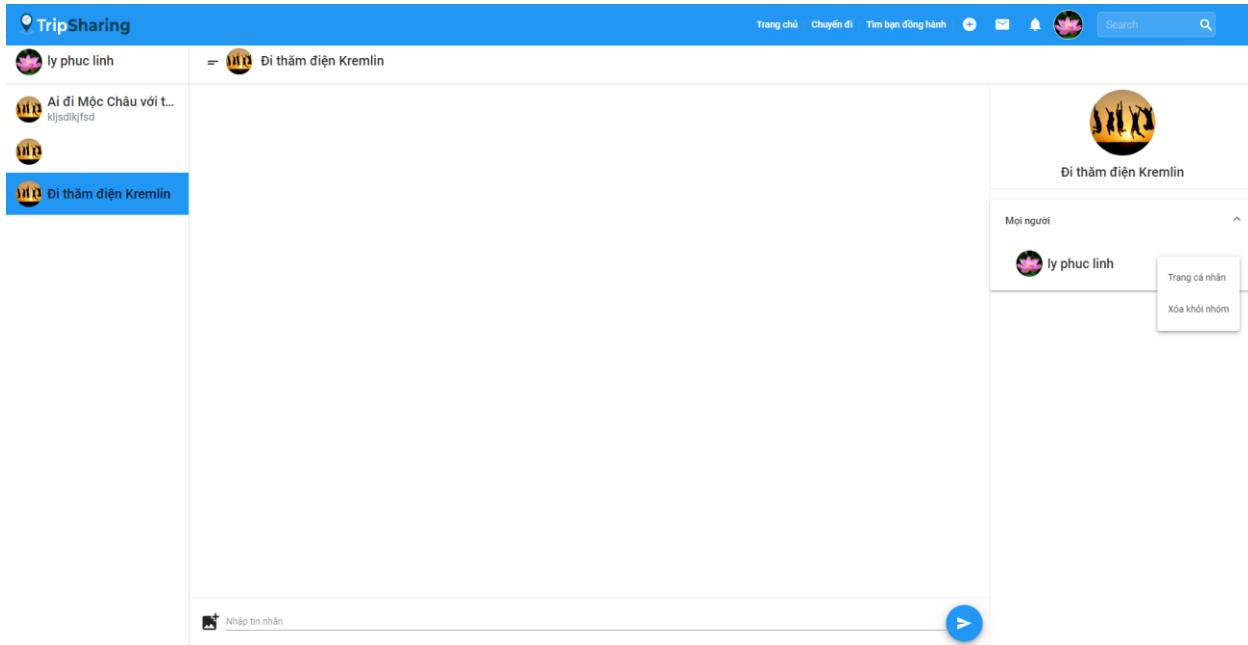


Figure 4-95: Remove a member Screen Design

Class Diagram

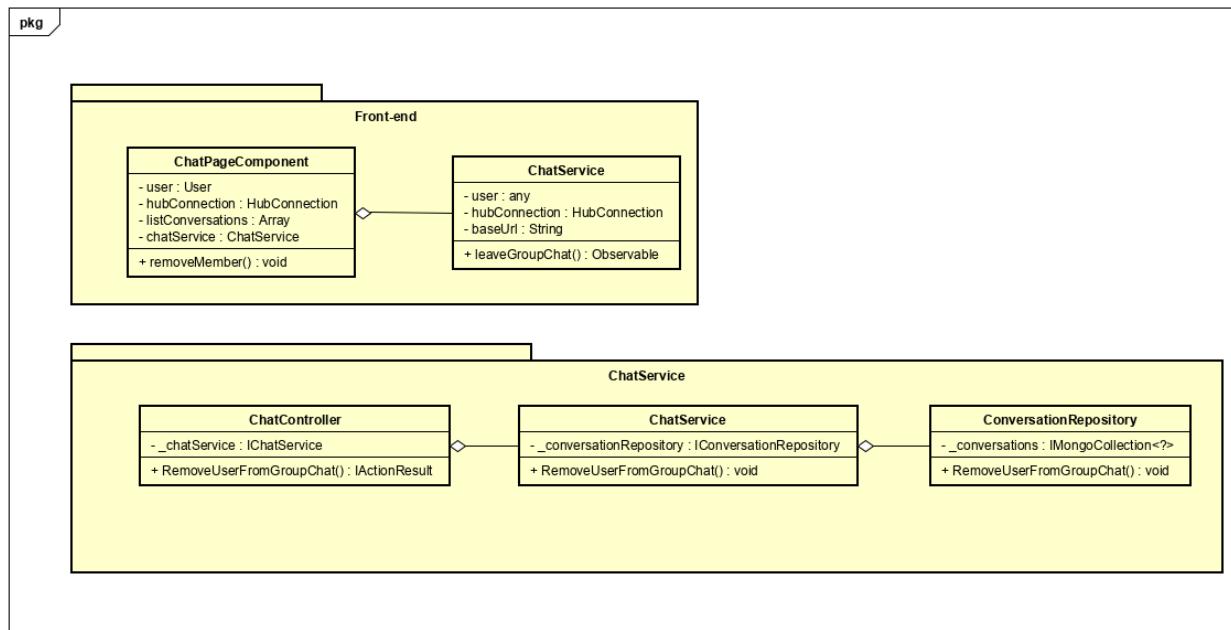


Figure 4-96: Remove a member Class Diagram

Class Specification

ChatPageComponent

ChatPageComponent	
Physical address	src\app\pages\chat-page\chat-page.component.ts

Base Class	Class		
Attributes			
No	Name	Type	Description
1	user	User	
2	hubConnection	HubConnection	
3	listConversation	Array	
4	chatService	ChatService	
Operations			
No	Name	Return Type	Description
1	removeMember	Void	

ChatService

ChatService			
Physical address	src\app\core\services\chat-service\chat.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	user	Any	
2	hubConnection	HubConnection	
3	baseUrl	String	
Operations			
No	Name	Return Type	Description
1	leaveGroupChat	Observable	

ChatController

ChatController			
Physical address	src\Services\ChatService\ChatService\Controllers\ChatController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_chatService	IChatService	
Operations			
No	Name	Return Type	Description
1	RemoveUserFromGroupChat	IActionResult	

ChatService

ChatService			
Physical address	src\Services\PostService\PostService\Services\CompanionPostService.cs		
Base Class	IChatService		
Attributes			

No	Name	Type	Description
1	_conversationRepository	IConversationRepository	
Operations			
No	Name	Return Type	Description
1	RemoveUserFromGroupChat	Bool	

ConversationRepository

ConversationRepository			
Physical address	src\Services\ChatService\ChatService\Repositories\ConversationRepository.cs		
Base Class	IConversationRepository		
Attributes			
No	Name	Type	Description
1	_conversations	IMongoCollection<?>	
Operations			
No	Name	Return Type	Description
1	RemoveUserFromGroupChat	Bool	

Sequence Diagram

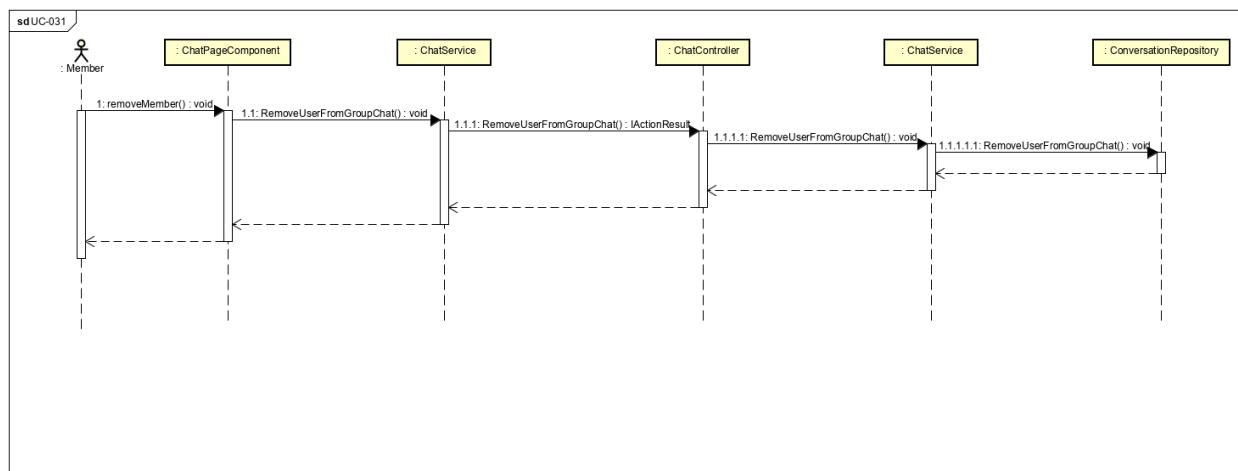


Figure 4-97: Remove a member Sequence Diagram

4.3.4.30 Send/receive messages

Screen design

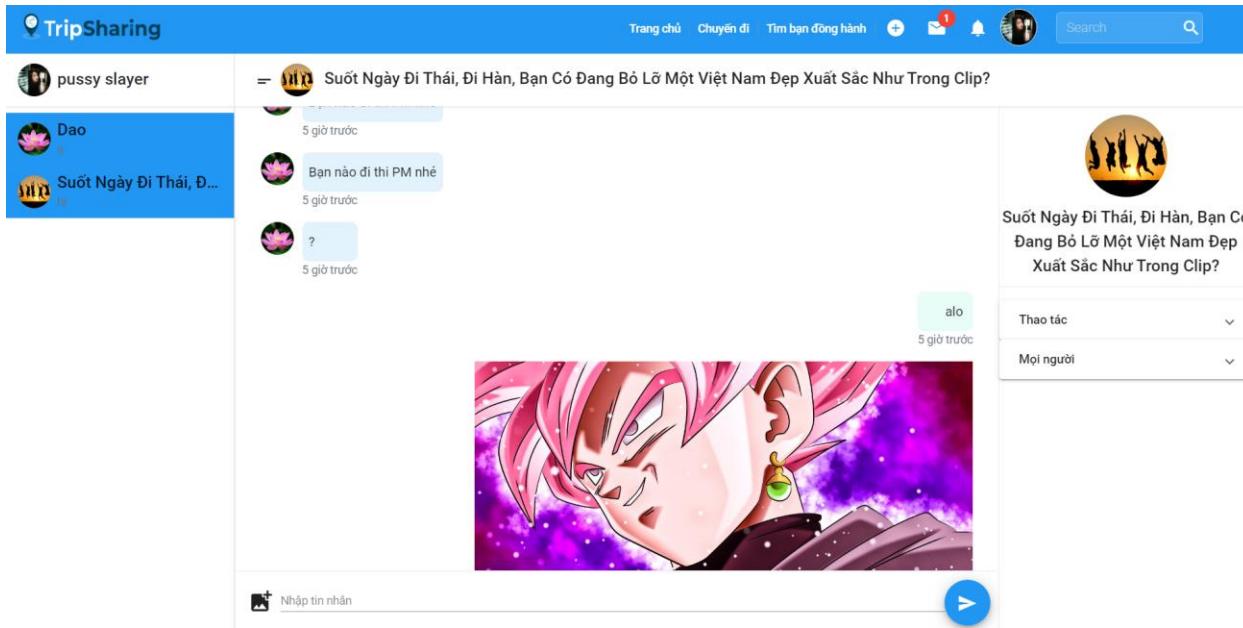


Figure 4-98: Send/receive messages Screen Design

Class Diagram

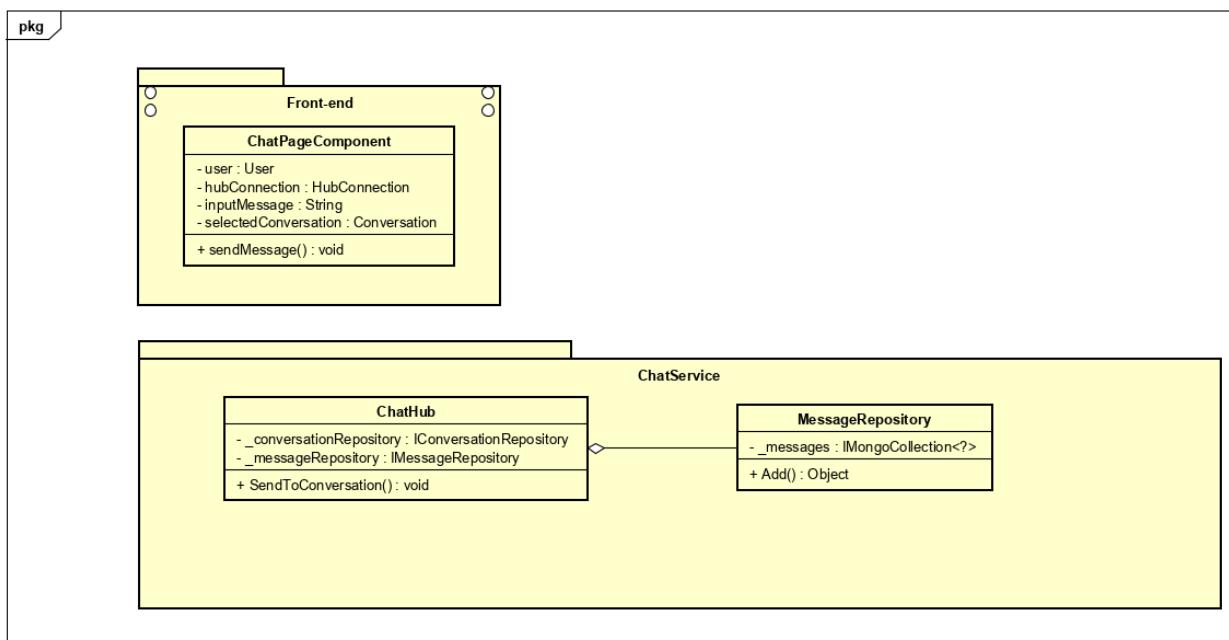


Figure 4-99: Send/receive messages Class Diagram

Class Specification

ChatPageComponent

ChatPageComponent	
Physical address	src\app\pages\chat-page\chat-page.component.ts
Base Class	Class

Attributes			
No	Name	Type	Description
1	user	User	
2	hubConnection	HubConnection	
3	listConversation	Array	
4	chatService	ChatService	
Operations			
No	Name	Return Type	Description
1	sendMessage	Void	

ChatHub

ChatHub			
Physical address	src\Services\ChatService\ChatService\HubConfig\ChatHub.cs		
Base Class	Hub		
Attributes			
No	Name	Type	Description
1	_messageRepository	IMessageRepository	
Operations			
No	Name	Return Type	Description
1	SendToConversation	Void	

MessageRepository

MessageRepository			
Physical address	src\Services\ChatService\ChatService\Repositories\MessageRepository.cs		
Base Class	IConversationRepository		
Attributes			
No	Name	Type	Description
1	_messages	IMongoCollection<?>	
Operations			
No	Name	Return Type	Description
1	Add	Object	

Sequence Diagram

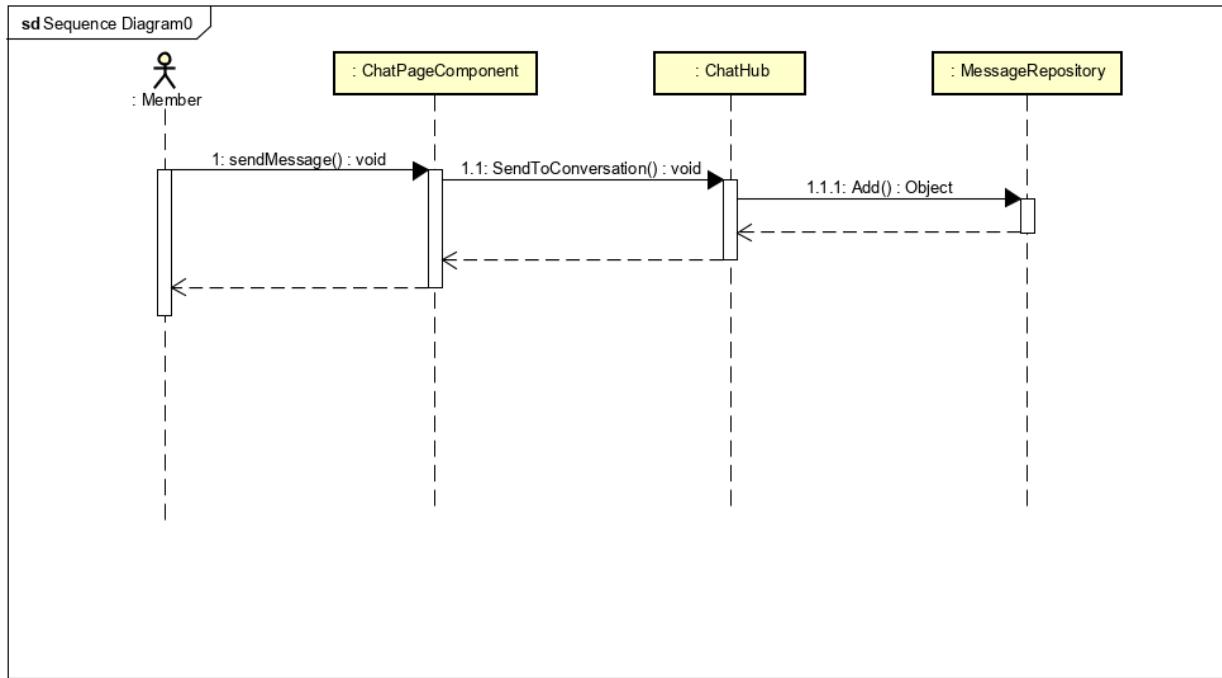


Figure 4-100: Send/receive messages Sequence Diagram

4.3.4.31 View recommended articles

Screen design

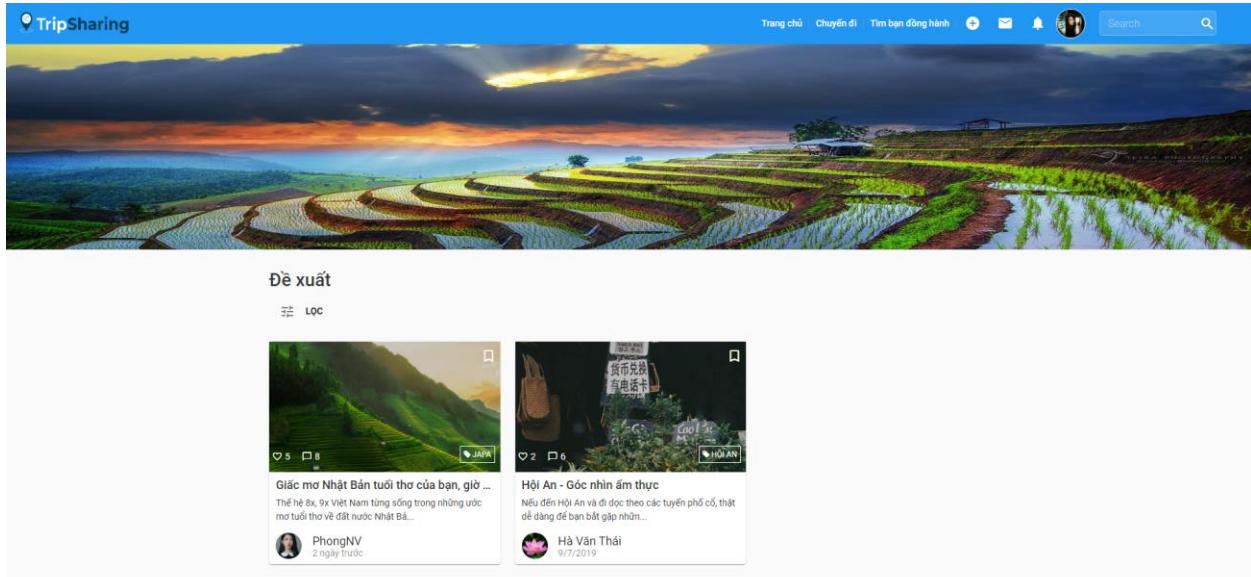


Figure 4-101: View recommended articles Screen Design

Class Diagram

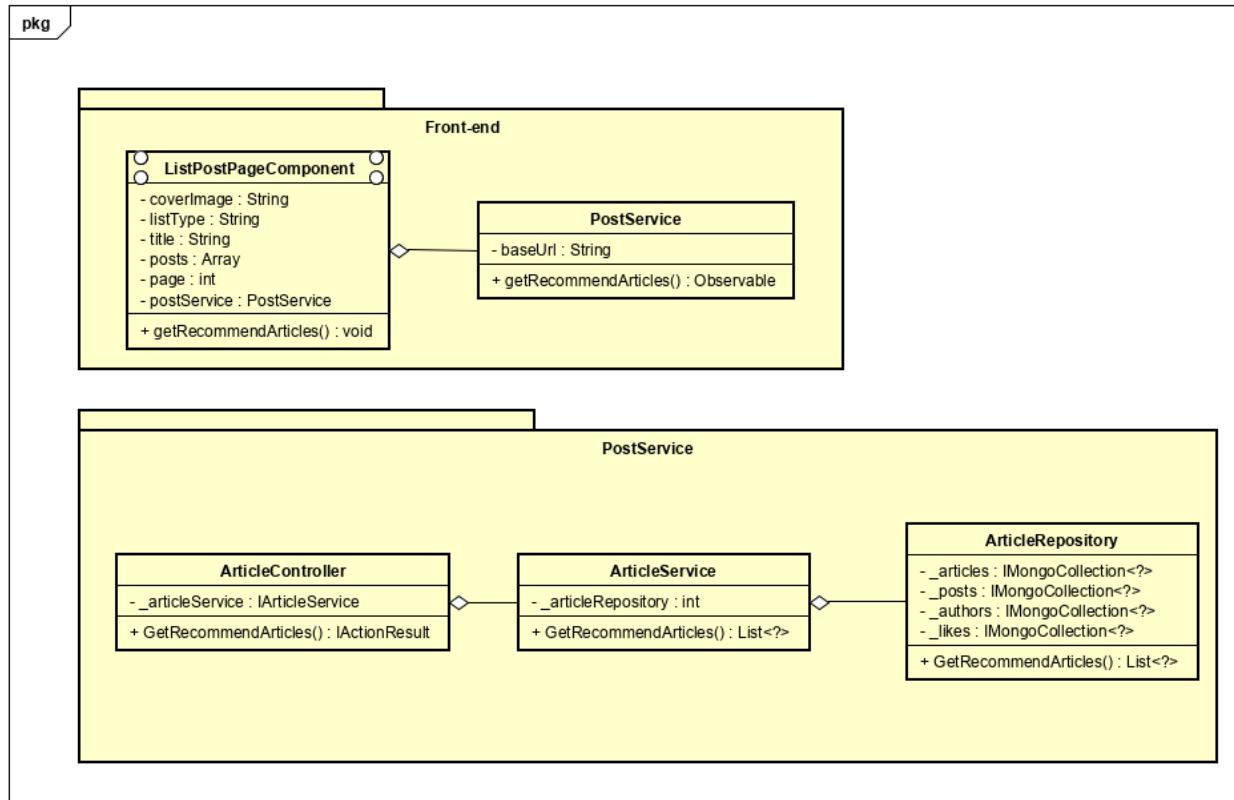


Figure 4-102: View recommended articles Class Diagram

Class Specification

ListPostPageComponent

ListPostPageComponent			
Physical address	src\app\pages\list-post-page\list-post-page.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	coverImage	String	
2	listType	String	
3	title	String	
4	posts	Array	
5	page	Int	
6	postService	PostService	
Operations			
No	Name	Return Type	Description
1	getRecommendArticles	Void	

PostService

PostService			
Physical address	src\app\core\services\post-service\post.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	baseUrl	String	
Operations			
No	Name	Return Type	Description
1	getRecommendArticles	Observable	

ArticleController

ArticleController			
Physical address	src\Services\PostService\PostService\Controllers\ArticleController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_articleService	IArticleService	
Operations			
No	Name	Return Type	Description
1	GetRecommendArticles	IActionResult	

ChatService

ArticleService			
Physical address	src\Services\PostService\PostService\Services\ArticleService.cs		
Base Class	IArticleService		
Attributes			
No	Name	Type	Description
1	_articleRepository	IArticleRepository	
Operations			
No	Name	Return Type	Description
1	GetRecommendArticles	List<?>	

ArticleRepository

ArticleRepository			
Physical address	src\Services\PostService\PostService\Repositories\ArticleRepository.cs		
Base Class	IArticleRepository		
Attributes			
No	Name	Type	Description
1	_articles	IMongoCollection<?>	
2	_posts	IMongoCollection<?>	

3	_authors	IMongoCollection<?>	
4	_likes	IMongoCollection<?>	
Operations			
No	Name	Return Type	Description
1	GetRecommendArticles	List<?>	

Sequence Diagram

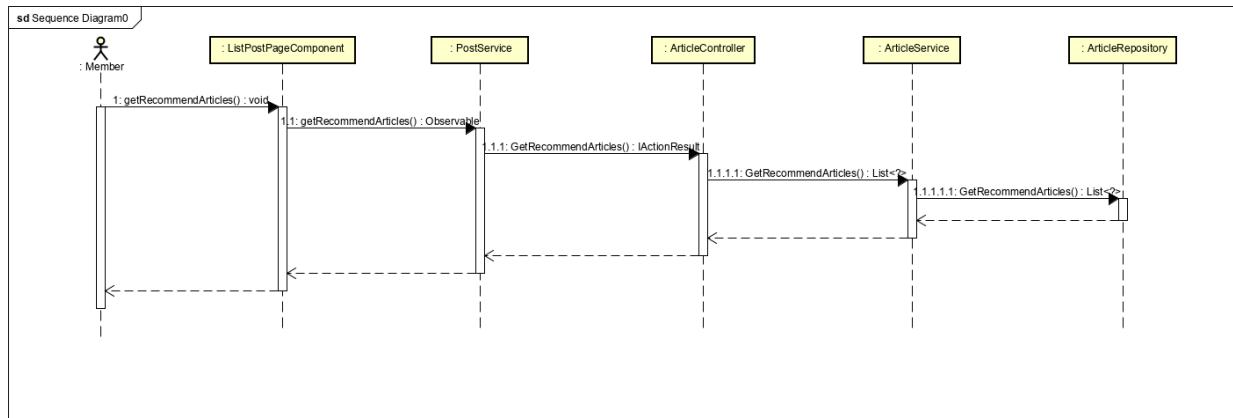


Figure 4-103: View recommended articles Sequence Diagram

4.3.4.32 View all users' account

Screen design

The screenshot shows the 'Người dùng' (Users) section of the TripSharing application. The left sidebar includes links for Dashboard, Tổng quan, Bài viết, Người dùng, Chủ đề, and Vi phạm. The main area displays a search bar with placeholder 'Tim kiếm' (Search) and a list of users:

- Hà Văn Thái (@havanthai) - Sông Lô, Vĩnh Phúc - 0 lượt theo dõi • 6 điểm tích lũy
- PhongNV (@phongnv) - Vĩnh Phúc - 0 lượt theo dõi • 36 điểm tích lũy
- Admin (@admin) - Sông Lô, Vĩnh Phúc - 0 lượt theo dõi • 0 điểm tích lũy
- PhongTV (@phongtv) - Vũ Bản Nam Định - 0 lượt theo dõi • 5 điểm tích lũy
- Minh Nguyệt (@Nguyetnga) - Thái Bình - 0 lượt theo dõi • 0 điểm tích lũy

Figure 4-104: View all users' account Screen Design

Class Diagram

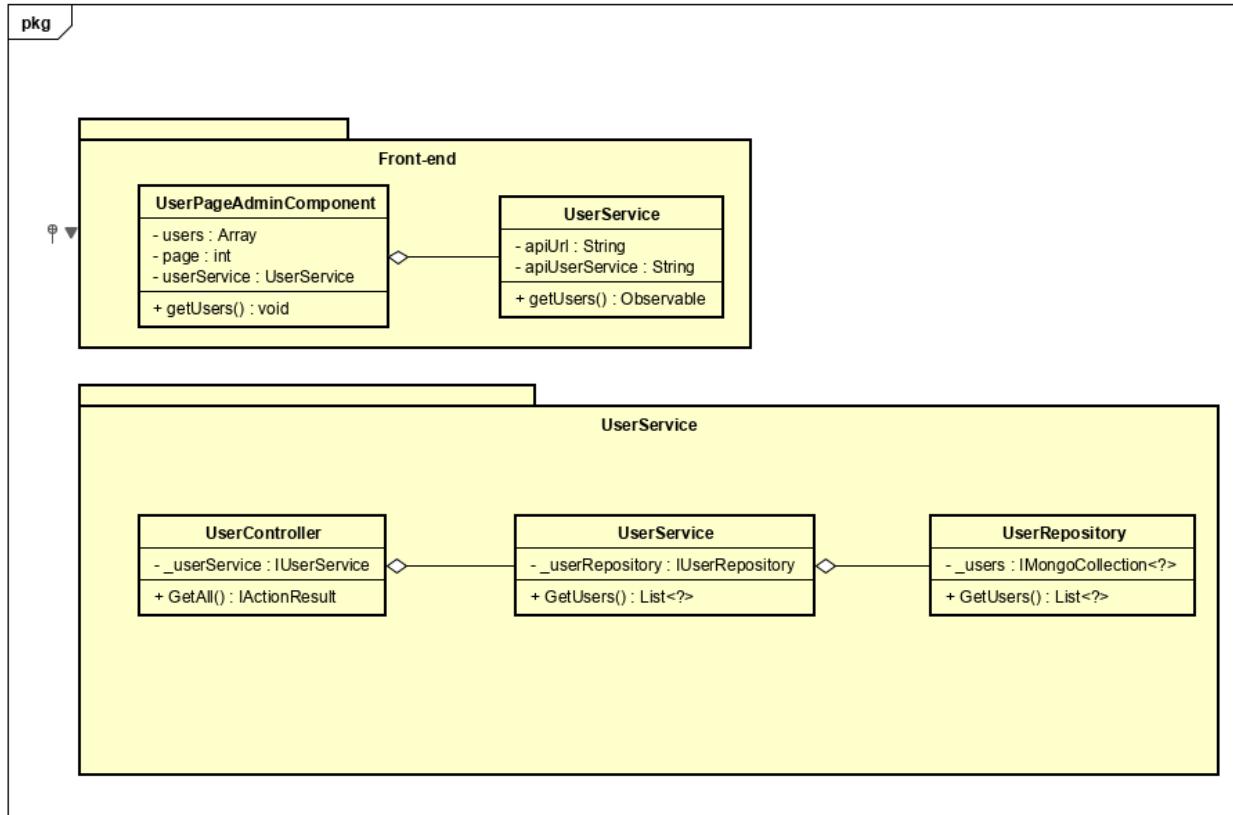


Figure 4-105: View all users' account Class Diagram

Class Specification

UserPageAdminComponent

UserPageAdminComponent			
Physical address	src\app\admin\pages\dashboard-page\components\user-page-admin\user-page-admin.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	users	Array	
2	userService	UserService	
3	page	Int	
Operations			
No	Name	Return Type	Description
1	getRecommendArticles	Void	

UserService

UserService			
Physical address			
src\app\core\services\user-service\user.service.ts			

Base Class	Class		
Attributes			
No	Name	Type	Description
1	apiUrl	String	
2	apiUserService	String	
Operations			
No	Name	Return Type	Description
1	getUsers	Observable	

UserController

UserController			
Physical address	src\Services\UserService\UserService\Controllers\UserController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_userService	IUserService	
Operations			
No	Name	Return Type	Description
1	GetAll	IActionResult	

UserService

UserService			
Physical address	src\Services\UserService\UserService\Services\UserService.cs		
Base Class	IUserService		
Attributes			
No	Name	Type	Description
1	_userRepository	IUserRepository	
Operations			
No	Name	Return Type	Description
1	GetUsers	List<?>	

UserRepository

UserRepository			
Physical address	src\Services\UserService\UserService\Repositories\UserRepository.cs		
Base Class	IUserRepository		
Attributes			
No	Name	Type	Description
1	_users	IMongoCollection<?>	
Operations			
No	Name	Return Type	Description

1	GetUsers	List<?>	
---	----------	---------	--

Sequence Diagram

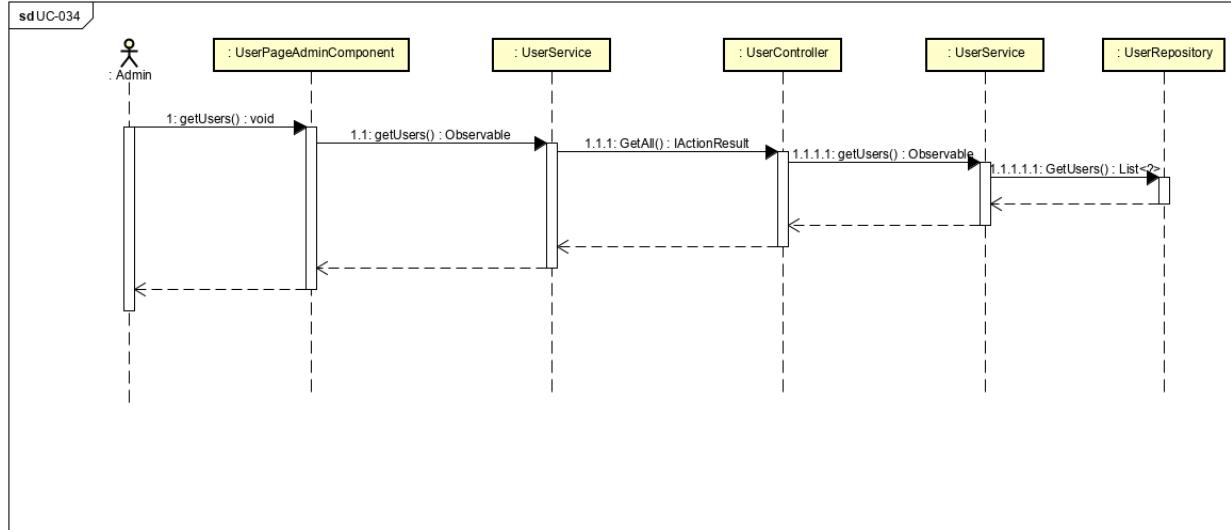


Figure 4-106: View all users' account Sequence Diagram

4.3.4.33 View reported accounts

Screen design

Người dùng vi phạm	PhongNV	Vĩnh Phúc	0 lượt theo dõi • 36 điểm tích lũy
	PhongNV @phongnv	Vĩnh Phúc	0 lượt theo dõi • 36 điểm tích lũy
	PhongNV @phongnv	Vĩnh Phúc	0 lượt theo dõi • 36 điểm tích lũy
	PhongNV @phongnv	Vĩnh Phúc	0 lượt theo dõi • 36 điểm tích lũy
	Hà Văn Thái @havanthai	Sông Lô, Vĩnh Phúc	0 lượt theo dõi • 6 điểm tích lũy

Figure 4-107: View reported accounts Screen Design

Class Diagram

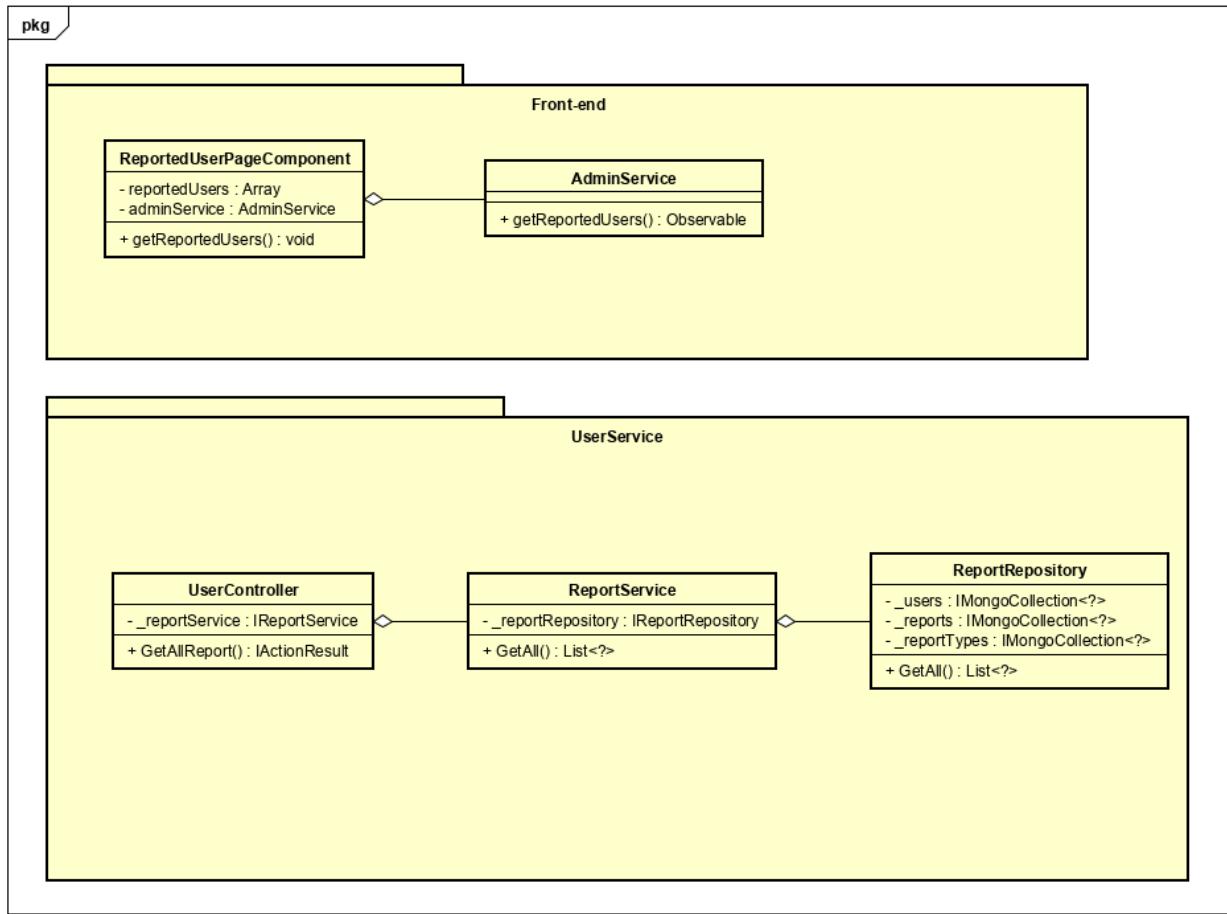


Figure 4-108: View reported accounts Class Diagram

Class Specification

ReportedUserPageComponent

UserPageAdminComponent			
Physical address	src\app\admin\pages\dashboard-page\components\reported-user-page\reported-user-page.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	reportedUsers	Array	
2	adminService	AdminService	
Operations			
No	Name	Return Type	Description
1	getReportedUsers	Void	

AdminService

AdminService

Physical address	src\app\admin\services\admin-service\admin.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
Operations			
No	Name	Return Type	Description
1	getReportedUsers	Observable	

UserController

UserController			
Physical address	src\Services\UserService\UserService\Controllers\UserController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_reportService	IReportService	
Operations			
No	Name	Return Type	Description
1	GetAllReport	IActionResult	

ReportService

ReportService			
Physical address	src\Services\UserService\UserService\Services\ReportService.cs		
Base Class	IReportService		
Attributes			
No	Name	Type	Description
1	_reportRepository	IReportRepository	
Operations			
No	Name	Return Type	Description
1	GetAll	List<?>	

ReportRepository

UserRepository			
Physical address	src\Services\UserService\UserService\Repositories\ReportRepository.cs		
Base Class	IReportRepository		
Attributes			
No	Name	Type	Description
1	_users	IMongoCollection<?>	
2	_reports	IMongoCollection<?>	
3	_reportTypes	IMongoCollection<?>	

Operations			
No	Name	Return Type	Description
1	GetAll	List<?>	

Sequence Diagram

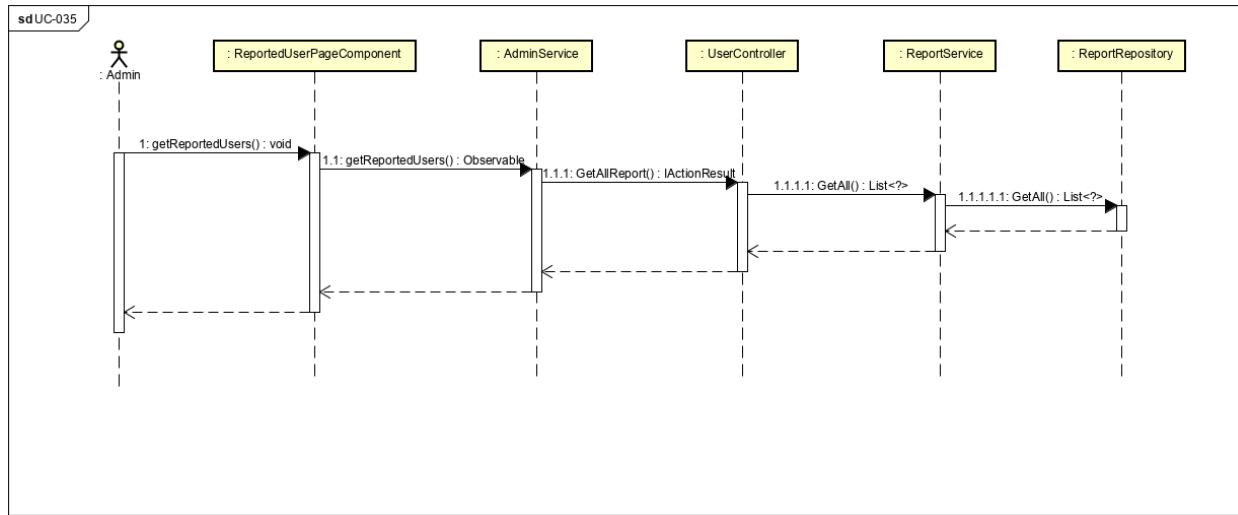


Figure 4-109: View reported accounts Sequence Diagram

4.3.4.34 Ban a user

Screen design

The screenshot shows the 'TripSharing' application interface. The top navigation bar includes 'TripSharing', 'Hi, ly phuc linh', and a profile icon. The left sidebar menu has sections like 'Dashboard', 'Tổng quan', 'Bài viết', 'Người dùng', 'Chủ đề', 'Vi phạm', 'Người dùng', 'Bài đăng', and 'Bình luận'. The main content area is titled 'Nguời dùng vi phạm' and lists users with their violations:

- PhongNV (@phongnv) - Vinh Phúc, 0 lượt theo dõi, 36 điểm tích lũy. Violation: Mạo danh người khác (1 ngày trước).
- PhongNV (@phongnv) - Vinh Phúc, 0 lượt theo dõi, 36 điểm tích lũy. Violation: Spam bài viết, bình luận (1 ngày trước).
- PhongNV (@phongnv) - Vinh Phúc, 0 lượt theo dõi, 36 điểm tích lũy. Violation: Mạo danh người khác (1 ngày trước).
- Hà Văn Thái (@havanthai) - Sông Lô, Vinh Phúc, 0 lượt theo dõi, 6 điểm tích lũy. Violation: Mạo danh người khác (6 ngày trước).

A sidebar on the right provides options: 'Trang cá nhân', 'Định chỉ', and 'Đã giải quyết'.

Figure 4-110: Ban a user Screen Design

Class Diagram

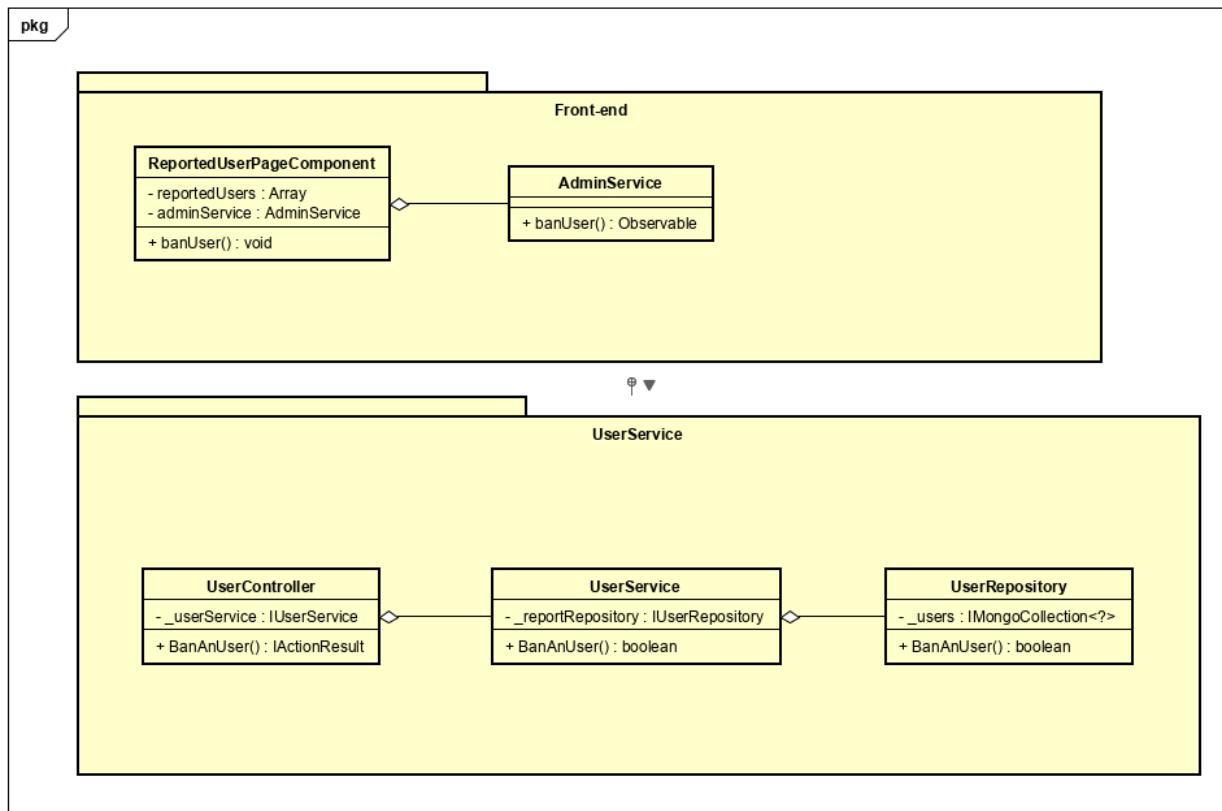


Figure 4-111: Ban a user Class Diagram

Class Specification

ReportedUserPageComponent

UserPageAdminComponent			
Physical address	src\app\admin\pages\dashboard-page\components\reported-user-page\reported-user-page.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	reportedUsers	Array	
2	adminService	AdminService	
Operations			
No	Name	Return Type	Description
1	banUser	Void	

AdminService

AdminService			
Physical address	src\app\admin\services\admin-service\admin.service.ts		
Base Class	Class		

Attributes			
No	Name	Type	Description
Operations			
No	Name	Return Type	Description
1	banUser	Observable	

UserController

UserController			
Physical address	src\Services\UserService\UserService\Controllers\UserController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_reportService	IReportService	
Operations			
No	Name	Return Type	Description
1	BanAnUser	IActionResult	

UserService

ReportService			
Physical address	src\Services\UserService\UserService\Services\UserService.cs		
Base Class	IUserService		
Attributes			
No	Name	Type	Description
1	_userRepository	IUserRepository	
Operations			
No	Name	Return Type	Description
1	BanAnUser	Bool	

UserRepository

UserRepository			
Physical address	src\Services\UserService\UserService\Repositories\UserRepository.cs		
Base Class	IUserRepository		
Attributes			
No	Name	Type	Description
1	_users	IMongoCollection<?>	
Operations			
No	Name	Return Type	Description
1	BanAnUser	Bool	

Sequence Diagram

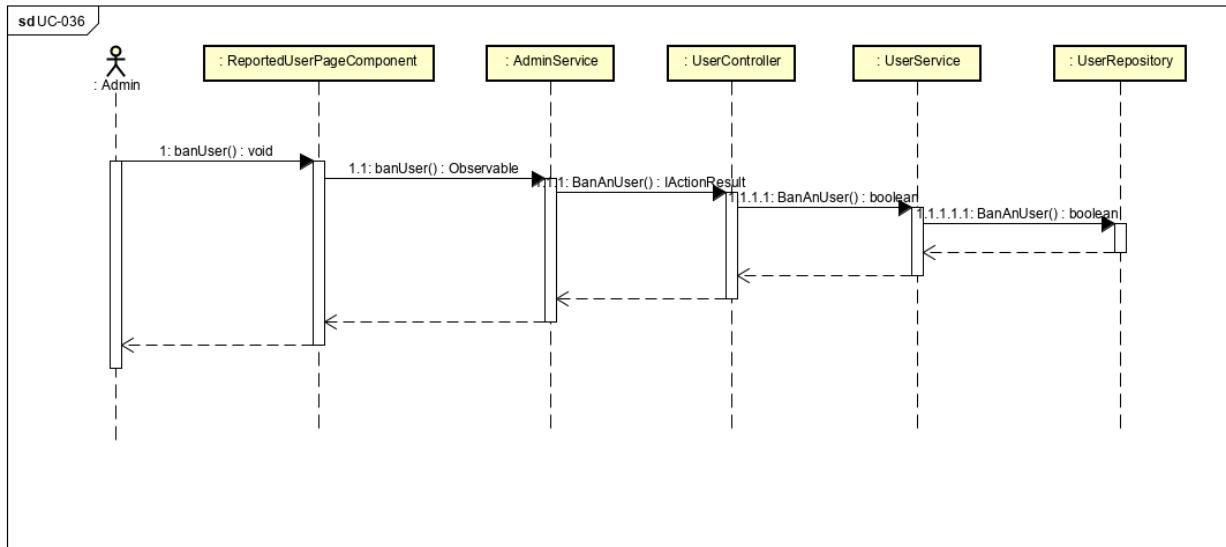


Figure 4-112: Ban a user Sequence Diagram

4.3.4.35 Unban a user

Screen design

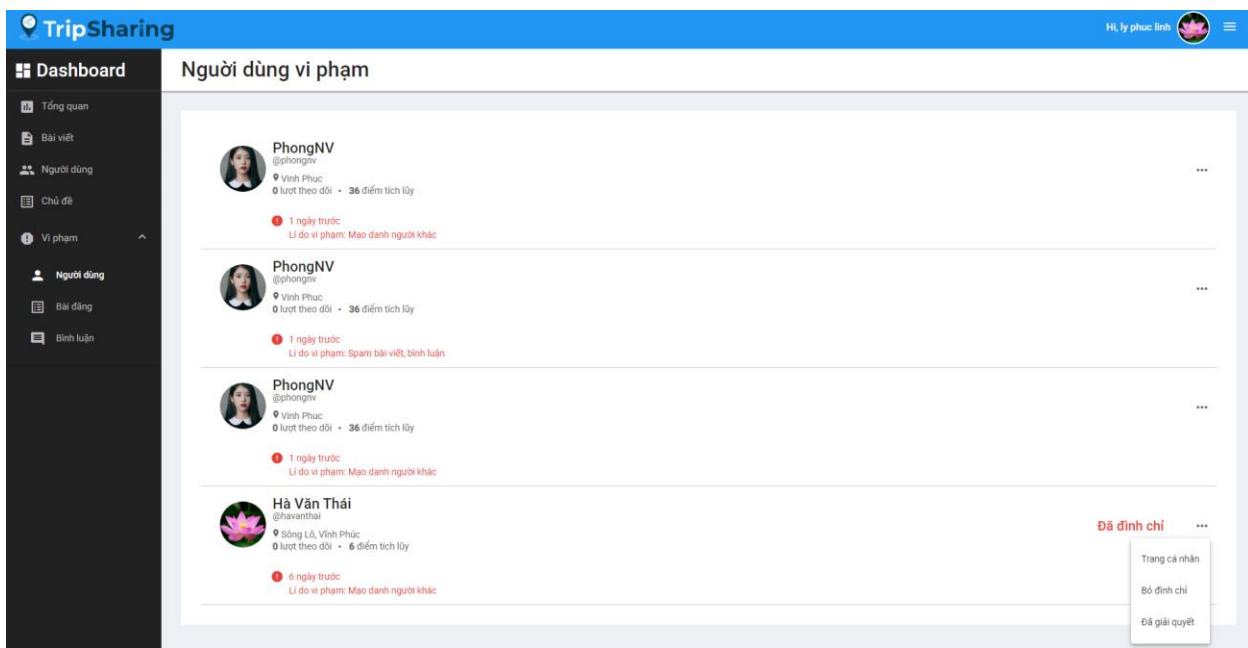


Figure 4-113: Unban a user Screen Design

Class Diagram

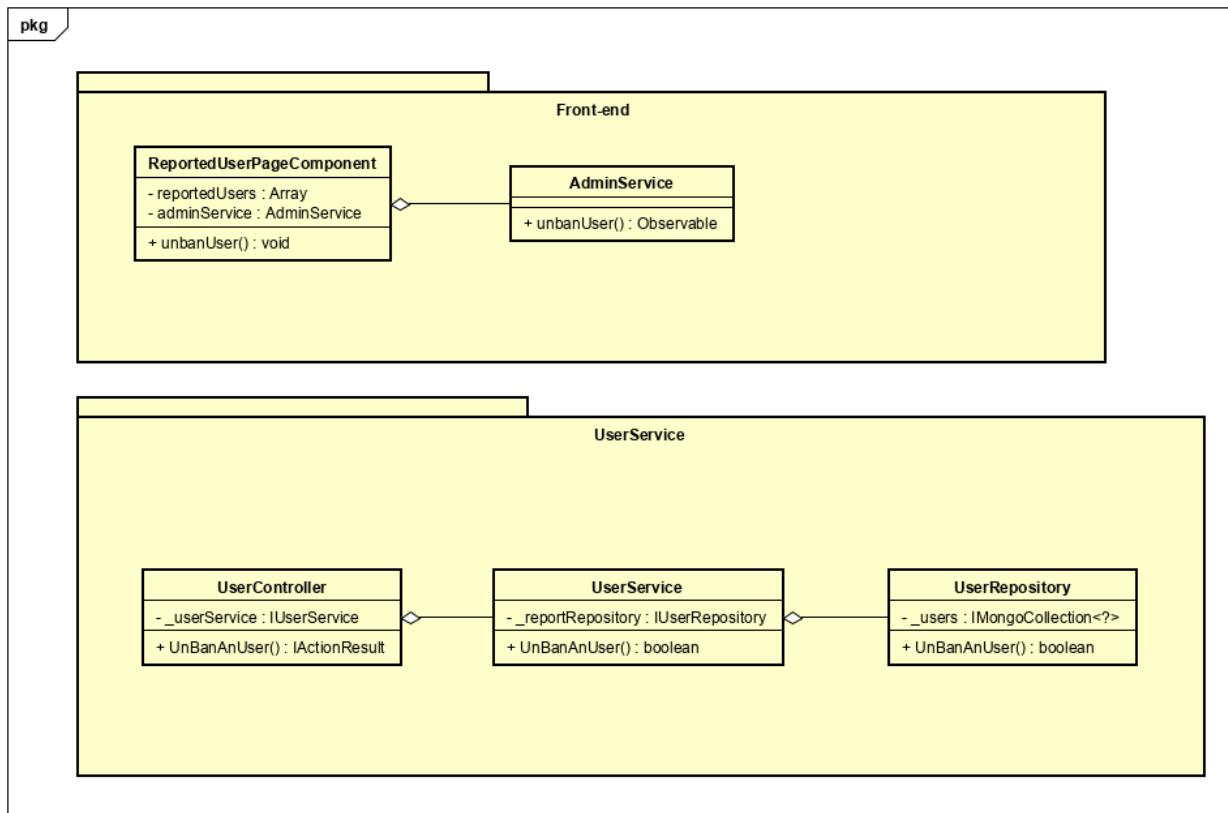


Figure 4-114: Unban a user Class Diagram

Class Specification

ReportedUserPageComponent

UserPageAdminComponent			
Physical address	src\app\admin\pages\dashboard-page\components\reported-user-page\reported-user-page.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	reportedUsers	Array	
2	adminService	AdminService	
Operations			
No	Name	Return Type	Description
1	unbanUser	Void	

AdminService

AdminService			
Physical address	src\app\admin\services\admin-service\admin.service.ts		
Base Class	Class		

Attributes			
No	Name	Type	Description
Operations			
No	Name	Return Type	Description
1	unbanUser	Observable	

UserController

UserController			
Physical address	src\Services\UserService\UserService\Controllers\UserController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_reportService	IReportService	
Operations			
No	Name	Return Type	Description
1	UnBanAnUser	IActionResult	

UserService

ReportService			
Physical address	src\Services\UserService\UserService\Services\UserService.cs		
Base Class	IUserService		
Attributes			
No	Name	Type	Description
1	_userRepository	IUserRepository	
Operations			
No	Name	Return Type	Description
1	UnBanAnUser	Bool	

UserRepository

UserRepository			
Physical address	src\Services\UserService\UserService\Repositories\UserRepository.cs		
Base Class	IUserRepository		
Attributes			
No	Name	Type	Description
1	_users	IMongoCollection<?>	
Operations			
No	Name	Return Type	Description
1	UnBanAnUser	Bool	

Sequence Diagram

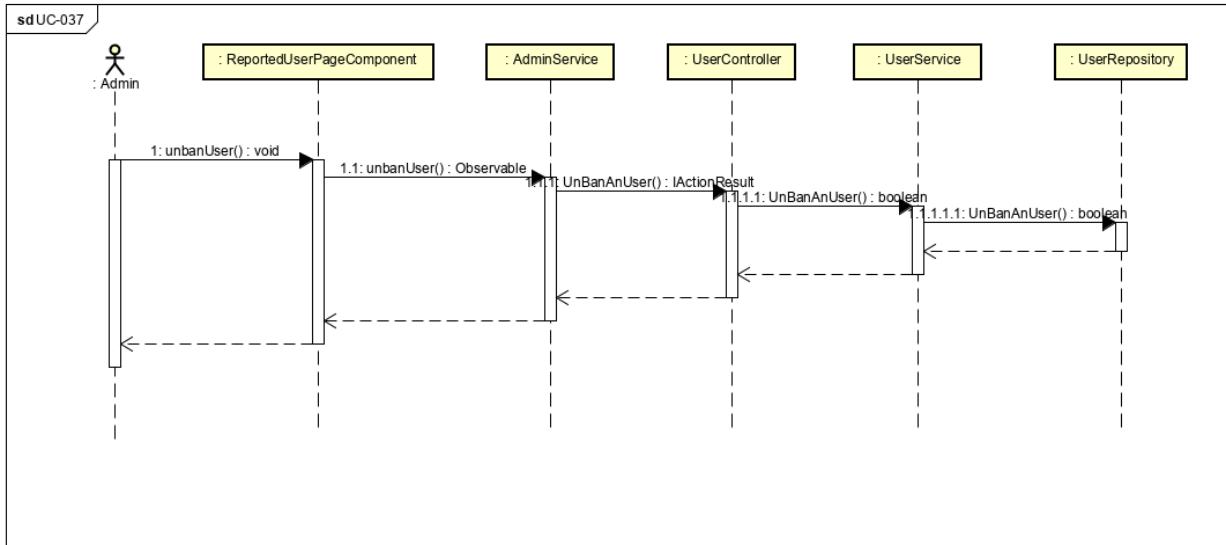


Figure 4-115: Unban a user Sequence Diagram

4.3.4.36 View reported posts

Screen design

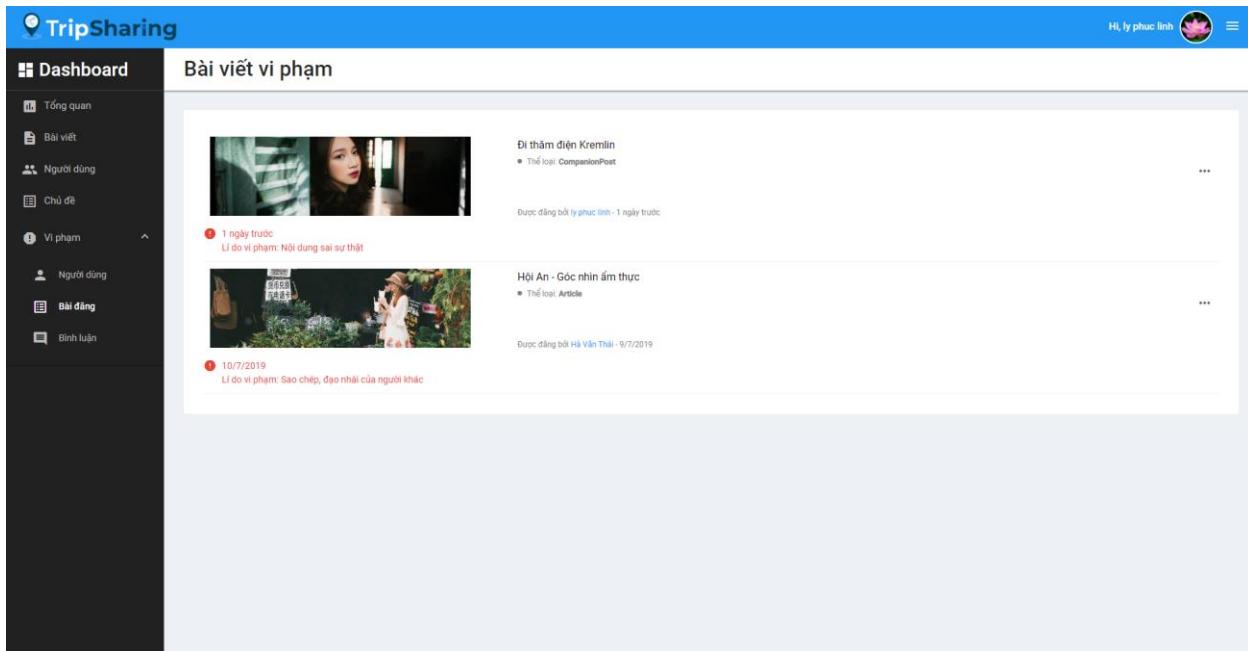


Figure 4-116: View reported posts Screen Design

Class Diagram

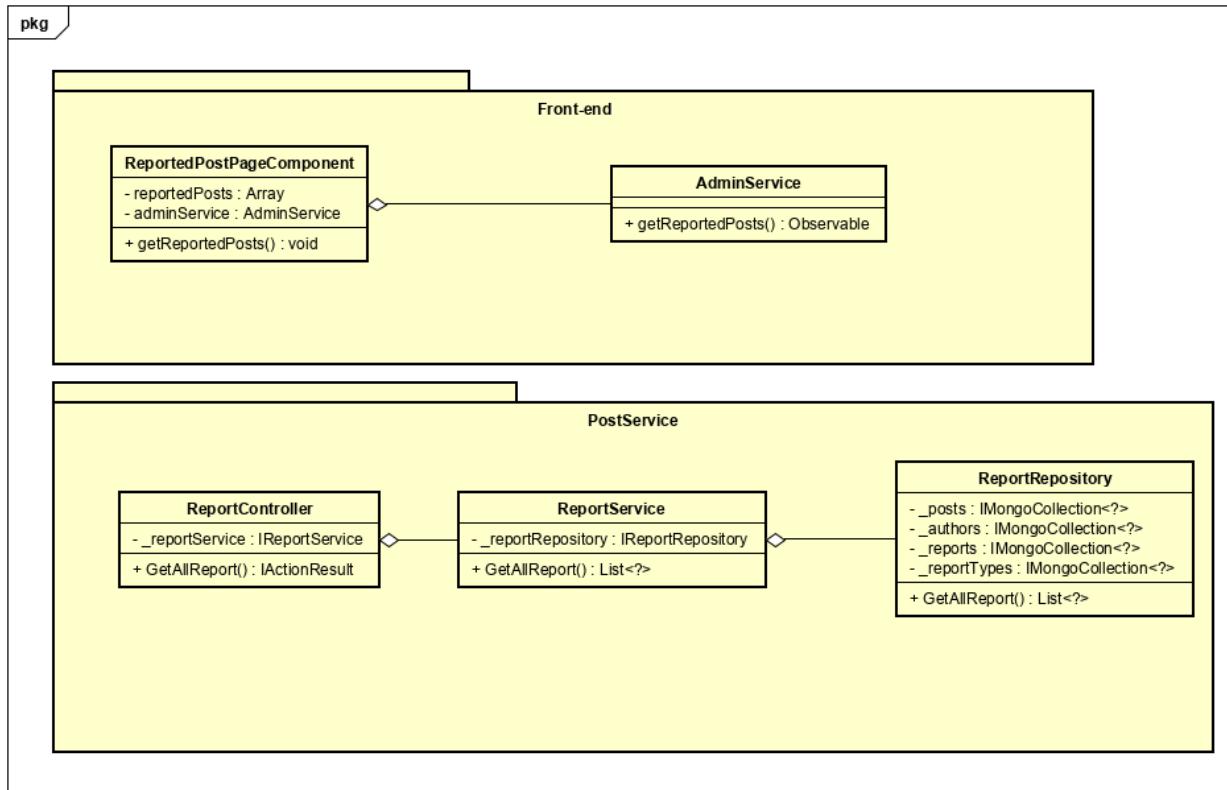


Figure 4-117: View reported posts Class Diagram

Class Specification

ReportedUserPageComponent

ReportedUserPageComponent			
Physical address	src\app\admin\pages\dashboard-page\components\reported-user-page\reported-user-page.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	reportedPosts	Array	
2	adminService	AdminService	
Operations			
No	Name	Return Type	Description
1	getReportedPosts	Void	

AdminService

AdminService			
Physical address	src\app\admin\services\admin-service\admin.service.ts		
Base Class	Class		
Attributes			

No	Name	Type	Description
Operations			
No	Name	Return Type	Description
1	getReportedPosts	Observable	

ReportController

ReportController			
Physical address	src\Services\PostService\PostService\Controllers\ReportController.cs		
Base Class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_reportService	IReportService	
Operations			
No	Name	Return Type	Description
1	GetAllReport	IActionResult	

ReportService

ReportService			
Physical address	src\Services\PostService\PostService\Services\ReportService.cs		
Base Class	IReportService		
Attributes			
No	Name	Type	Description
1	_reportRepository	IReportRepository	
Operations			
No	Name	Return Type	Description
1	GetAllReport	List<?>	

ReportRepository

ReportRepository			
Physical address	src\Services\PostService\PostService\Repositories\ReportRepository.cs		
Base Class	IUserRepository		
Attributes			
No	Name	Type	Description
1	_posts	IMongoCollection<?>	
2	_authors	IMongoCollection<?>	
3	_reports	IMongoCollection<?>	
4	_reportTypes	IMongoCollection<?>	
Operations			
No	Name	Return Type	Description

1	GetAllReport	List<?>	
---	--------------	---------	--

Sequence Diagram

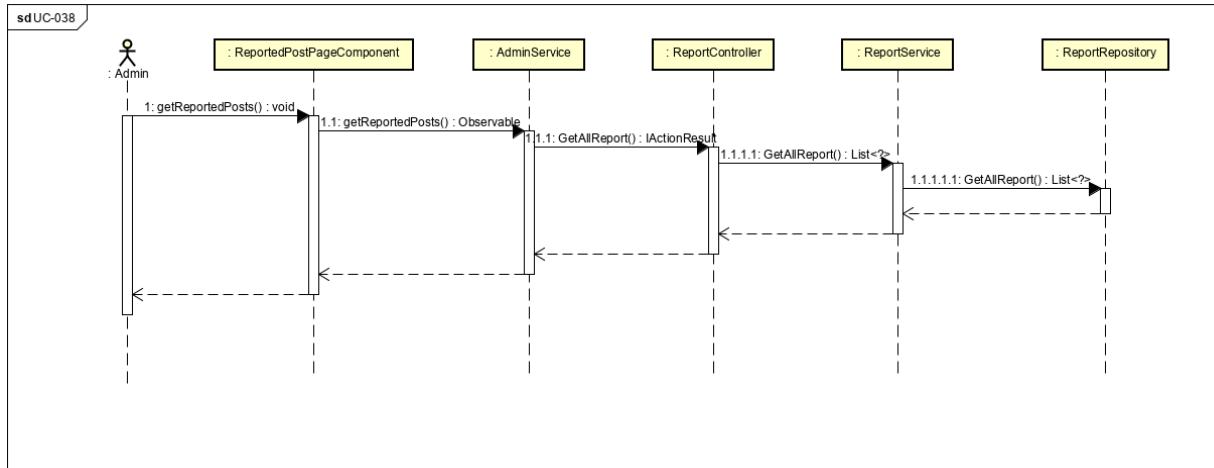


Figure 4-118: View reported posts Sequence Diagram

4.3.4.37 Remove a reported post

Screen Design

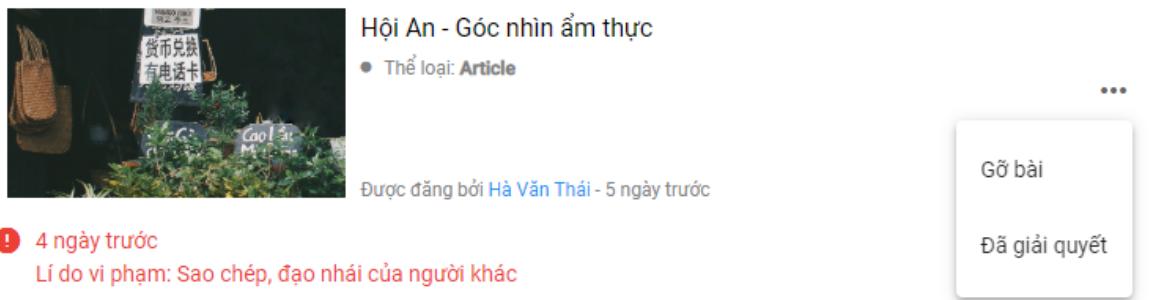


Figure 4-119: Remove a reported post screen design

Class Diagram

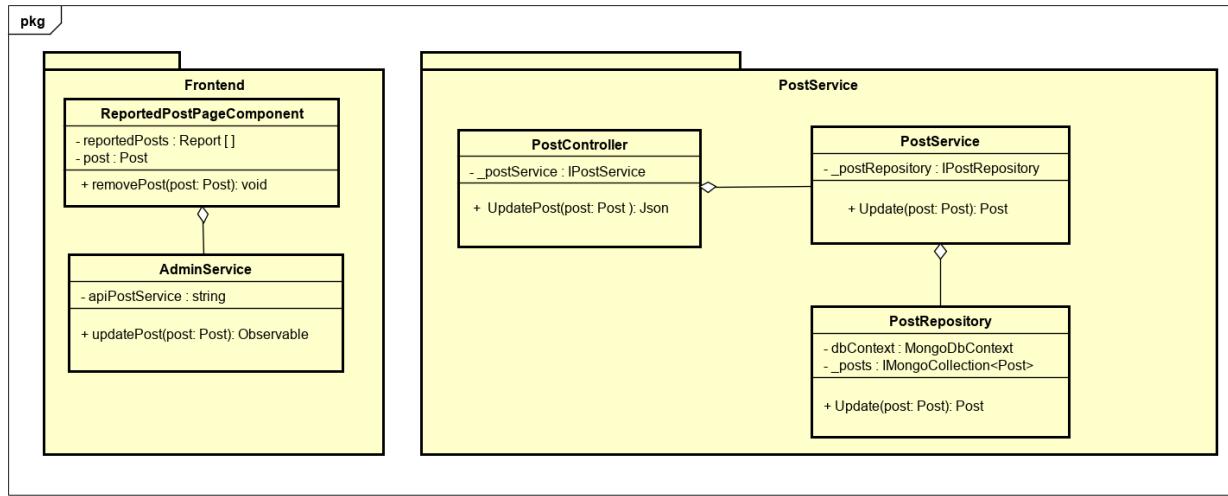


Figure 4-120: Administrator removes a reported post class diagram

Class Specification

ReportedPostPageComponent

ReportedPostPageComponent			
Physical address	src/app/admin/pages/dashboard-page/components/reported-post-page/reported-post-page.component.ts		
Base class	Class		
Attributes			
No	Name	Type	Description
1	reportedPosts	Report []	
2	post	Post	
Operation			
No	Name	Return Type	Description
1	removePost	void	

AdminService

AdminService			
Physical address	src/app/admin/services/admin-service/admin.service		
Base class	Class		
Attributes			
No	Name	Type	Description
1	apiPostService	string	
Operation			
No	Name	Retun Type	Description
1	updatePost	Observable	

PostController

PostController			
Physical address	src/Services/PostService/PostService/Controllers/PostController.cs		
Base class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_postService	IPostService	
Operation			
No	Name	Retun Type	Description
1	UpdatePost	Json	

PostService

PostService			
Physical address	src/Services/PostService/PostService/Service/PostService.cs		
Base class	IPostService		
Attributes			
No	Name	Type	Description
1	_postRepository	IPostRepository	
Operation			
No	Name	Retun Type	Description
1	Update	Post	

PostReposiory

PostRepository			
Physical address	src/Services/PostService/PostService/Repository/PostRepository.cs		
Base class	IPostRepository		
Attributes			
No	Name	Type	Description
1	dbContext	MongoDbContext	
2	_posts	ImongoCollection<Post>	
Operation			
No	Name	Retun Type	Description
1	Update	Post	

Sequence Diagram

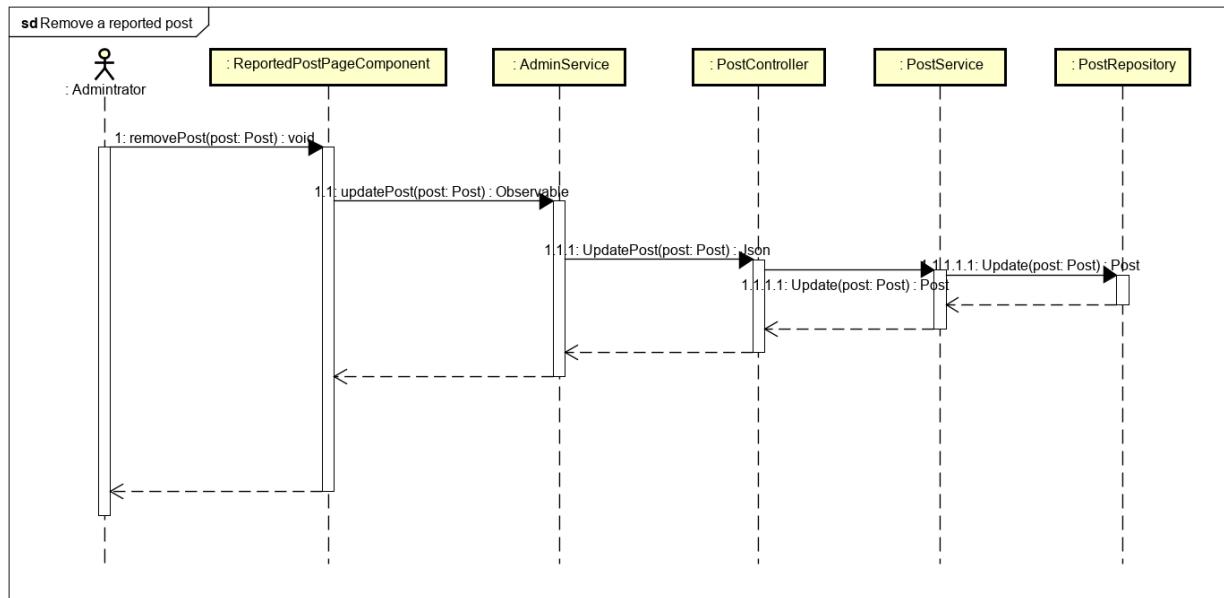


Figure 4-121: Administrator remove a reported post sequence diagram

4.3.4.38 Restore a removed post

Screen Design

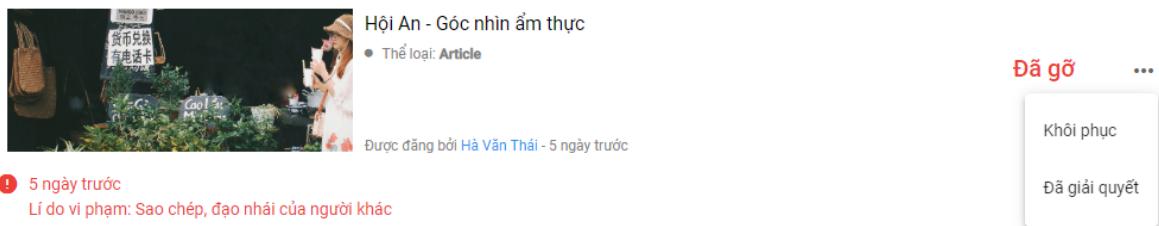


Figure 2-122: Restore a removed post screen design

Class Diagram

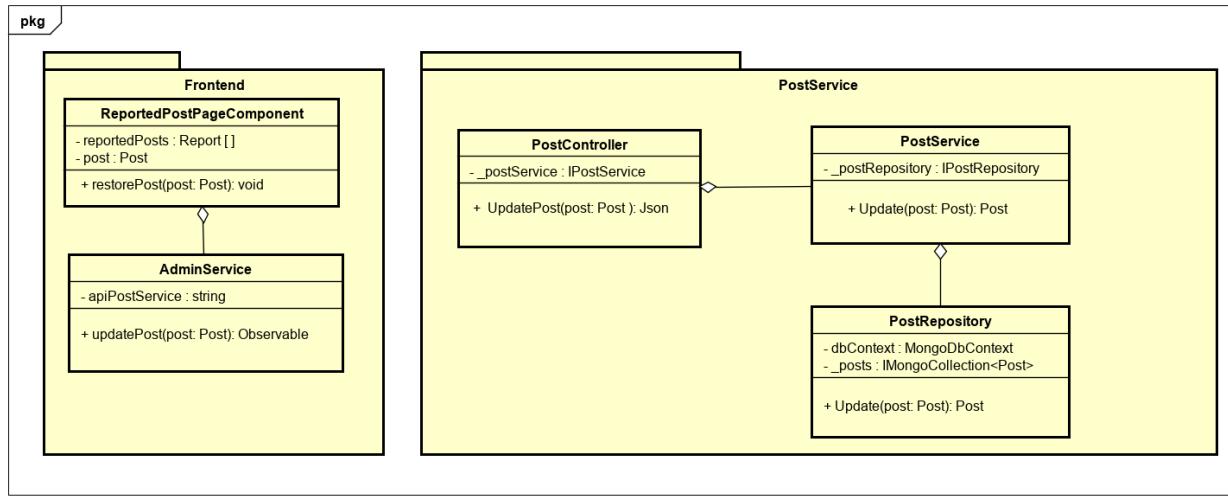


Figure 4-123: Administrator restore a removed post class diagram

Class Specification

ReportedPostPageComponent

ReportedPostPageComponent			
Physical address	src/app/admin/pages/dashboard-page/components/reported-post-page/reported-post-page.component.ts		
Base class	Class		
Attributes			
No	Name	Type	Description
1	reportedPosts	Report []	
2	post	Post	
Operation			
No	Name	Return Type	Description
1	restorePost	void	

AdminService

AdminService			
Physical address	src/app/admin/services/admin-service/admin.service		
Base class	Class		
Attributes			
No	Name	Type	Description
1	apiPostService	string	
Operation			
No	Name	Retun Type	Description
1	updatePost	Observable	

PostController

PostController			
Physical address	src/Services/PostService/PostService/Controllers/PostController.cs		
Base class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_postService	IPostService	
Operation			
No	Name	Retun Type	Description
1	UpdatePost	Json	

PostService

PostService			
Physical address	src/Services/PostService/PostService/Service/PostService.cs		
Base class	IPostService		
Attributes			
No	Name	Type	Description
1	_postRepository	IPostRepository	
Operation			
No	Name	Retun Type	Description
1	Update	Post	

PostReposiory

PostRepository			
Physical address	src/Services/PostService/PostService/Repository/PostRepository.cs		
Base class	IPostRepository		
Attributes			
No	Name	Type	Description
1	dbContext	MongoDbContext	
2	_posts	ImongoCollection<Post>	
Operation			
No	Name	Retun Type	Description
1	Update	Post	

Sequence Diagram

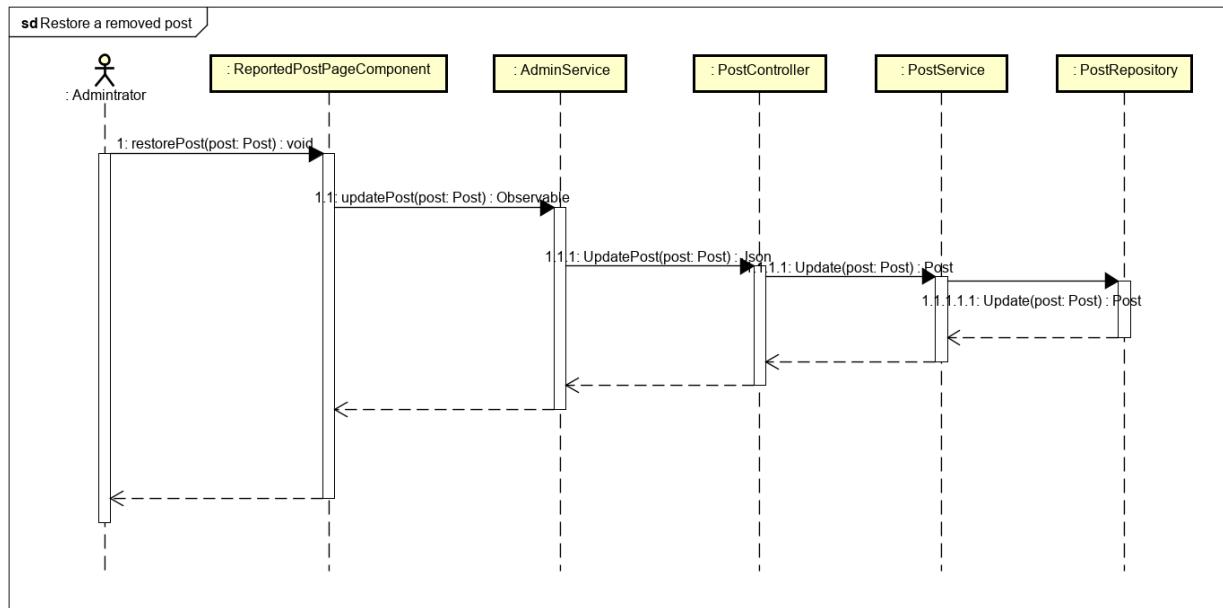


Figure 4-124: Administrator restore a removed post sequence diagram

4.3.4.39 View reported comments

Screen Design

Du lịch 0đ chỉ cần đặt cọc trước 500.000đ sẽ được hoàn trả sau chuyến đi. Ai có nhu cầu liên hệ 0987489397
 tại bài viết [5d4e880ce1f3720001ec1c19](#)
 Đăng bởi: [Hà Văn Thái](#) - 3 phút trước

- ! 1 phút trước
Lí do vi phạm: Lừa đảo, đăng thông tin sai sự thật

Figure 4-125: View reported comments screen design

Class Diagram

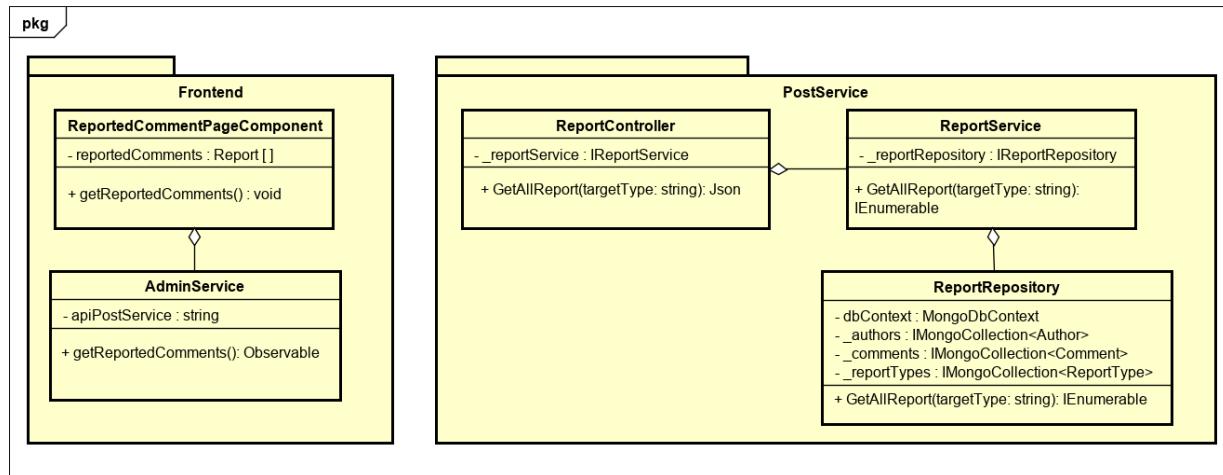


Figure 4-126: Administrator view reported comment class diagram

Class Specification

ReportedCommentPageComponent

ReportedCommentPageComponent			
Physical address	src/app/admin/pages/dashboard-page/components/reported-comment-page/reported-comment-page.component.ts		
Base class	Class		
Attributes			
No	Name	Type	Description
1	reportedComments	Report []	
Operation			
No	Name	Return Type	Description
1	getReportedComments	void	

AdminService

AdminService			
Physical address	src/app/admin/services/admin-service/admin.service		
Base class	Class		
Attributes			
No	Name	Type	Description
1	apiPostService	String	
Operation			
No	Name	Type	Description
1	getReportedComments	Observable	

ReportController

ReportController			
Physical address	src/Services/PostService/PostService/Controllers/ReportController.cs		
Base class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_reportService	IReportService	
Operation			
No	Name	Return Type	Description
1	GetAllReport	Json	

ReportService

ReportService			
Physical address	src/Services/PostService/PostService/Service/ReportService.cs		
Base class	IReportService		
Attributes			

No	Name	Type	Description
1			
Operation			
No	Name	Return Type	Description
1	GetAllReport	IEnumerable	

ReportRepository

ReportRepository			
Physical address	src/Services/PostService/PostService/Repository/ReportRepository.cs		
Base class	IReportRepository		
Attributes			
No	Name	Type	Description
1	dbContext	MongoDbContext	
2	_authors	ImongoCollection<Author>	
3	_comments	ImongoCollection<Comment>	
4	_reportTypes	ImongoCollection<ReportType>	
Operation			
No	Name	Return Type	Description
1	GetAllReport	IEnumerable	

Sequence Diagram

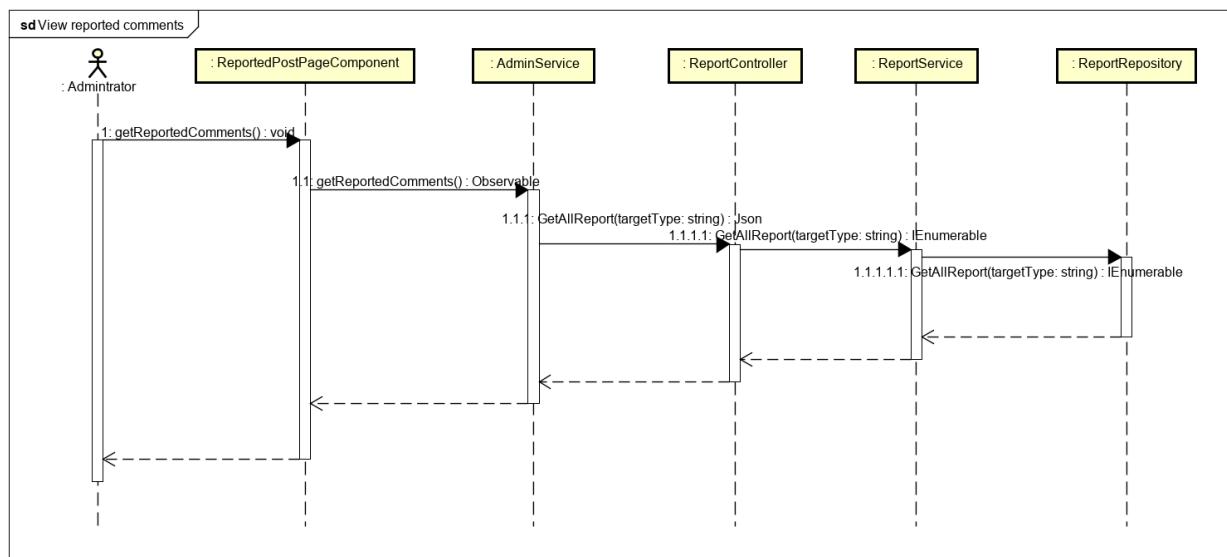


Figure 4-127: Administrator view reported comment sequence diagram

4.3.4.40 Remove a reported comment

Screen Design



Figure 4-128: Remove a reported comment screen design

Class Diagram

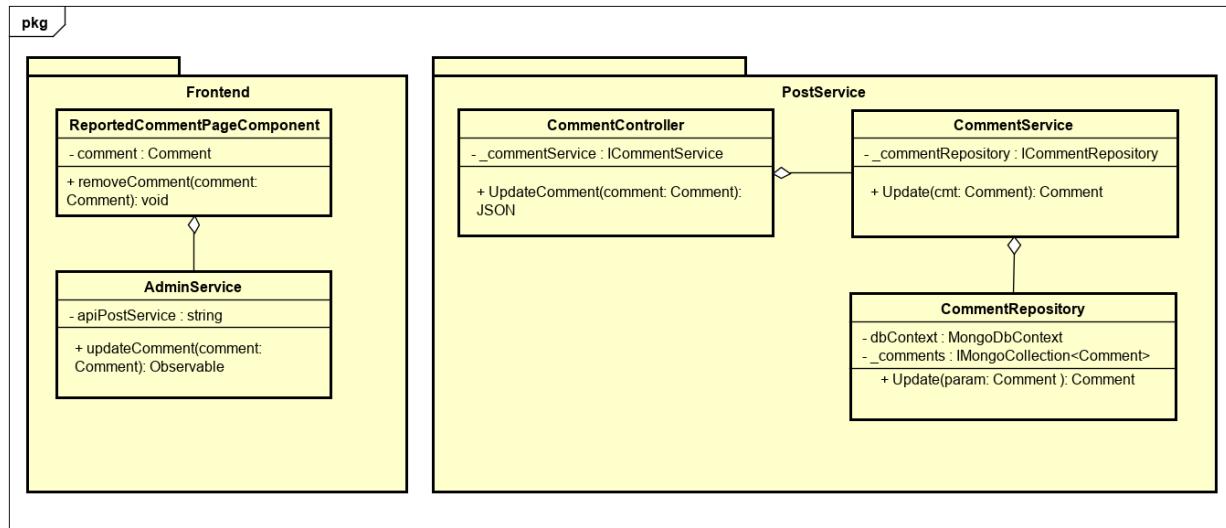


Figure 4-129: Administrator remove a reported comment class diagram

Class Specification

ReportedCommentPageComponent

ReportedCommentPageComponent			
Physical address	src/app/admin/pages/dashboard-page/components/reported-comment-page/reported-comment-page.component.ts		
Base class	Class		
Attributes			
No	Name	Type	Description

1	Comment	Comment	
Operation			
No	Name	Type	Description
1	removeComment	void	

AdminService

AdminService			
Physical address	./ src/app/admin/services/admin-service/admin.service		
Base class	Class		
Attributes			
No	Name	Type	Description
1	apiPostService	String	
Operation			
No	Name	Return Type	Description
1	updateComment	Observable	

CommentController

CommentController			
Physical address	src/Services/PostService/PostService/Controllers/CommentController.cs		
Base class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_commentService	ICommentService	
Operation			
No	Name	Type	Description
1	UpdateComment	Json	

CommentService

CommentService			
Physical address	src/Services/PostService/PostService/Service/CommentService.cs		
Base class	ICommentService		
Attributes			
No	Name	Type	Description
1	_commentRepository	ICommentRepository	
Operation			
No	Name	Type	Description
1	Update	Comment	

CommentRepository

CommentRepository

Physical address	src/Services/PostService/PostService/Repository/CommentRepository.cs		
Base class	ICommentRepository		
Attributes			
No	Name	Type	Description
1	dbContext	MongoDbContext	
2	_comments	ImongoCollection<Comment>	
Operation			
No	Name	Type	Description
1	Update	Comment	

Sequence Diagram

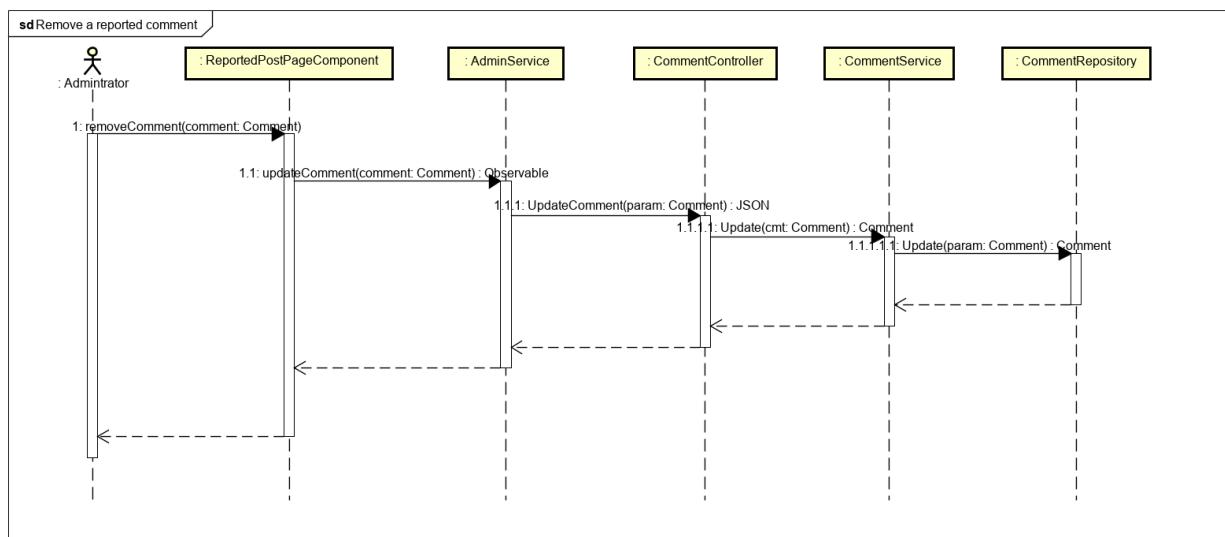


Figure 4-130: Administrator remove a reported comment sequence diagram

4.3.4.41 Restore a removed comment

Screen Design

Du lịch 0đ chỉ cần đặt cọc trước 500.000đ sẽ được hoàn trả sau chuyến đi. Ai có nhu cầu liên hệ 0987489397
tại bài viết [5d4e880ce1f3720001ec1c19](#)
Đăng bởi: [Hà Văn Thái](#) - 3 phút trước

Đã gỡ ...

! 1 phút trước
Lý do vi phạm: Lừa đảo, đăng thông tin sai sự thật

Ikajsd
tại bài viết [5d4e880ce1f3720001ec1c19](#)
Đăng bởi: [ly phuc linh](#) - 14/7/2019

! 15/7/2019
Lý do vi phạm: Nội dung sai sự thật

Khôi phục
Đã giải quyết
...

Figure 4-131: Restore a removed comment screen design

Class Diagram

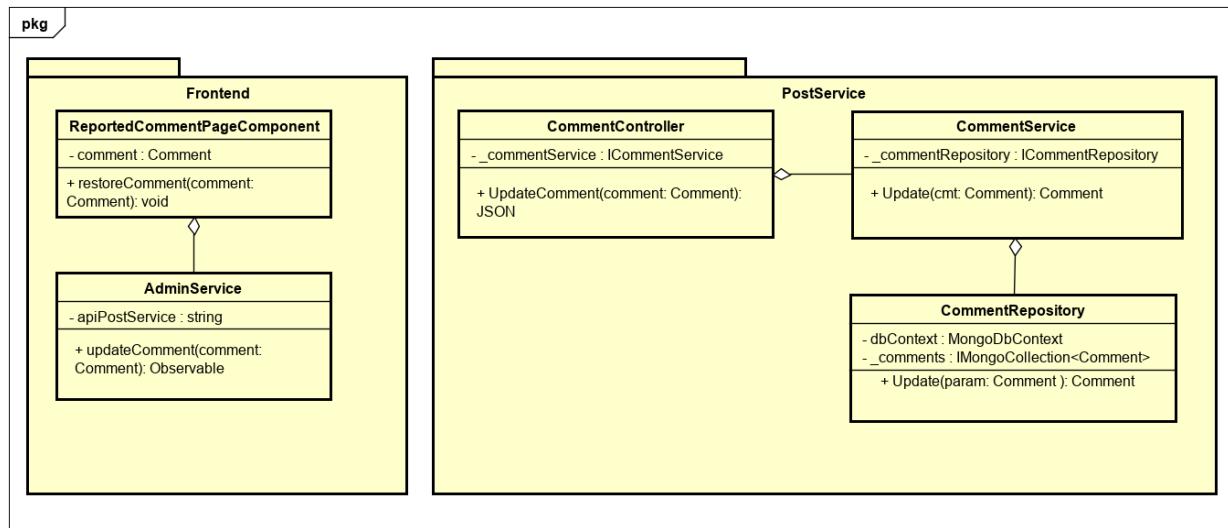


Figure 4-132: Administrator restore a removed comment class diagram

Class Specification

ReportedCommentPageComponent

ReportedCommentPageComponent			
Physical address	src/app/admin/pages/dashboard-page/components/reported-comment-page/reported-comment-page.component.ts		
Base class	Class		
Attributes			
No	Name	Type	Description
1	comment	Comment	
Operation			
No	Name	Type	Description
1	restoreComment	void	

AdminService

AdminService			
Physical address	src/app/admin/services/admin-service/admin.service		
Base class	Class		
Attributes			
No	Name	Type	Description
1	apiPostService	String	
Operation			
No	Name	Return Type	Description
1	updateComment	Observable	

CommentController

CommentController			
Physical address	src/Services/PostService/PostService/Controllers/CommentController.cs		
Base class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_commentService	ICommentService	
Operation			
No	Name	Type	Description
1	UpdateComment	Json	

CommentService

CommentService			
Physical address	src/Services/PostService/PostService/Service/CommentService.cs		
Base class	ICommentService		
Attributes			
No	Name	Type	Description
1	_commentRepository	ICommentRepository	
Operation			
No	Name	Type	Description
1	Update	Comment	

CommentRepository

CommentRepository			
Physical address	src/Services/PostService/PostService/Repository/CommentRepository.cs		
Base class	ICommentRepository		
Attributes			
No	Name	Type	Description
1	dbContext	MongoDbContext	
2	_comments	ImongoCollection<Comment>	
Operation			
No	Name	Type	Description
1	Update	Comment	

Sequence Diagram

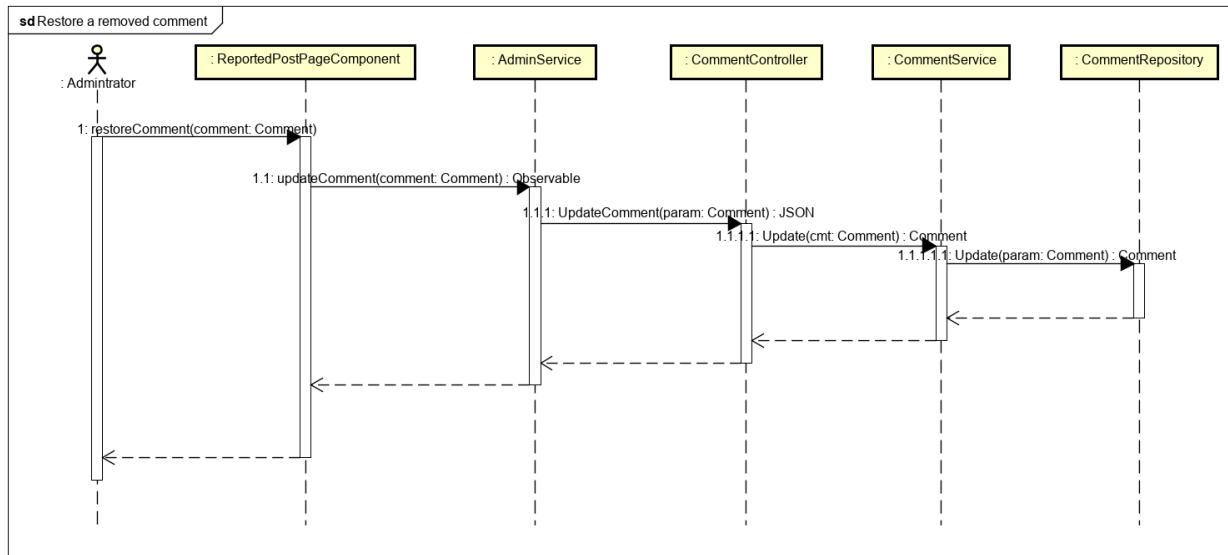


Figure 4-133: Administrator restore a removed comment sequence diagram

4.3.4.42 Add a travel topic

Screen Design

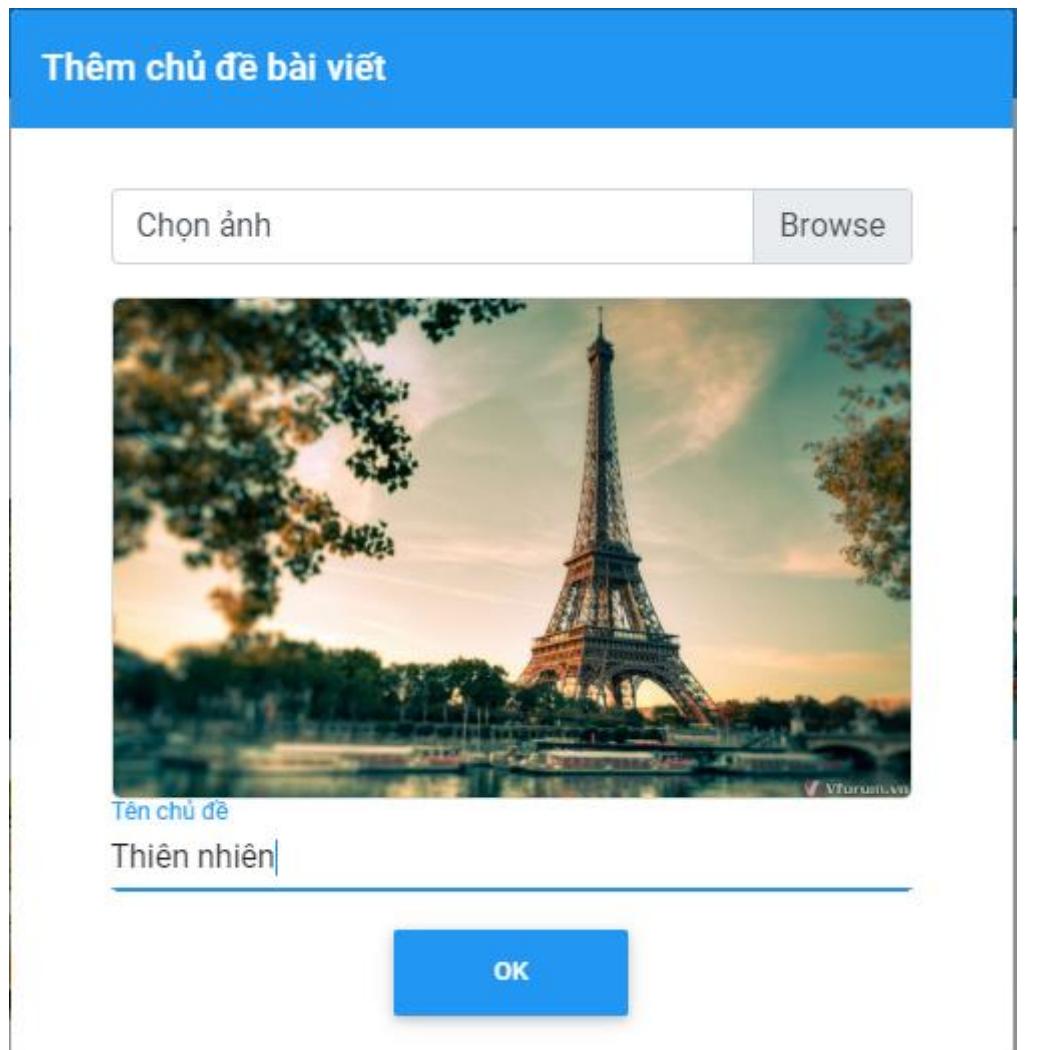


Figure 4-134: Add a travel topic screen design

Class Diagram

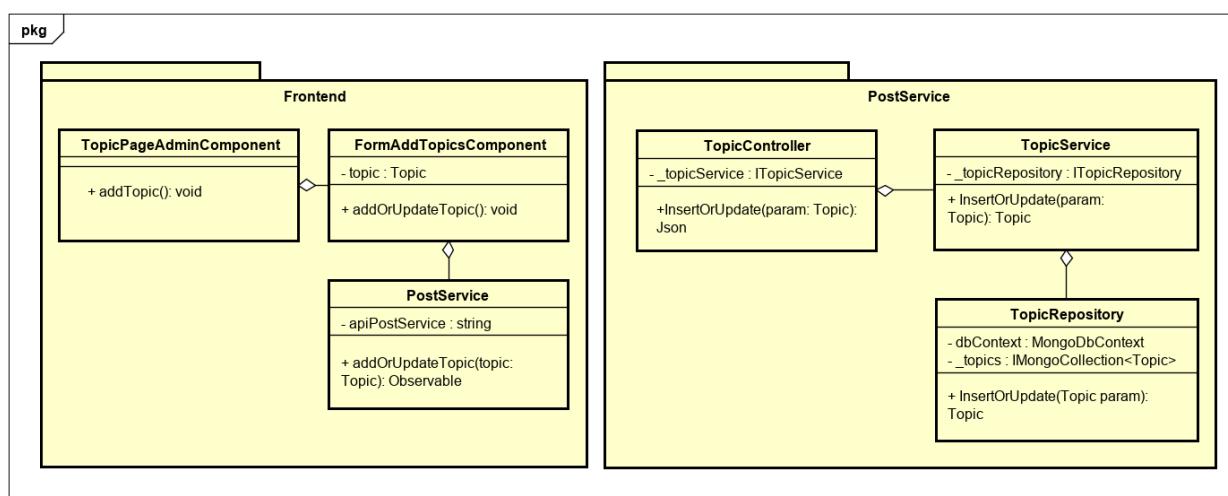


Figure 4-135: Administrator add a travel topic class diagram

Class Specification

TopicPageAdminComponent

TopicPageAdminComponent			
Physical address	src/app/admin/pages/dashboard-page/components/topic-page-admin/topic-page-admin.component.ts		
Base class	Class		
Attributes			
No	Name	Type	Description
Operation			
No	Name	Type	Description
1	addTopic	void	

FormAddTopicsComponent

FormAddTopicsComponent			
Physical address	src/app/admin/pages/dashboard-page/components/topic-page-admin/form-add-topics/form-add-topics.component.ts		
Base class	Class		
Attributes			
No	Name	Type	Description
1	topic	Topic	
Operation			
No	Name	Type	Description
1	addOrUpdateTopic	void	

PostService

PostService			
Physical address	src/app/core/services/post-service/post.service.ts		
Base class	Class		
Attributes			
No	Name	Type	Description
1	apiPostService	String	
Operation			
No	Name	Return Type	Description
1	addOrUpdateTopic	Observable	

TopicController

TopicController			
-----------------	--	--	--

Physical address	src/Services/PostService/PostService/Controllers/TopicController.cs		
Base class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_topicService	ITopicService	
Operation			
No	Name	Type	Description
1	InsertOrUpdate	Json	

TopicService

TopicService			
Physical address	src/Services/PostService/PostService/Service/TopicService.cs		
Base class	ITopicService		
Attributes			
No	Name	Type	Description
1	_topicRepository	ITopicRepository	
Operation			
No	Name	Type	Description
1	InsertOrUpdate	Topic	

TopicRepository

TopicRepository			
Physical address	src/Services/PostService/PostService/Repository/TopicRepository.cs		
Base class	ITopicRepository		
Attributes			
No	Name	Type	Description
1	dbContext	MongoDbContext	
2	_topics	ImongoCollection <Topic>	
Operation			
No	Name	Type	Description
1	InsertOrUpdate	Topic	

Sequence Diagram

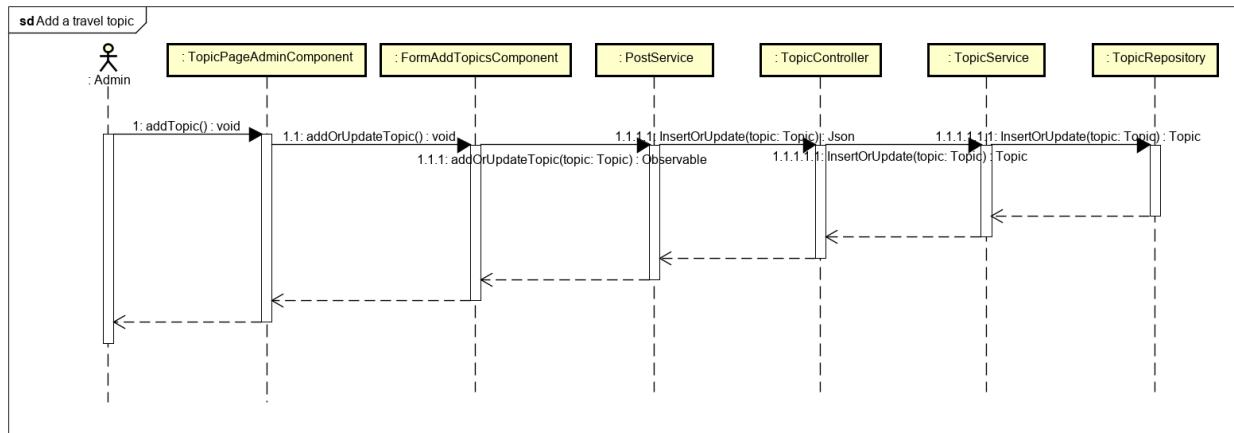


Figure 4-136: Administrator add a travel topic sequence diagram

4.3.4.43 Edit a travel topic

Screen Design

Chỉnh sửa chủ đề bài viết

Chọn ảnh

Browse



Tên chủ đề

Khám phá mạo hiểm

OK

Figure 4-137: Edit a travel topic screen design

Class Diagram

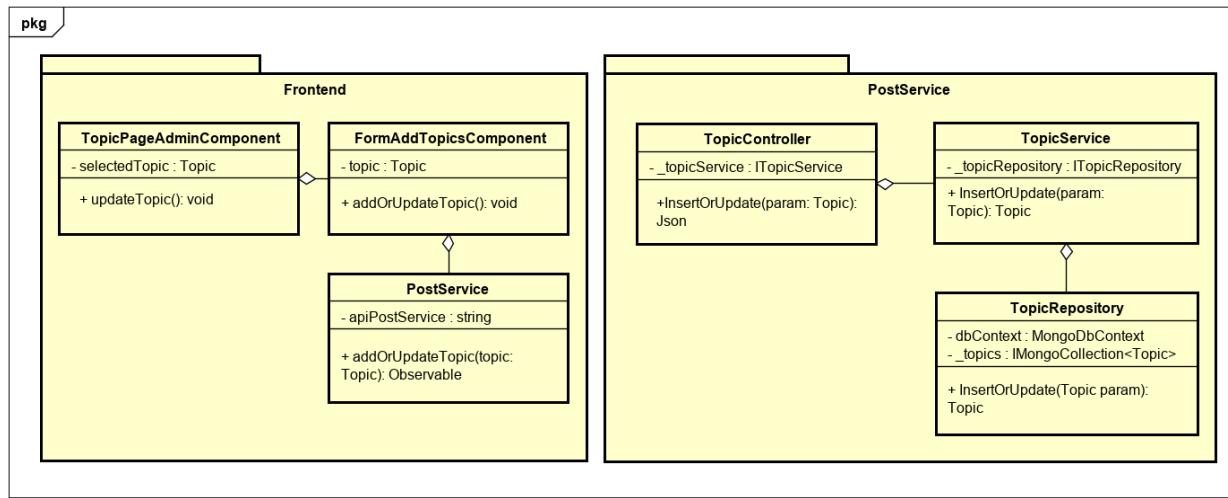


Figure 4-138: Administrator edit a travel topic class diagram

Class Specification

TopicPageAdminComponent

TopicPageAdminComponent			
Physical address	src/app/admin/pages/dashboard-page/components/topic-page-admin/topic-page-admin.component.ts		
Base class	Class		
Attributes			
No	Name	Type	Description
1	selectedTopic	Topic	
Operation			
No	Name	Type	Description
1	updateTopic	void	

FormAddTopicsComponent

FormAddTopicsComponent			
Physical address	src/app/admin/pages/dashboard-page/components/topic-page-admin/form-add-topics/form-add-topics.component.ts		
Base class	Class		
Attributes			
No	Name	Type	Description
1	topic	Topic	
Operation			
No	Name	Type	Description
1	addOrUpdateTopic	void	

PostService

PostService			
Physical address	src/app/core/services/post-service/post.service.ts		
Base class	Class		
Attributes			
No	Name	Type	Description
1	apiPostService	String	
Operation			
No	Name	Return Type	Description
1	addOrUpdateTopic	Observable	

TopicController

TopicController			
Physical address	src/Services/PostService/PostService/Controllers/TopicController.cs		
Base class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_topicService	ITopicService	
Operation			
No	Name	Type	Description
1	InsertOrUpdate	Json	

TopicService

TopicService			
Physical address	src/Services/PostService/PostService/Service/TopicService.cs		
Base class	ITopicService		
Attributes			
No	Name	Type	Description
1	_topicRepository	ITopicRepository	
Operation			
No	Name	Type	Description
1	InsertOrUpdate	Topic	

TopicRepository

TopicRepository			
Physical address	src/Services/PostService/PostService/Repository/TopicRepository.cs		
Base class	ITopicRepository		
Attributes			
No	Name	Type	Description
1	dbContext	MongoDbContext	
2	_topics	ImongoCollection<Topic>	

Operation			
No	Name	Type	Description
1	InsertOrUpdate	Topic	

Sequence Diagram

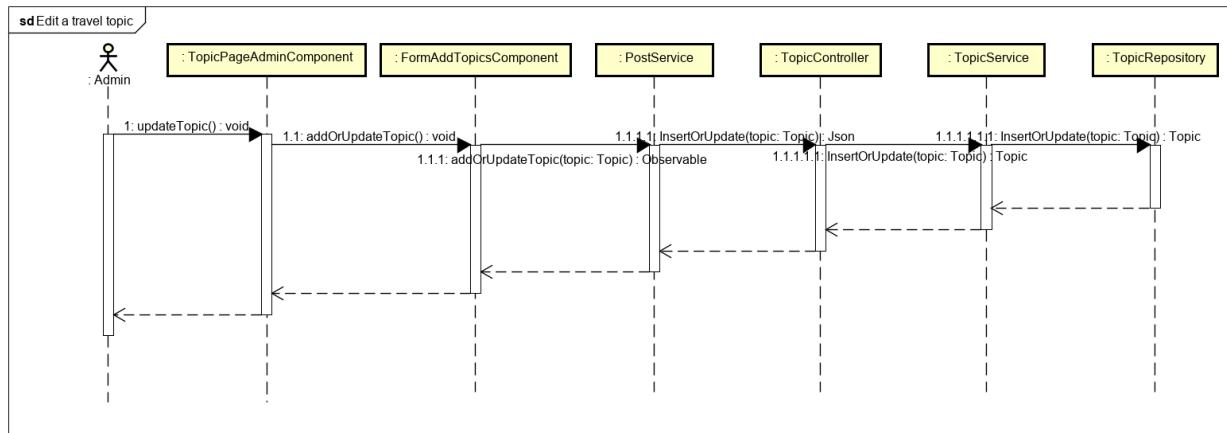


Figure 4-139: Administrator edit a travel topic sequence diagram

4.3.4.44 Delete travel topics

Screen Design

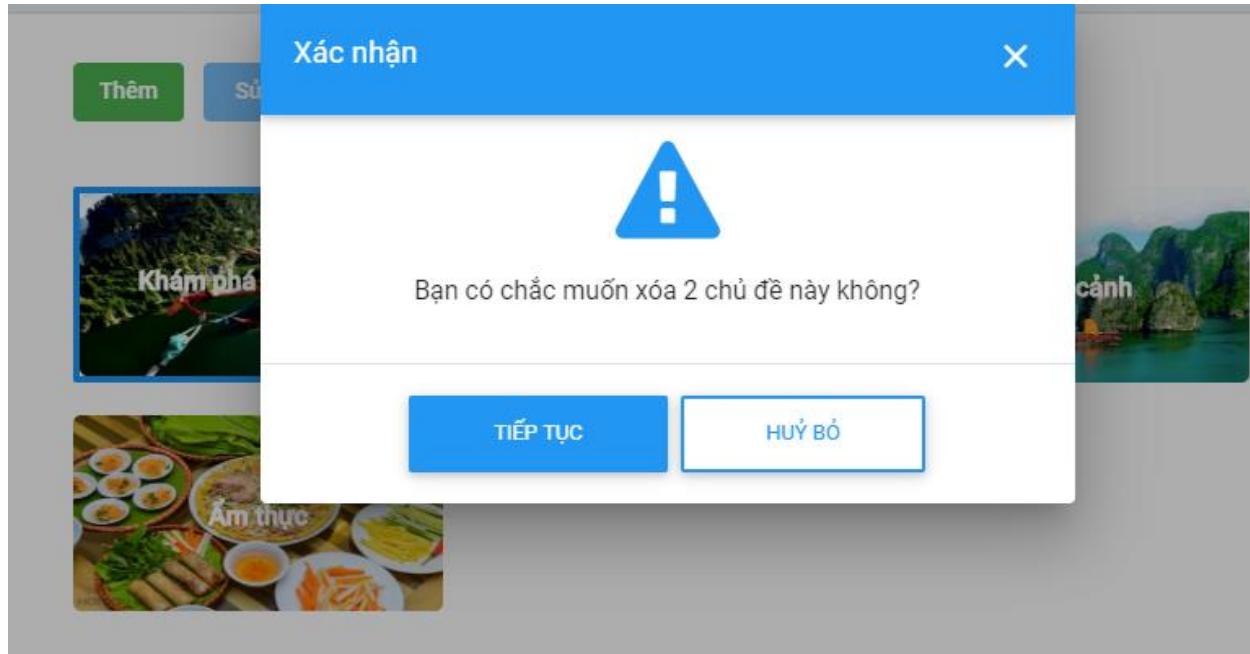


Figure 4-140: Delete travel topics screen design

Class Diagram

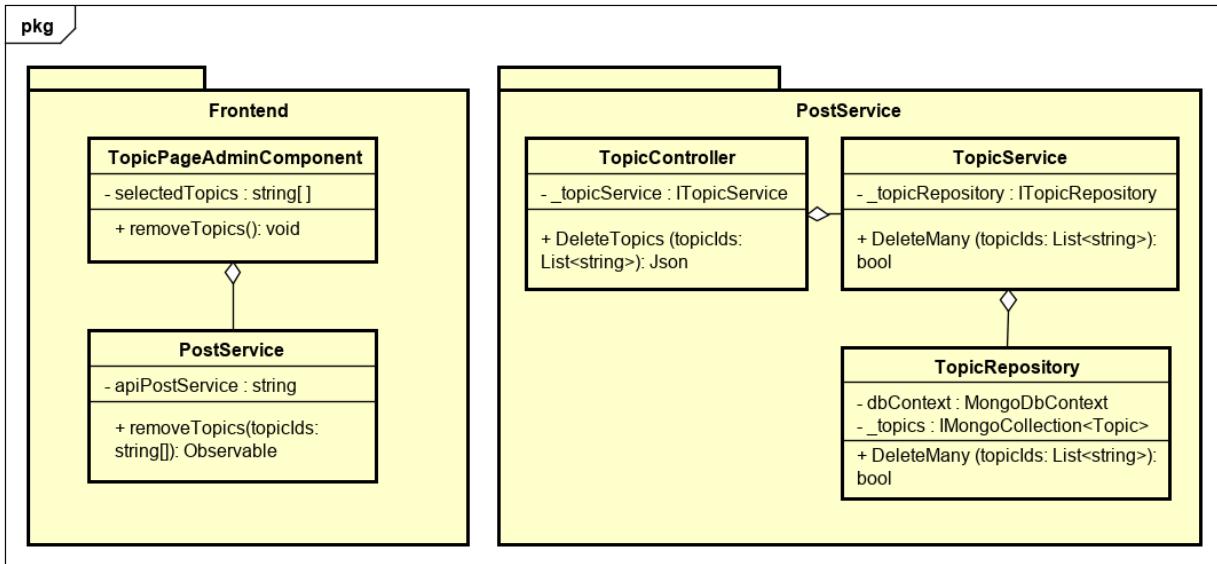


Figure 4-141: Administrator delete travel topics class diagram

Class Specification

TopicPageAdminComponent

TopicPageAdminComponent			
Physical address	src/app/admin/pages/dashboard-page/components/topic-page-admin/topic-page-admin.component.ts		
Base class	Class		
Attributes			
No	Name	Type	Description
1	selectedTopics	String []	
Operation			
No	Name	Type	Description
1	removeTopics	void	

PostService

PostService			
Physical address	src/app/core/services/post-service/post.service.ts		
Base class	Class		
Attributes			
No	Name	Type	Description
1	apiPostService	String	
Operation			
No	Name	Return Type	Description
1	removeTopics	Observable	

TopicController

TopicController			
Physical address	src/Services/PostService/PostService/Controllers/TopicController.cs		
Base class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_topicService	ITopicService	
Operation			
No	Name	Type	Description
1	DeleteTopics	Json	

TopicService

TopicService			
Physical address	src/Services/PostService/PostService/Service/TopicService.cs		
Base class	ITopicService		
Attributes			
No	Name	Type	Description
1	_topicRepository	ITopicRepository	
Operation			
No	Name	Type	Description
1	DeleteMany	Bool	

TopicRepository

TopicRepository			
Physical address	src/Services/PostService/PostService/Repository/TopicRepository.cs		
Base class	ITopicRepository		
Attributes			
No	Name	Type	Description
1	dbContext	MongoDbContext	
2	_topics	IMongoCollection<Topic>	
Operation			
No	Name	Type	Description
1	DeleteMany	Bool	

Sequence Diagram

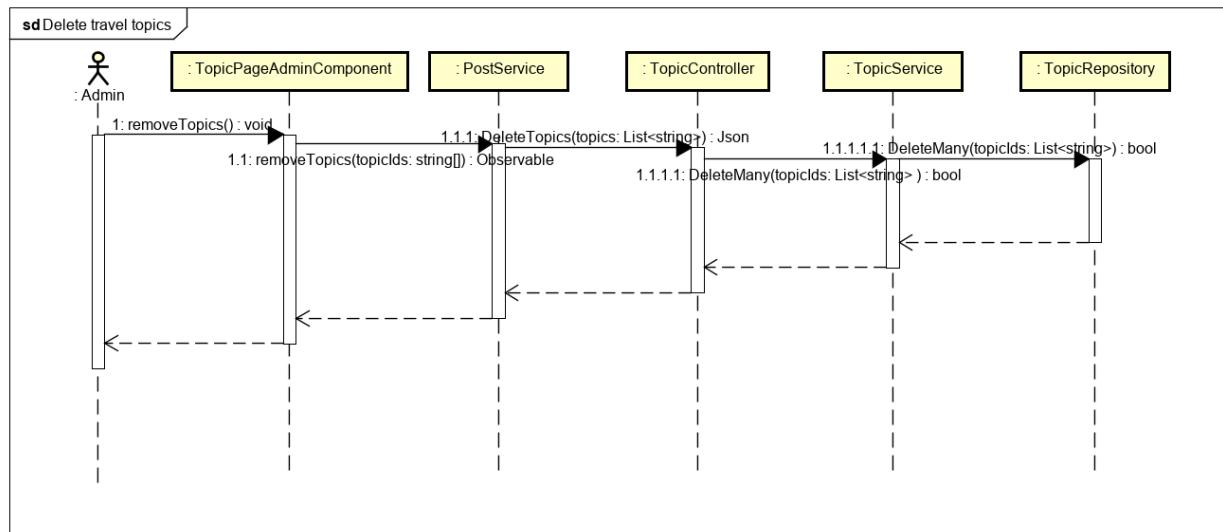


Figure 4-142: Administrator delete travel topics sequence diagram

4.3.4.45 View statistic of users

Screen Design

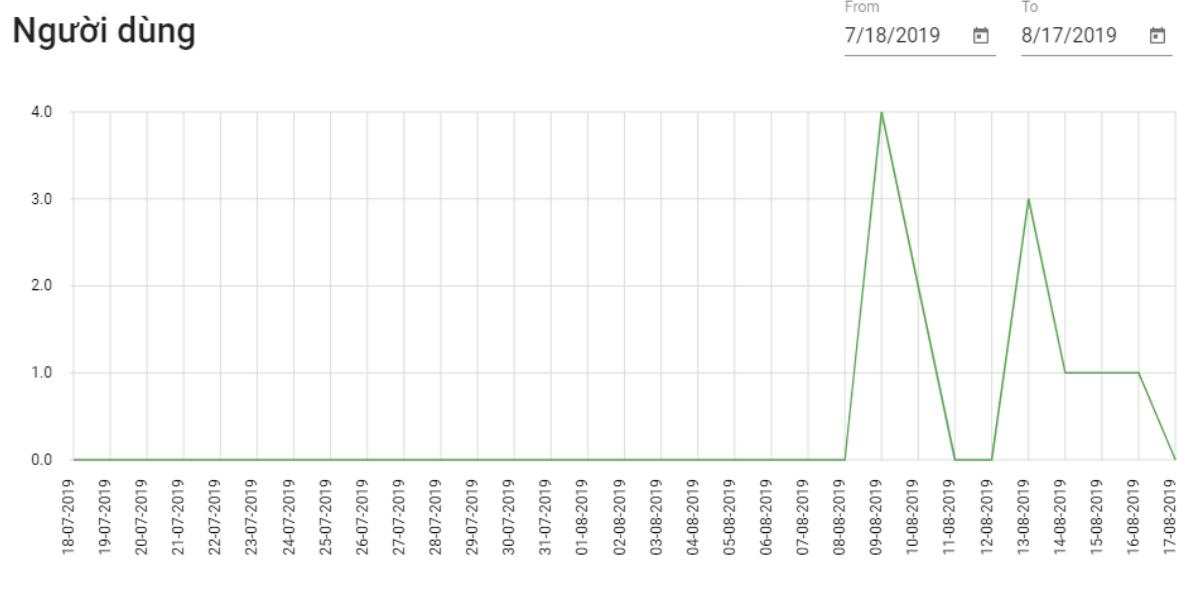


Figure 4-143: View statistic of user screen design

Class Diagram

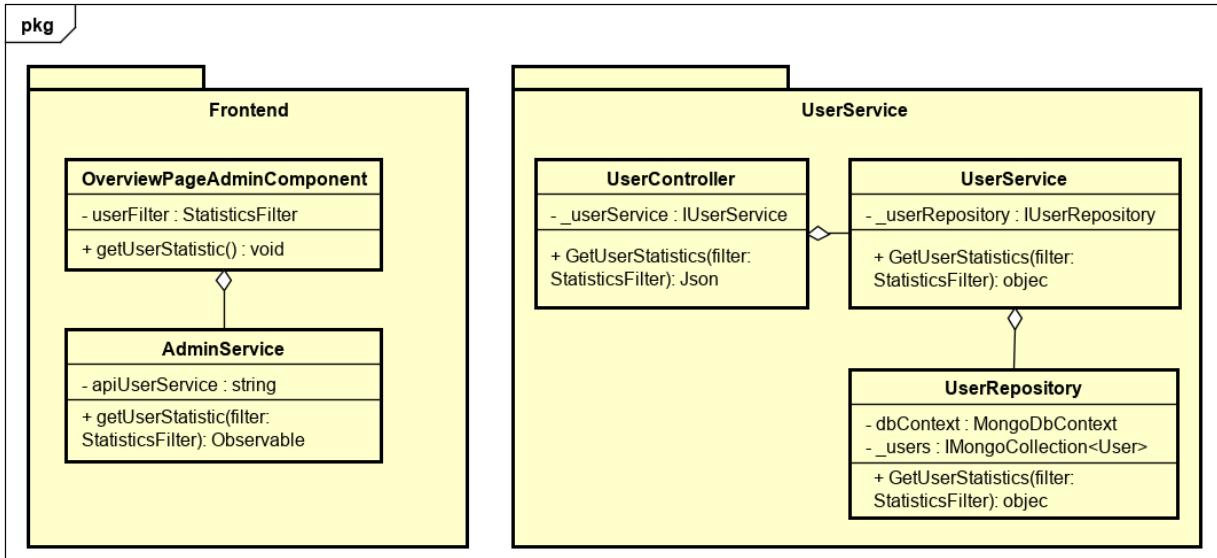


Figure 4-144: Administrator view statistic of user class diagram

Class Specification

OverviewPageAdminComponent

OverviewPageAdminComponent			
Physical address	src/app/admin/pages/dashboard-page/components/overview-page-admin/overview-page-admin.component.ts		
Base class	Class		
Attributes			
No	Name	Type	Description
1	userFilter	StatisticsFilter	
Operation			
No	Name	Type	Description
1	getUserStatistic	void	

AdminService

AdminService			
Physical address	src/app/admin/services/admin-service/admin.service.ts		
Base class	Class		
Attributes			
No	Name	Type	Description
1	apiUserService	String	
Operation			
No	Name	Return Type	Description
1	getUserStatistic	Observable	

UserController

UserController			
Physical address	src/Services/UserService/UserService/Controllers/UserController.cs		
Base class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_userService	IUserService	
Operation			
No	Name	Type	Description
1	GetUserStatistics	Json	

UserService

UserService			
Physical address	src/Services/UserService/UserService/Service/UserService.cs		
Base class	IUserService		
Attributes			
No	Name	Type	Description
1	_userRepository	IUserRepository	
Operation			
No	Name	Type	Description
1	GetUserStatistics	Object	

UserRepository

UserRepository			
Physical address	src/Services/UserService/UserService/Repository/UserRepository.cs		
Base class	IUserRepository		
Attributes			
No	Name	Type	Description
1	dbContext	MongoDbContext	
2	_users	ImongoCollection<User>	
Operation			
No	Name	Type	Description
1	GetUserStatistics	Object	

Sequence Diagram

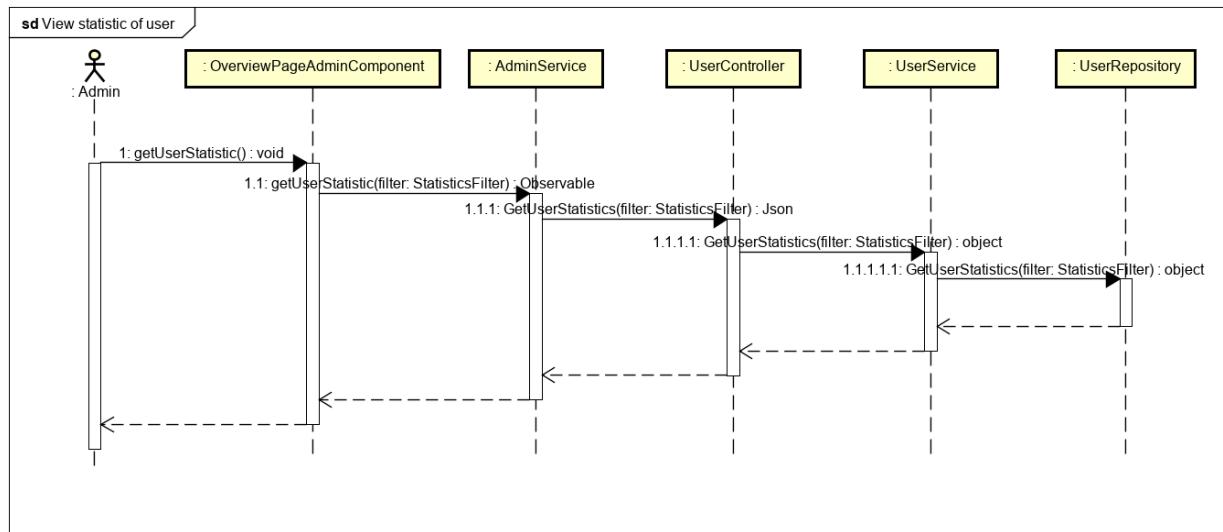


Figure 4-145: Administrator view statistic of user sequence diagram

4.3.4.46 View statistic of posts

Screen Design

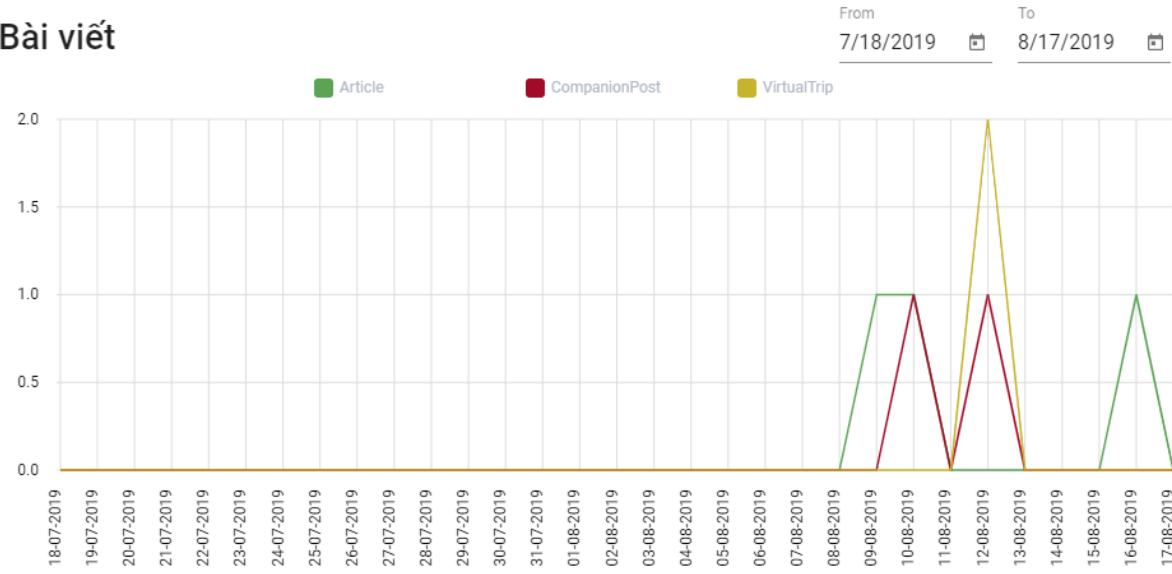


Figure 4-146: View statistic of posts screen design

Class Diagram

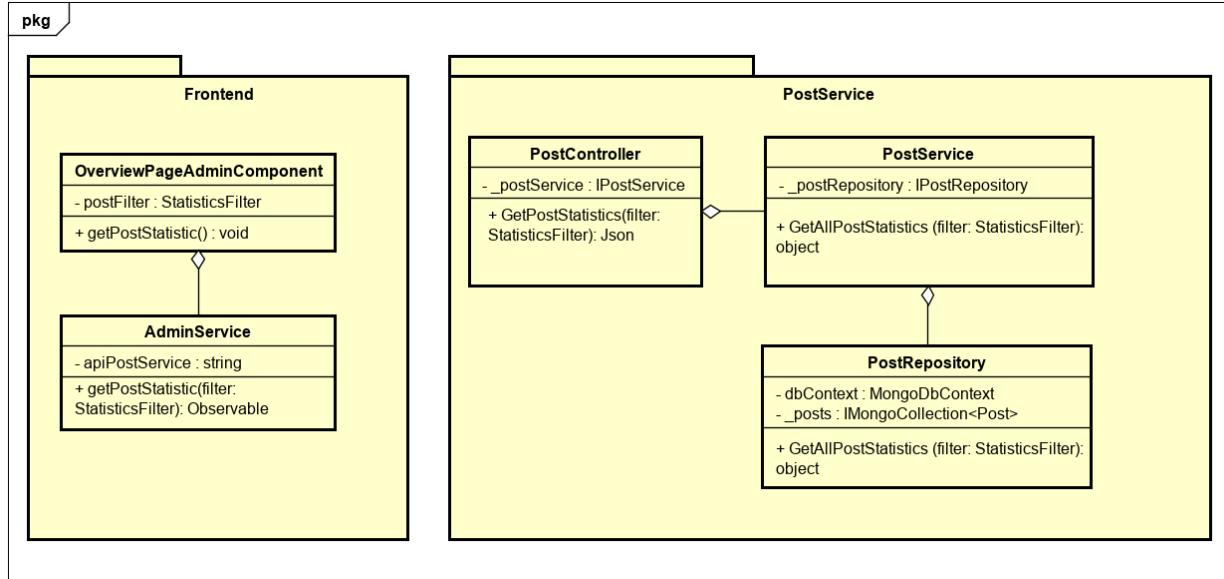


Figure 4-147: Administrator view statistic of posts class diagram

Class Specification

OverviewPageAdminComponent

OverviewPageAdminComponent			
Physical address	src/app/admin/pages/dashboard-page/components/overview-page-admin/overview-page-admin.component.ts		
Base class	Class		
Attributes			
No	Name	Type	Description
1	postFilter	StatisticsFilter	
Operation			
No	Name	Type	Description
1	getPostStatistic	void	

AdminService

AdminService			
Physical address	src/app/admin/services/admin-service/admin.service.ts		
Base class	Class		
Attributes			
No	Name	Type	Description
1	apiPostService	String	
Operation			
No	Name	Return Type	Description
1	getPostStatistic	Observable	

PostController

PostController			
Physical address	src/Services/PostService/PostService/Controllers/PostController.cs		
Base class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_postService	IPostService	
Operation			
No	Name	Type	Description
1	GetPostStatistics	Json	

PostService

PostService			
Physical address	src/Services/PostService/PostService/Service/PostService.cs		
Base class	IPostService		
Attributes			
No	Name	Type	Description
1	_postRepository	IPostRepository	
Operation			
No	Name	Type	Description
1	GetAllPostStatistics	Object	

PostRepository

PostRepository			
Physical address	./src/Services/ PostService/PostService /Repository/PostRepository.cs		
Base class	IPostRepository		
Attributes			
No	Name	Type	Description
1	dbContext	MongoDbContext	
2	_posts	IMongoCollection<Post>	
Operation			
No	Name	Type	Description
1	GetAllPostStatistics	Object	

Sequence Diagram

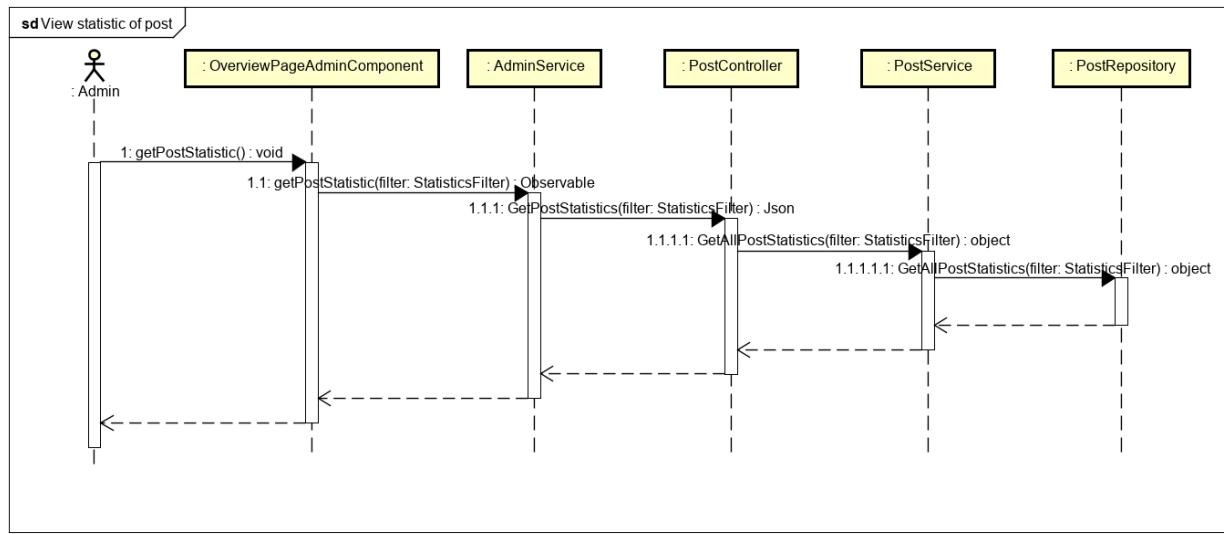


Figure 4-148: Administrator view statistic of posts sequence diagram

4.3.4.47 Search posts

Screen Design

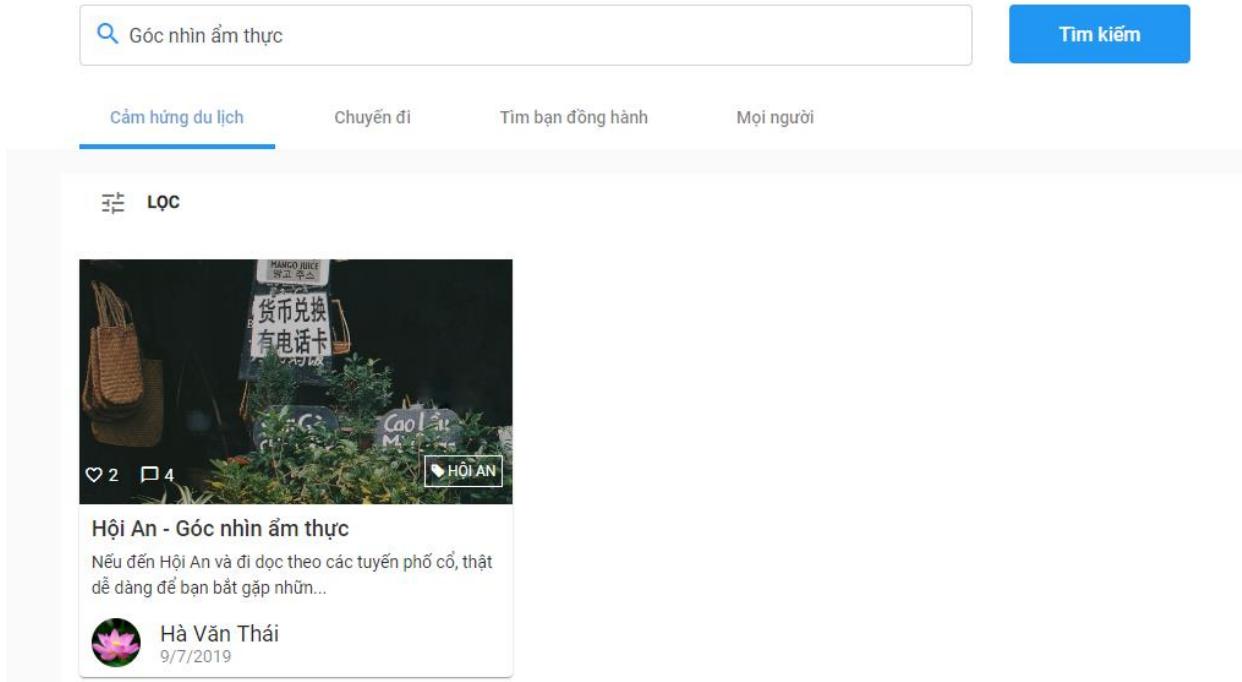


Figure 4-149: Search posts screen design

Class Diagram

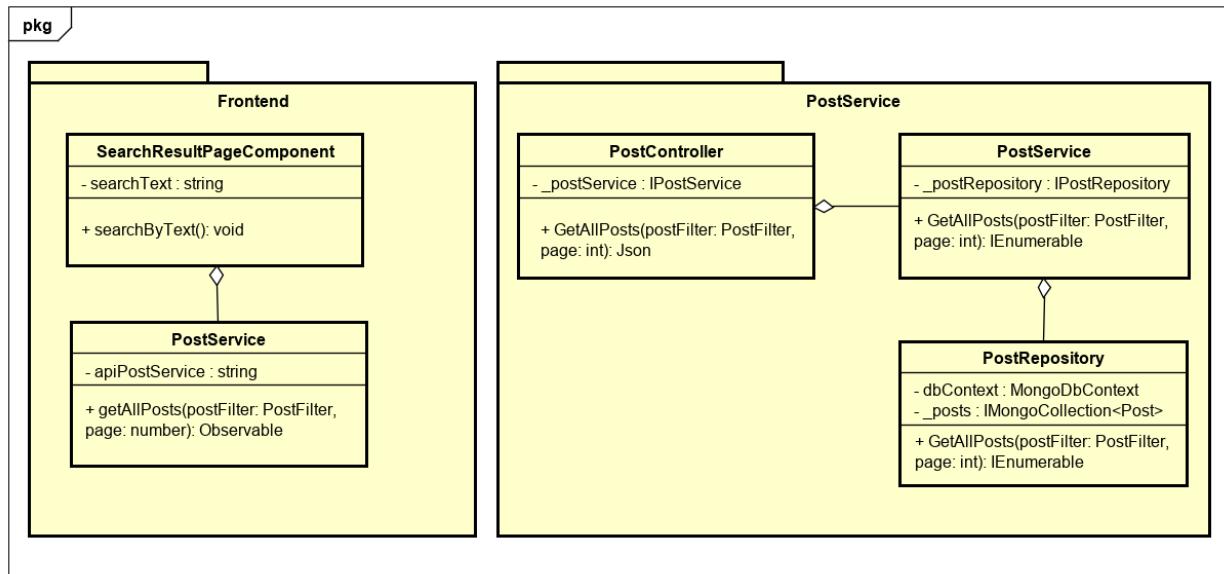


Figure 4-150: Search posts class diagram

Class Specification

SearchResultPageComponent

SearchResultPageComponent			
Physical address	src/app/pages/search-result-page/search-result-page.component.ts		
Base class	Class		
Attributes			
No	Name	Type	Description
1	searchText	String	
Operation			
No	Name	Type	Description
1	searchByText	void	

PostService

PostService			
Physical address	src/app/core/services/post-service/post.service.ts		
Base class	Class		
Attributes			
No	Name	Type	Description
1	apiPostService	String	
Operation			
No	Name	Return Type	Description
1	getAllPosts	Observable	

PostController

PostController			
Physical address	src/Services/PostService/PostService/Controllers/PostController.cs		
Base class	ControllerBase		
Attributes			
No	Name	Type	Description
1	_postService	IPostService	
Operation			
No	Name	Type	Description
1	GetAllPosts	Json	

PostService

PostService			
Physical address	./src/Services/ PostService/PostService/Service/PostService.cs		
Base class	IPostService		
Attributes			
No	Name	Type	Description
1	_postRepository	IPostRepository	
Operation			
No	Name	Type	Description
1	GetAllPosts	IEnumerable	

PostRepository

PostRepository			
Physical address	./src/Services/ PostService/PostService /Repository/PostRepository.cs		
Base class	IPostRepository		
Attributes			
No	Name	Type	Description
1	dbContext	MongoDbContext	
2	_posts	ImongoCollection <Post>	
Operation			
No	Name	Type	Description
1	GetAllPost	IEnumerable	

Sequence Diagram

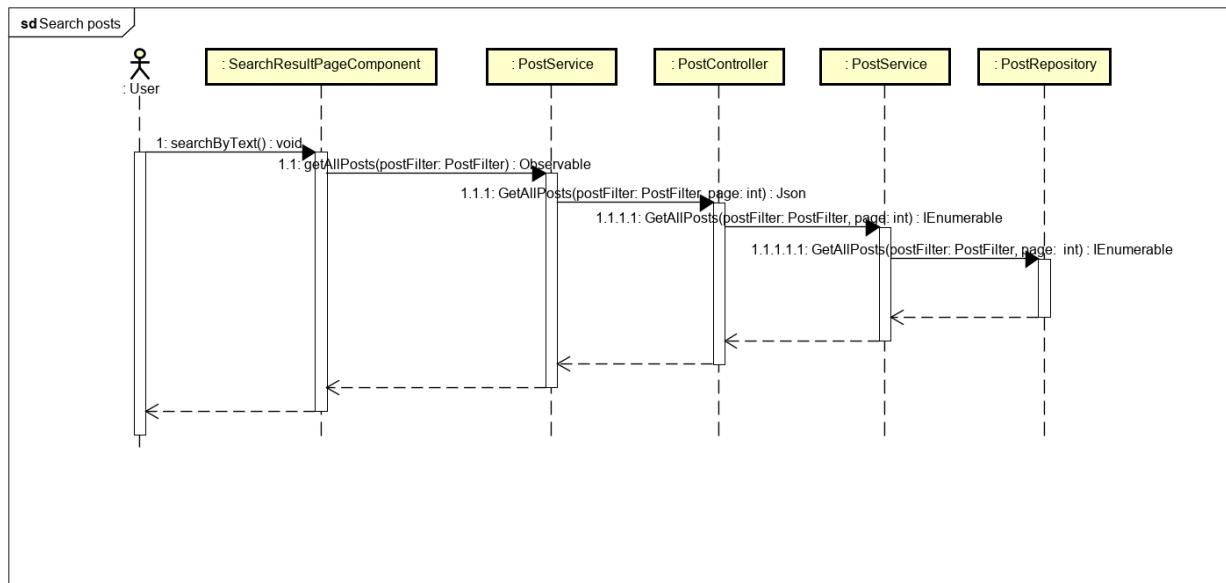


Figure 4-151: Search posts sequence diagram

4.3.4.48 Search a user

Screen Design

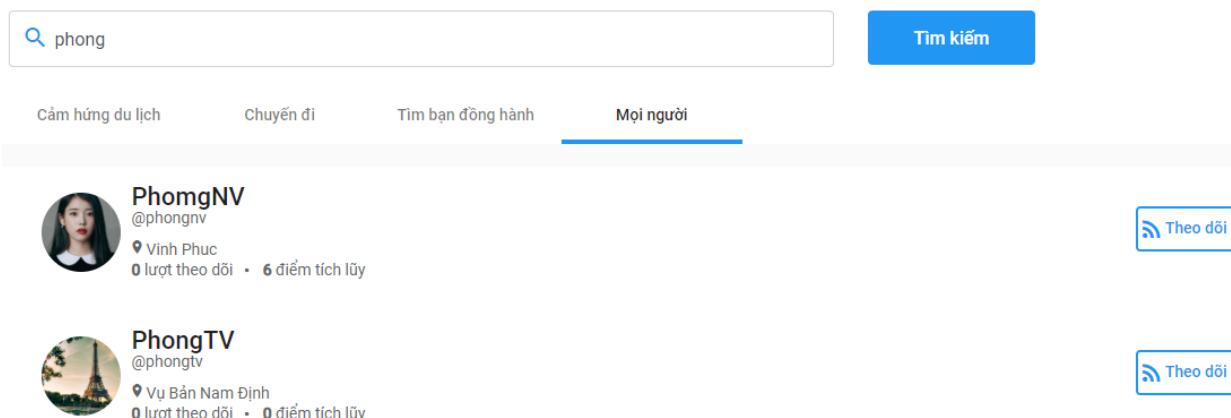


Figure 4-152: Search a user class diagram

Class Diagram

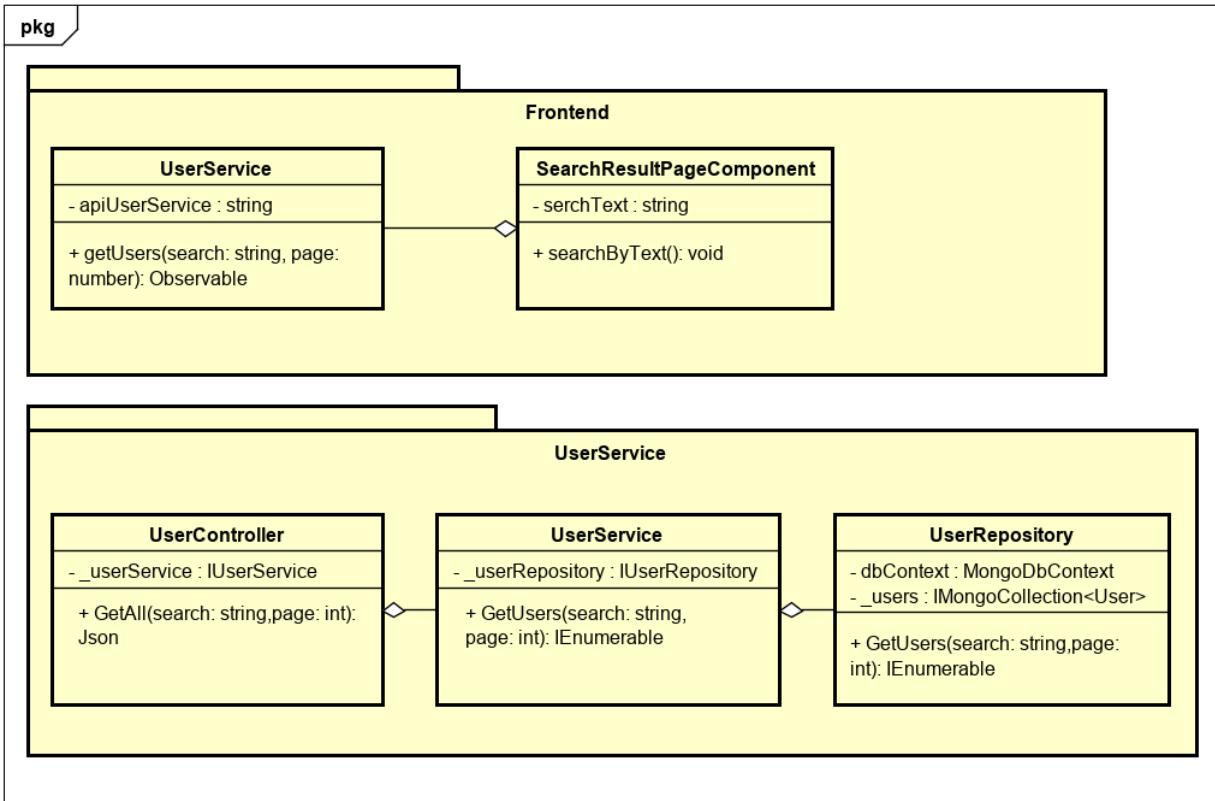


Figure 4-153: Search a user class diagram

Class Specification

SearchResultPageComponent

SearchResultPageComponent			
Physical address	src/app/pages/search-result-page/search-result-page.component.ts		
Base class	Class		
Attributes			
No	Name	Type	Description
1	serchText	String	
Operation			
No	Name	Type	Description
1	searchByText	void	

UserService

UserService			
Physical address	src/app/core/services/user-service/user.service.ts		
Base class	Class		
Attributes			
No	Name	Type	Description

1	apiUserService	String	
Operation			
No	Name	Return Type	Description
1	getUsers	Observable	

UserController

UserController			
Physical address			src/Services/UserService/UserService/Controllers/UserController.cs
Base class			ControllerBase
Attributes			
No	Name	Type	Description
1	_userService	IUserService	
Operation			
No	Name	Type	Description
1	GetAll	Json	

UserService

UserService			
Physical address			src/Services/UserService/UserService/Service/UserService.cs
Base class			IUserService
Attributes			
No	Name	Type	Description
1	_userRepository	IUserRepository	
Operation			
No	Name	Type	Description
1	GetUsers	IEnumerable	

UserRepository

UserRepository			
Physical address			src/Services/UserService/UserService/Repository/UserRepository.cs
Base class			IUserRepository
Attributes			
No	Name	Type	Description
1	dbContext	MongoDbContext	
2	_users	ImongoCollection<User>	
Operation			
No	Name	Type	Description
1	GetUsers	IEnumerable	

Sequence Diagram

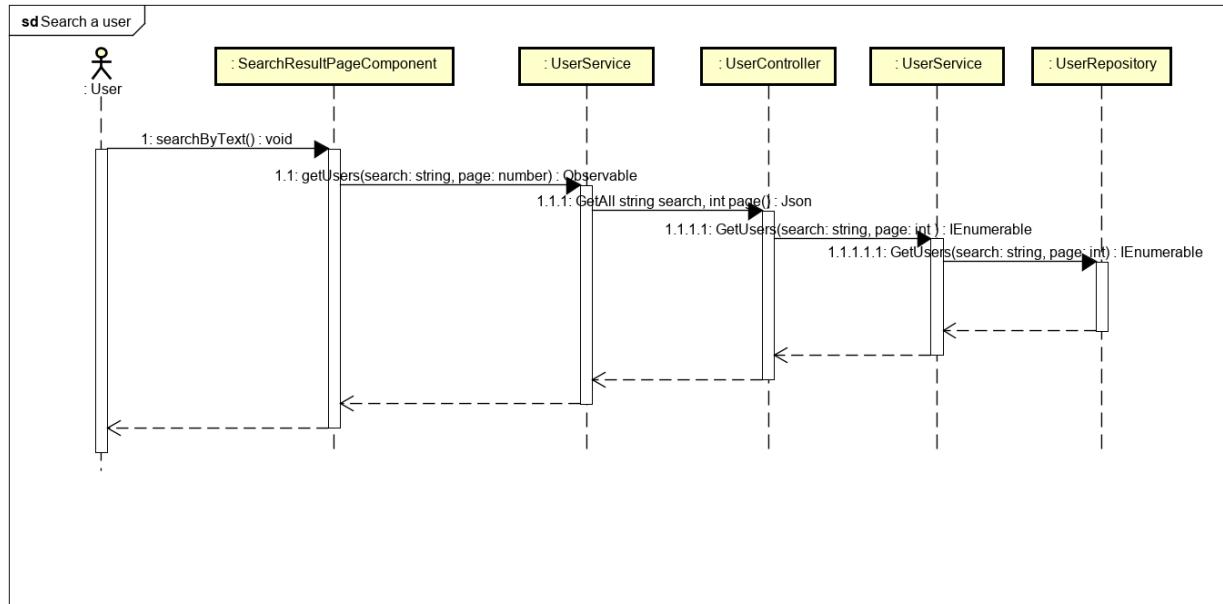


Figure 4-154: Search a user sequence diagram

4.3.4.49 View popular articles

Screen design

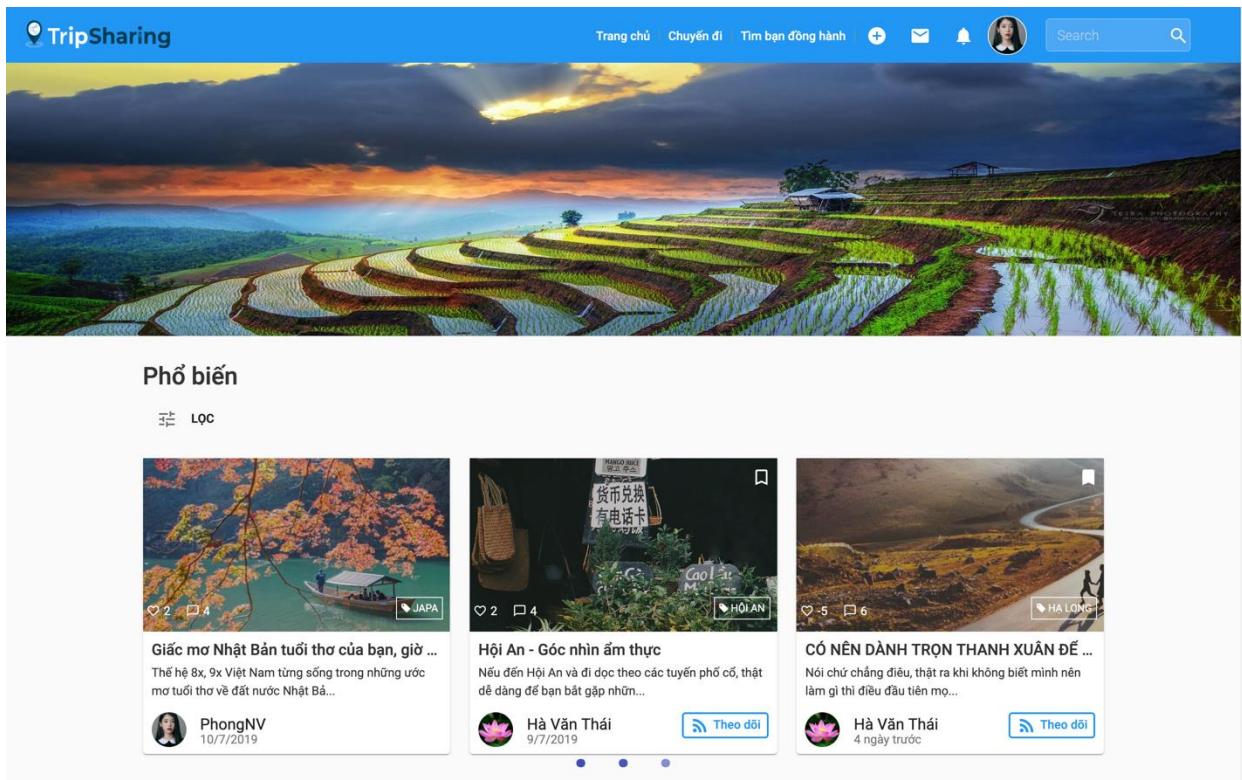


Figure 4-155: View popular articles screen

Class Diagram

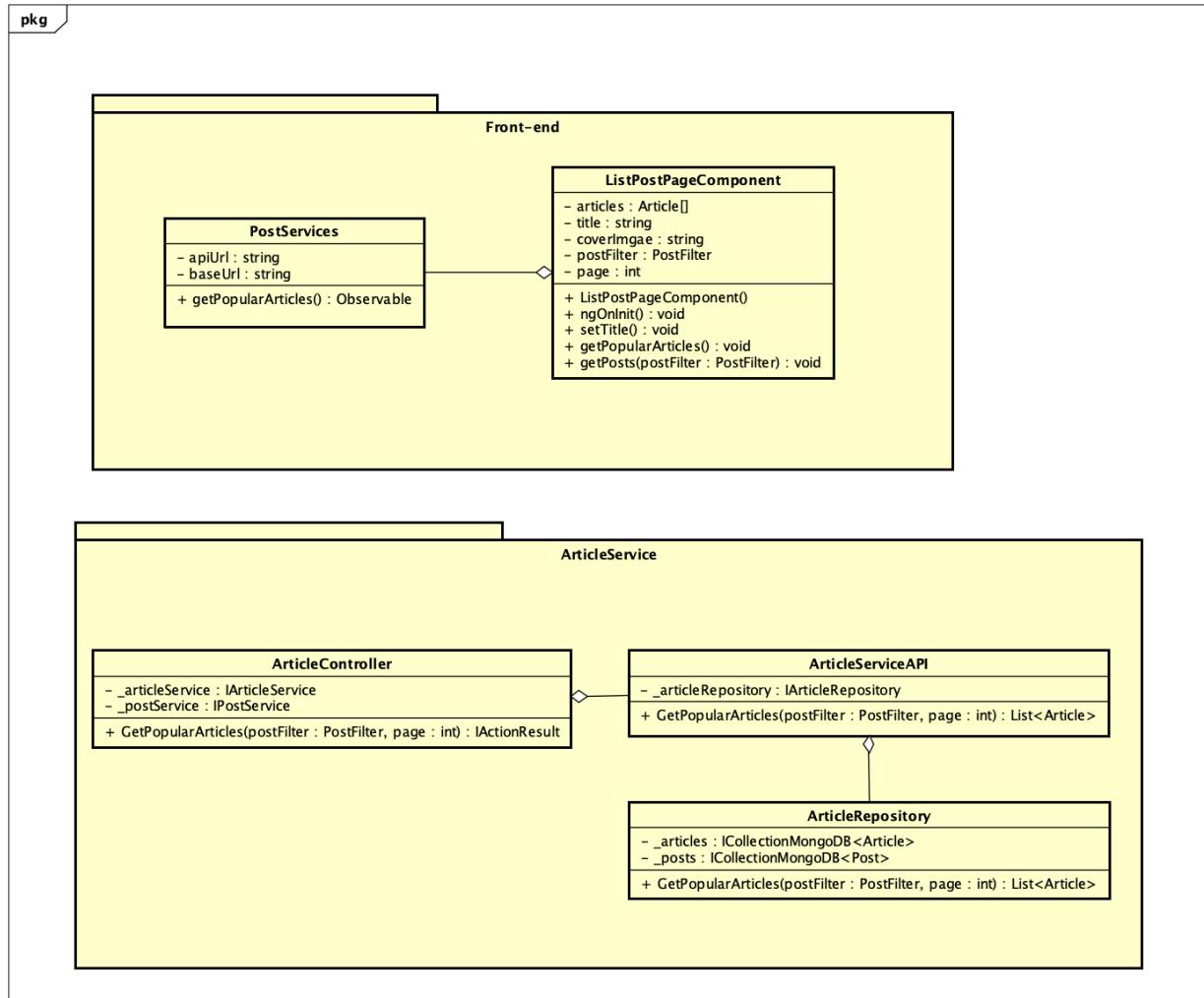


Figure 4-156: View Popular Articles Class Diagram

Class Specification

ListPostPageComponent

ListPostPageComponent			
Physical address	src\app\pages\list-post-page\list-post-page.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	articles	Article[]	
2	title	String	
3	coverImage	String	
4	postFilter	PostFilter	
5	page	Int	
Operations			

No	Name	Return Type	Description
1	ListPostPageComponent	void	Constructor execute function setTitle()
2	ngOnInit	void	Life circle hook in Angular execute function getPosts()
3	setTitle	void	Set page's title
4	getPopularArticles	void	Get popular articles posts
5	getPosts		Get articles posts execute function getPopularArticles()

PostServices

PostServices			
Physical address	src\app\core\services\post-service\post.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	apiUrl	String	
2	baseUrl	String	
Operations			
No	Name	Return Type	Description
1	getPopularArticles	Observable	Call api get data from server

ArticleController

ArticleController			
Physical address	Src\Services\PostService\PostService\Controllers\ArticleController.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	_articleService	IArticleService	
2	_postService	IPostService	
Operations			
No	Name	Return Type	Description
1	GetPopularArticles	IActionResult	Get popular article posts and return data to front-end.

ArticleServiceAPI

ArticleServiceAPI

Physical address	Src\Services\PostService\PostService\Services\ArticleServiceAPI.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	_articleRepository	IArticleRepository	
Operations			
No	Name	Return Type	Description
1	GetPopularArticles	List<Object>	Get popular article posts

ArticleRepository

ArticleRepository			
Physical address	Src\Services\PostService\PostService\Repositories\ArticleRepository.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	_articles	ICollectionMongoDB<Article>	
2	_posts	ICollectionMongoDB <Post>	
Operations			
No	Name	Return Type	Description
1	GetPopularArticles	List<Article>	Get popular article posts from database

Sequence Diagram

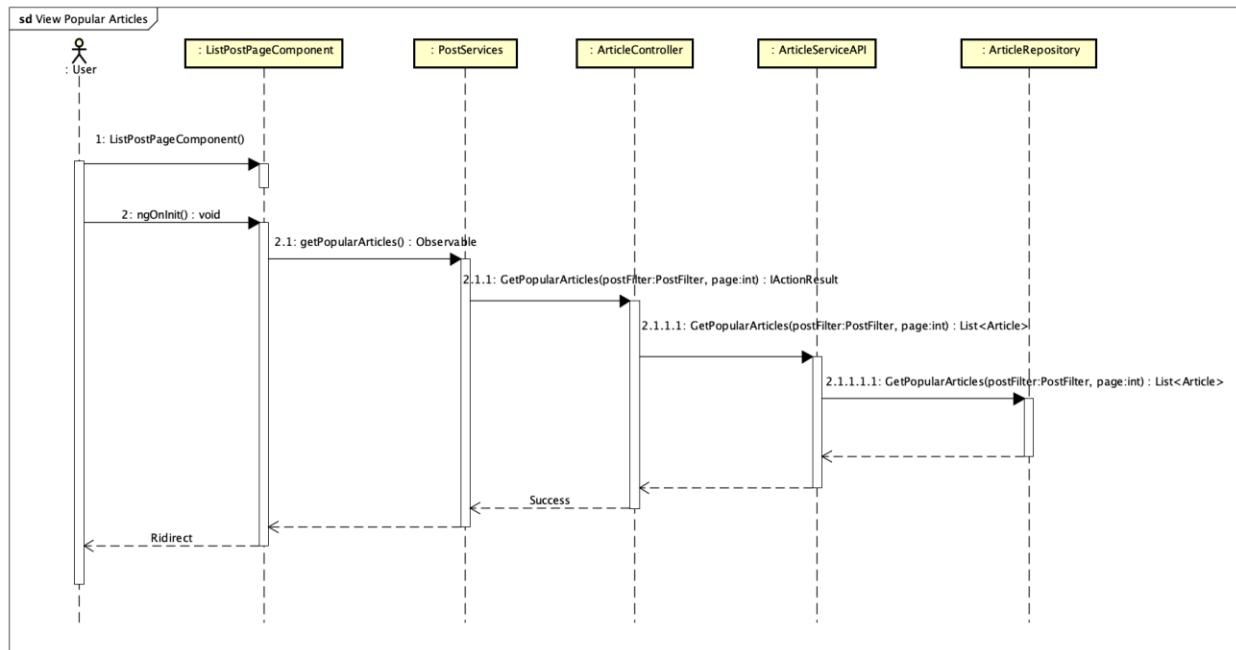


Figure 4-157: View Popular Articles Sequence Diagram

4.3.4.50 View recently articles

Screen design

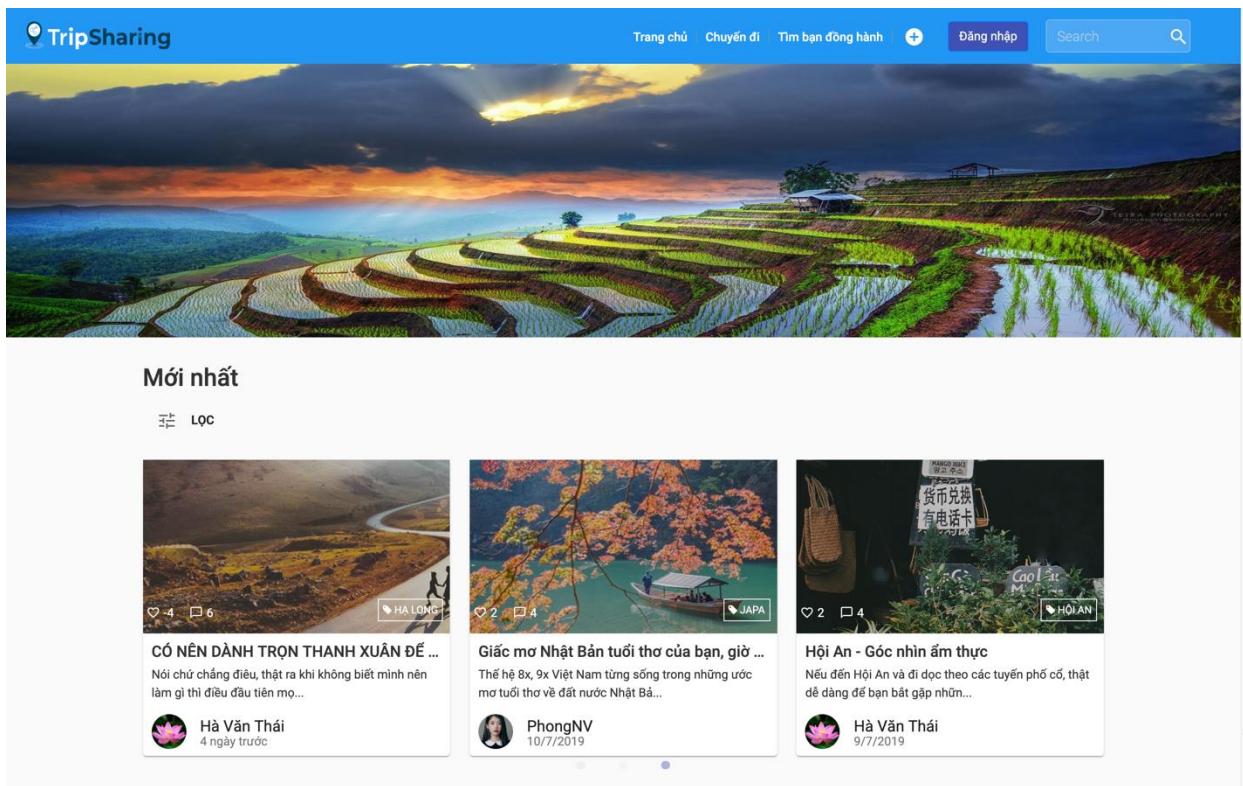


Figure 2-158: View Recently Articles screen

Class Diagram

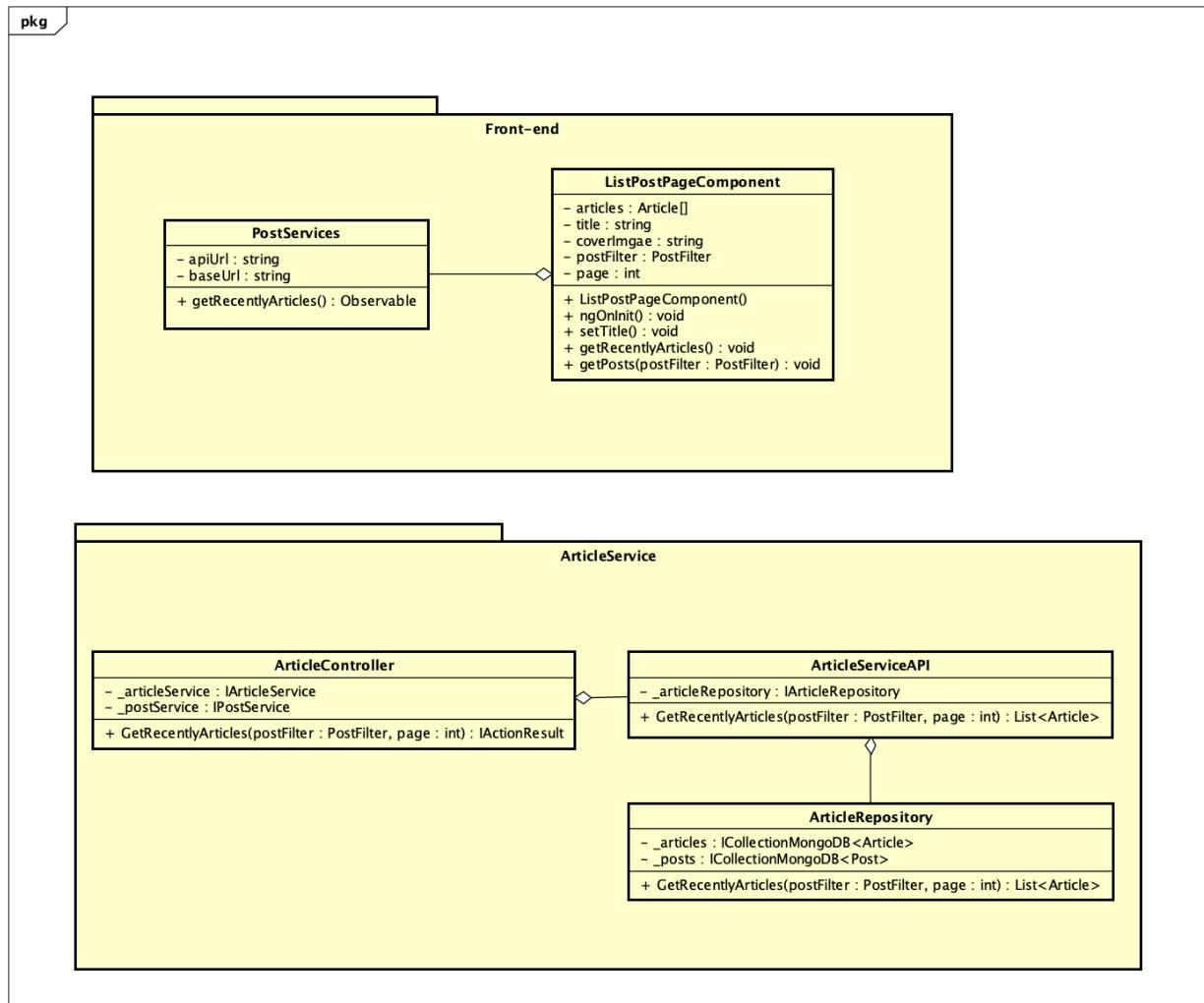


Figure 4-159: View recently articles Class Diagram

Class Specification

ListPostPageComponent

ListPostPageComponent			
Physical address	src\app\pages\list-post-page\list-post-page.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	articles	Article[]	
2	title	String	
3	coverImage	String	
4	postFilter	PostFilter	
5	page	Int	
Operations			
No	Name	Return Type	Description

1	ListPostPageComponent	void	Contructor execexcute function setTitle()
2	ngOninit	void	Life circle hook in Angular execute function getPosts()
3	setTitle	void	Set page's title
4	getRecentlyArticles	void	Get popular articles posts
5	getPosts		Get articles posts execute function getRecentlyArticles()

PostServices

PostServices			
Physical address	src\app\core\services\post-service\post.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	apiUrl	String	
2	baseUrl	String	
Operations			
No	Name	Return Type	Description
1	getRecentlyArticles	Observable	Call api get data from server

ArticleController

ArticleController			
Physical address	Src\Services\PostService\PostService\Controllers\ArticleController.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	_articleService	IArticleService	
2	_postService	IPostService	
Operations			
No	Name	Return Type	Description
1	GetRecentlyArticles	IActionResult	Get recently article posts and return data to font-end.

ArticleServiceAPI

ArticleServiceAPI			
Physical address	Src\Services\PostService\PostService\Services\ArticleServiceAPI.cs		

Base Class	Class		
Attributes			
No	Name	Type	Description
1	_articleRepository	IArticleRepository	
Operations			
No	Name	Return Type	Description
1	GetRecentlyArticles	List<Article >	Get recently article posts

ArticleRepository

ArticleRepository			
Physical address	Src\Services\PostService\PostService\Repositories\ArticleRepository.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	_articles	ICollectionMongoDB<Article >	
2	_posts	ICollectionMongoDB<Post>	
Operations			
No	Name	Return Type	Description
1	GetRecentlyArticles	List<Article >	Get recently article posts from database

Sequence Diagram

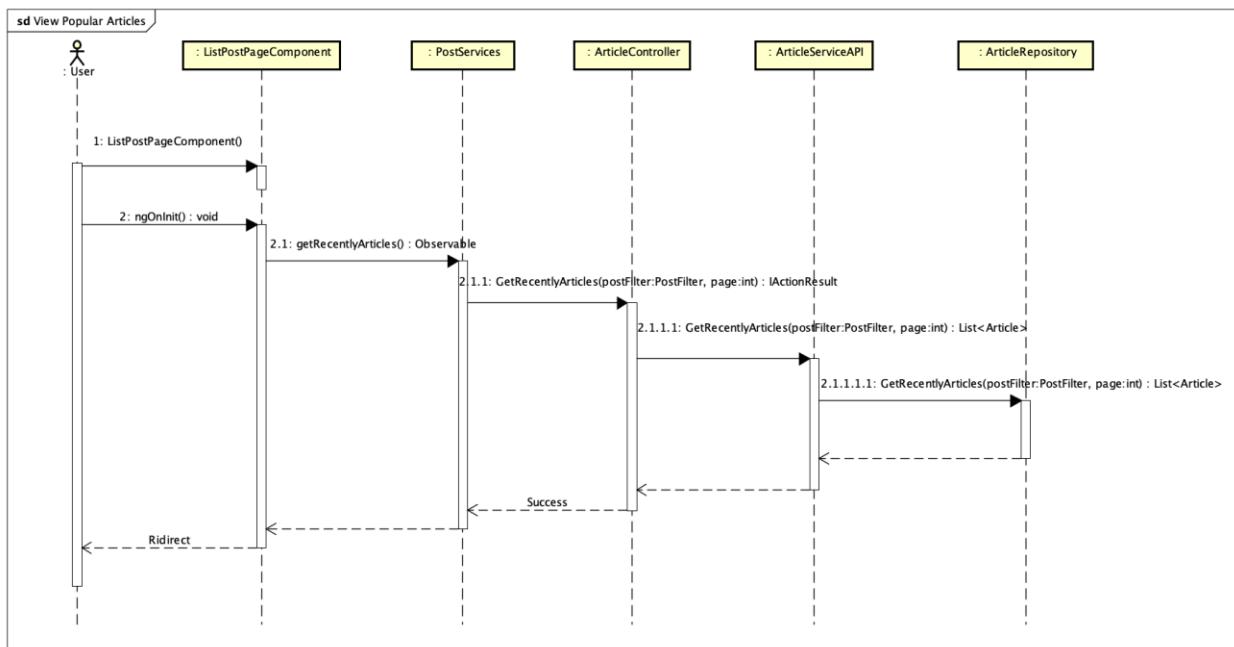


Figure 4-160: View Recently Articles Sequence Diagram

4.3.4.51 View a post detail

Screen design

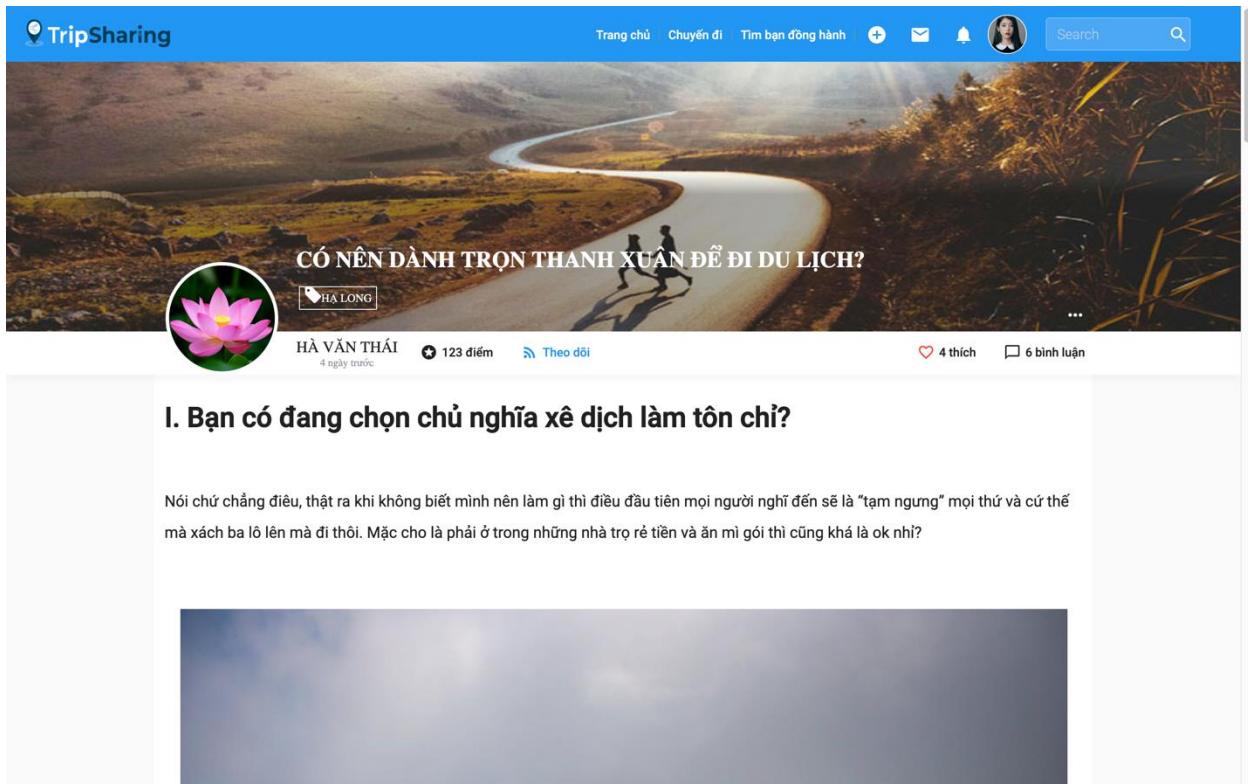


Figure 4-161: View a Post Detail screen

Class Diagram

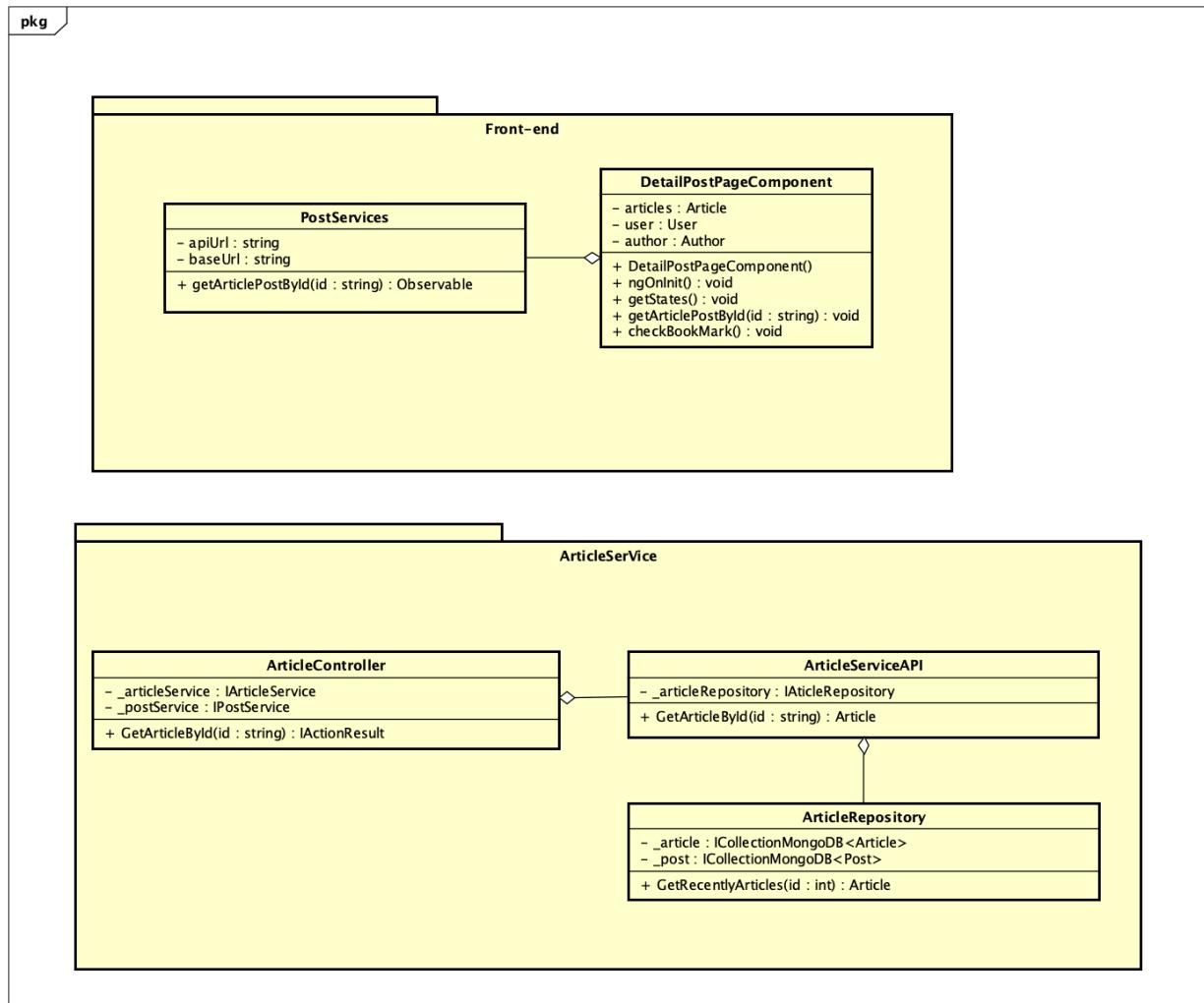


Figure 4-162: View a Post Detail Class Diagram

Class Specification

DetailPostPageComponent

ListPostPageComponent			
Physical address	src\app\pages\list-post-page\detail-post-page.component.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	article	Article	
2	user	User	
3	author	Author	
Operations			
No	Name	Return Type	Description
1	DetailPostPageComponent	void	Contructor

2	ngOninit	void	Life circle hook in Angular execute function getArticlePostById()
3	getState	void	Get state of post
4	getArticlePostById	void	Get detail article post by id

PostServices

PostServices			
Physical address	src\app\core\services\post-service\post.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	apiUrl	String	
2	baseUrl	string	
Operations			
No	Name	Return Type	Description
1	getArticlePostById	Observable	Call api get data from server

ArticleController

ArticleController			
Physical address	Src\Services\PostService\PostService\Controllers\ArticleController.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	_articleService	IArticleService	
2	_postService	IPostService	
Operations			
No	Name	Return Type	Description
1	GetArticlePostById	IActionResult	Get detail article post by id and return data to font-end.

ArticleServiceAPI

ArticleServiceAPI			
Physical address	Src\Services\PostService\PostService\Services\ArticleServiceAPI.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description

1	_articleRepository	IArticleRepository	
Operations			
No	Name	Return Type	Description
1	GetArticlePostById	Object	Get detail article post by id

ArticleRepository

ArticleRepository			
Physical address	Src\Services\PostService\PostService\Repositories\ArticleRepository.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	_articles	ICollectionMongoDB<Article>	
2	_posts	ICollectionMongoDB<Post>	
Operations			
No	Name	Return Type	Description
1	GetArticlePostById	Object	Get detail article post by id from database

Sequence Diagram

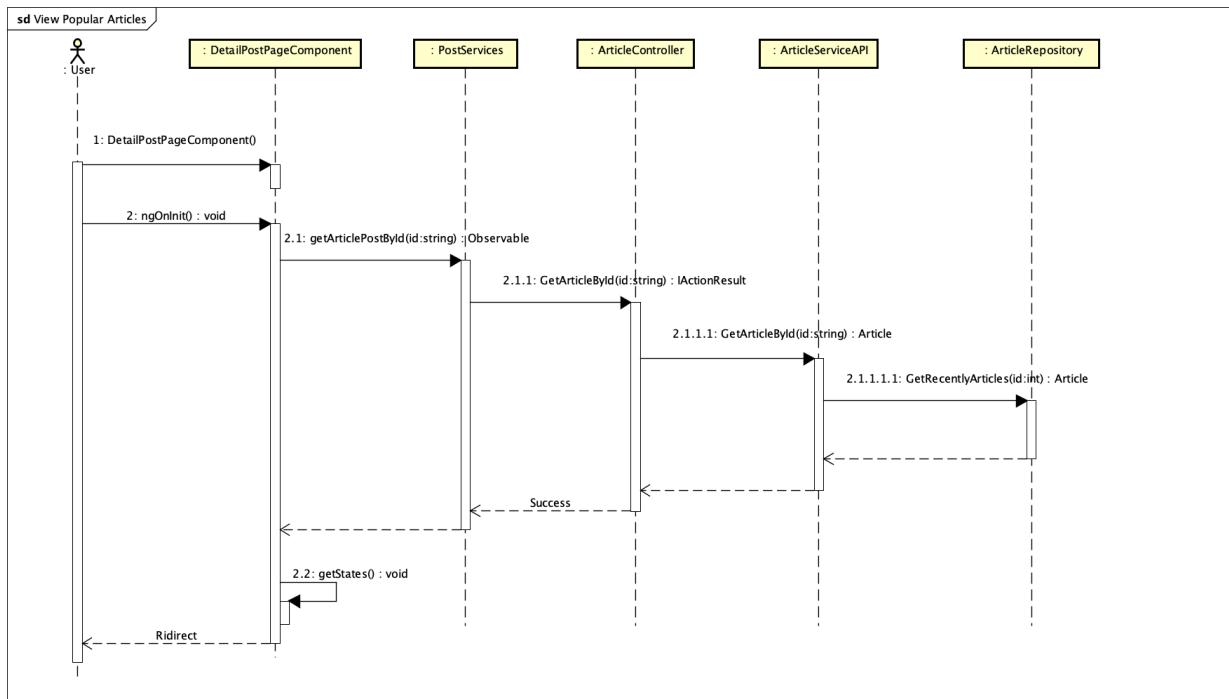


Figure 4-163: View detail a post sequence diagram

4.3.4.52 View user's created posts

Screen design

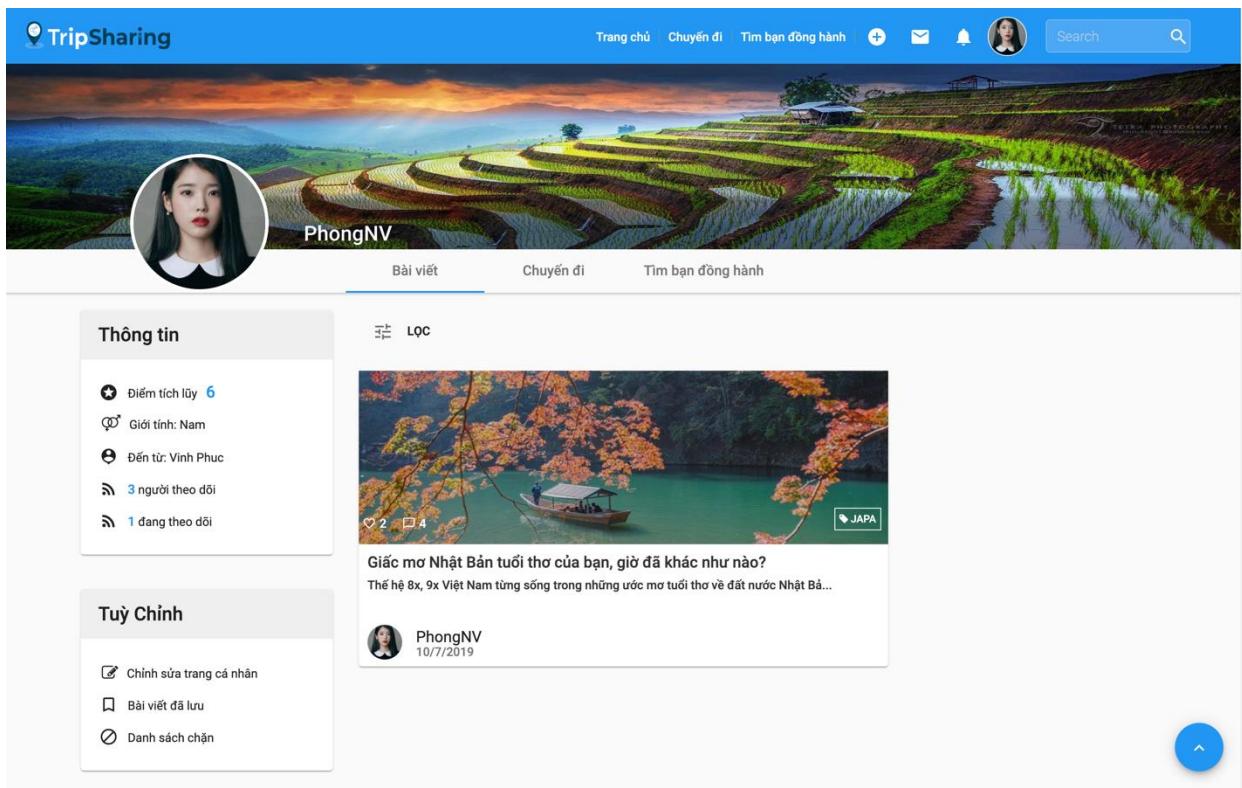


Figure 4-164: View user's created posts screen

Class Diagram

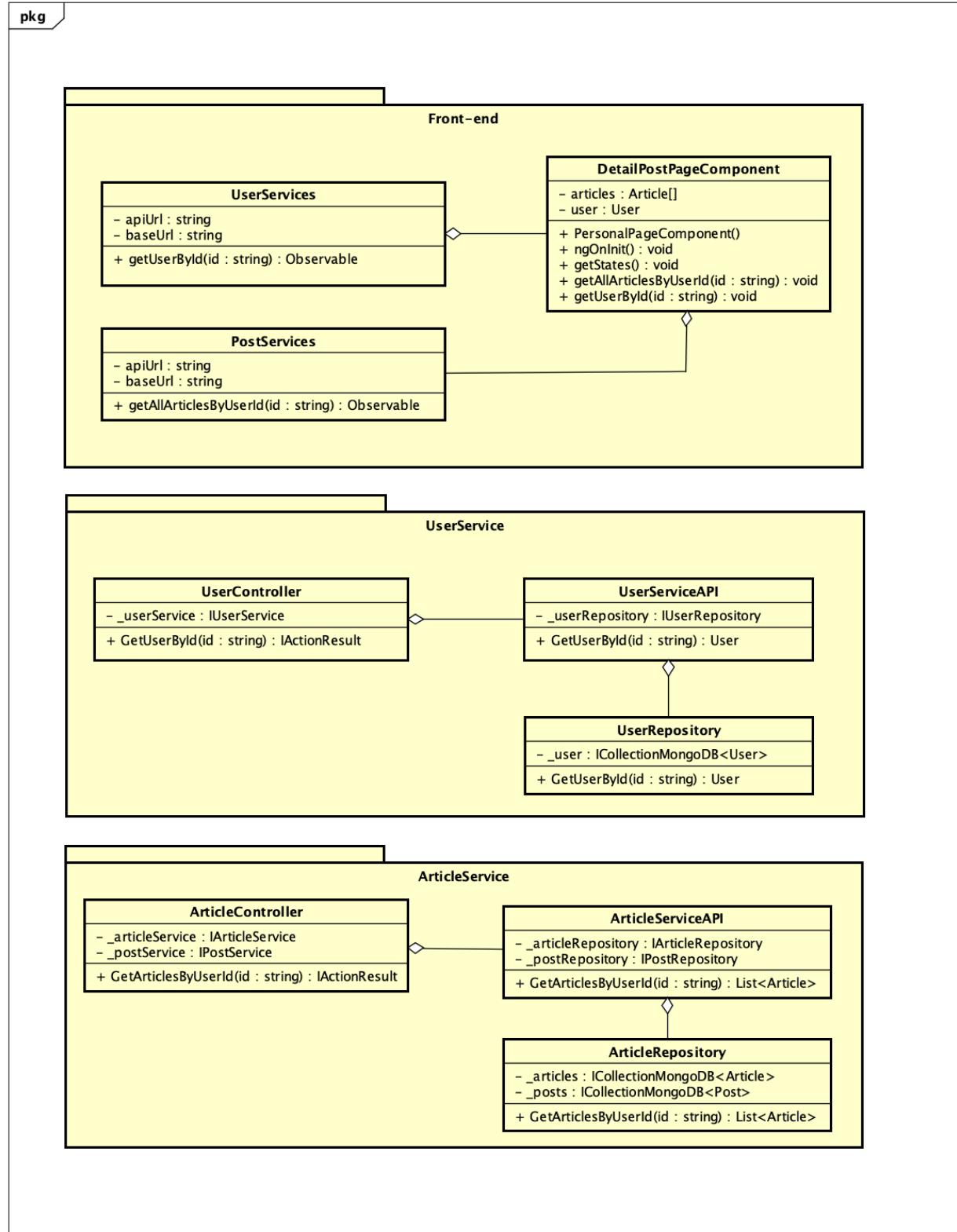


Figure 4-165: View User's created posts Class Diagram

Class Specification

PersonalPageComponent

PersonalPageComponent			
Physical address	src\app\pages\list-post-page\personal-page.component.ts		
Base Class	Class		
Attributes			
No	Name	Return Type	Description
1	articles	Article	
2	user	User	
Operations			
No	Name	Return Type	Description
1	PersonalPageComponent	void	Contractor
2	ngOnInit	void	Life circle hook in Angular
3	getUserById	void	Get user by id
4	getAllArticlesByUserId	void	Get all user's created article posts
5	getSate	void	Get state

PostServices

PostServices			
Physical address	src\app\core\services\post-service\post.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	apiUrl	string	
2	baseUrl	string	
Operations			
No	Name	Return Type	Description
1	getAllArticlesByUserId	Observable	Call api get data from server

ArticleController

ArticleController			
Physical address	Src\Services\PostService\PostService\Controllers\ArticleController.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	_articleService	IArticleService	
2	_postService	IPostService	
Operations			
No	Name	Return Type	Description

1	GetArticlesById	IActionResult	Get user's created article posts and return data to font-end.
---	-----------------	---------------	---

ArticleServiceAPI

ArticleService			
Physical address	Src\Services\PostService\PostService\Services\ArticleService.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	_articleRepository	IArticleRepository	
Operations			
No	Name	Return Type	Description
1	GetArticlesById	List<Object>	Get user's created posts

ArticleRepository

ArticleRepository			
Physical address	Src\Services\PostService\PostService\Repositories\ArticleRepository.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	_articles	ICollectionMongoDB<Article>	
2	_posts	ICollectionMongoDB<Post>	
Operations			
No	Name	Return Type	Description
1	GetArticlesById	List<Object>	Get user's created posts from database

UserServices

UserServices			
Physical address	src\app\core\services\post-service\user.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	apiUrl	String	
2	baseUrl	String	
Operations			
No	Name	Return Type	Description
1	getUserById	Observable	Call api get data from server

UserController

UserController			
Physical address	Src\Services\PostService\PostService\Controllers\UserController.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	_userService	IUserService	
Operations			
No	Name	Return Type	Description
1	GetUserById	IActionResult	Get user detail return data to front-end.

UserServiceAPI

UserServiceAPI			
Physical address	Src\Services\PostService\PostService\services\UserServiceAPI.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	_userRepository	IUserRepository	
Operations			
No	Name	Return Type	Description
1	GetUserById	Object	Get user detail

UserRepository

UserRepository			
Physical address	Src\Services\PostService\PostService\Repositories\UserRepository.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	_users	ICollectionMongoDB<User>	
Operations			
No	Name	Return Type	Description
1	GetUserById	Object	Get user detail from database

Sequence Diagram

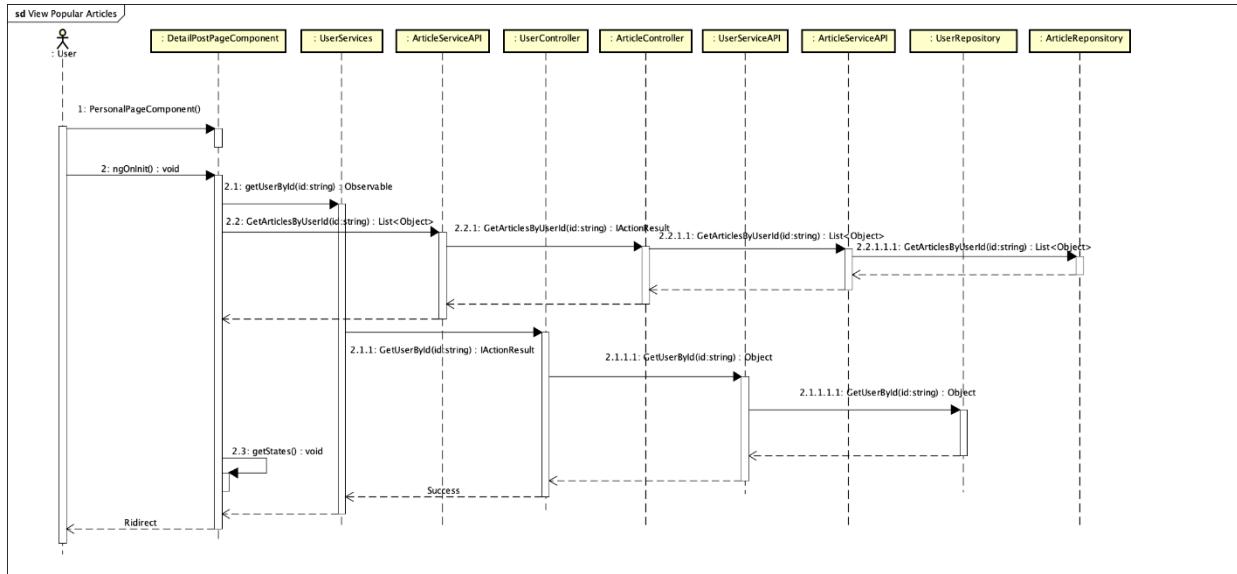


Figure 4-166: View user's created posts Sequence Diagram

4.3.4.53 View user's followings/followers

Screen design

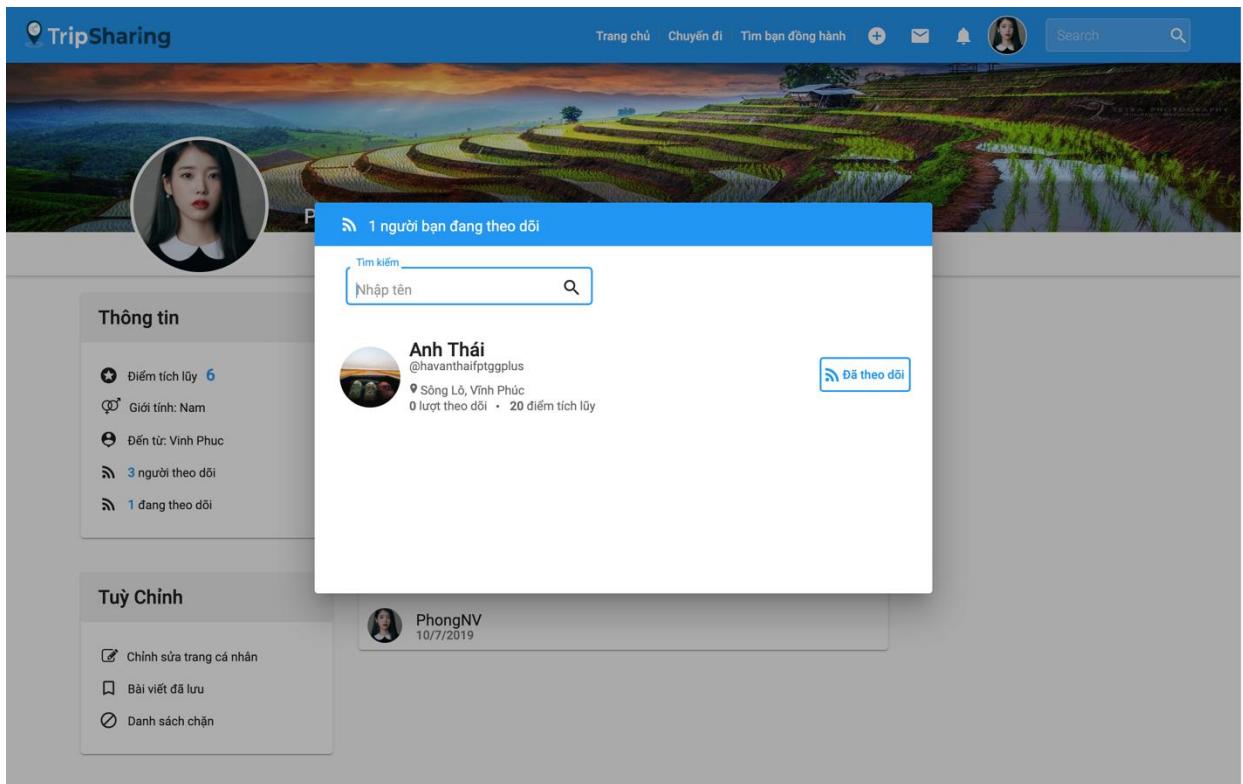


Figure 4-167: View user's followings/followers screen

Class Diagram

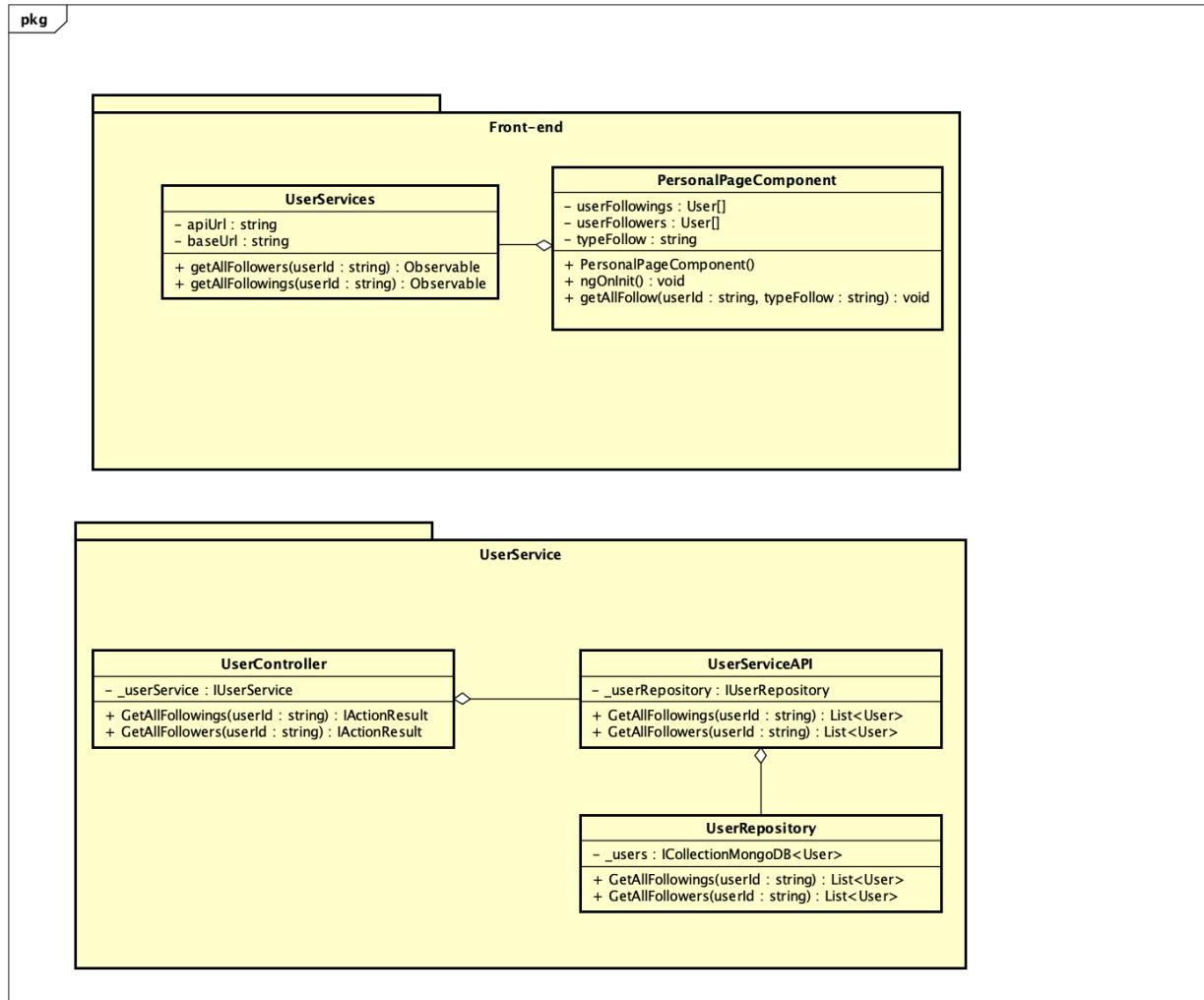


Figure 4-168: View user's following follower Class Diagram

Class Specification

PersonalPageComponent

PersonalPageComponent			
Physical address	src\app\pages\list-post-page\personal-page.component.ts		
Base Class	Class		
Attributes			
No	Name	Return Type	Description
1	userFollowings	User[]	
2	userFollowers	User[]	
Operations			
No	Name	Return Type	Description
1	PersonalPageComponent	void	Constructor
2	ngOnInit	void	Life circle hook in Angular

3	getAllFollowers	void	Get all user's followers
4	getAllFollowings	void	Get all user's followings

UserServices

UserServices			
Physical address	src\app\core\services\post-service\user.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	apiUrl	String	
2	baseUrl	String	
Operations			
No	Name	Return Type	Description
1	getAllFollowers	Observable	Call api get data from server
2	getAllFollowings	Observable	Call api get data from server

UserController

UserController			
Physical address	Src\Services\PostService\PostService\Controllers\UserController.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	_userService	IUserService	
Operations			
No	Name	Return Type	Description
1	GetAllFollowers	IActionResult	Get user's followers return data to font-end.
2	GetAllFollowings	IActionResult	Get user's following and return data to font-end.

UserServiceAPI

UserServiceAPI			
Physical address	Src\Services\PostService\PostService\services\UserServiceAPI.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	_userRepository	IUserRepository	
Operations			

No	Name	Return Type	Description
1	GetAllFollowers	List<Object>	Get user's followers
2	GetAllFollowings	List<Object>	Get user's followings

UserRepository

UserRepository			
Physical address	Src\Services\PostService\PostService\Repositories\UserRepository.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	_users	ICollectionMongoDB<User>	
Operations			
No	Name	Return Type	Description
1	GetAllFollowers	List<Object>	Get user's followers from database
2	GetAllFollowings	List<Object>	Get user's followings from database

Sequence Diagram

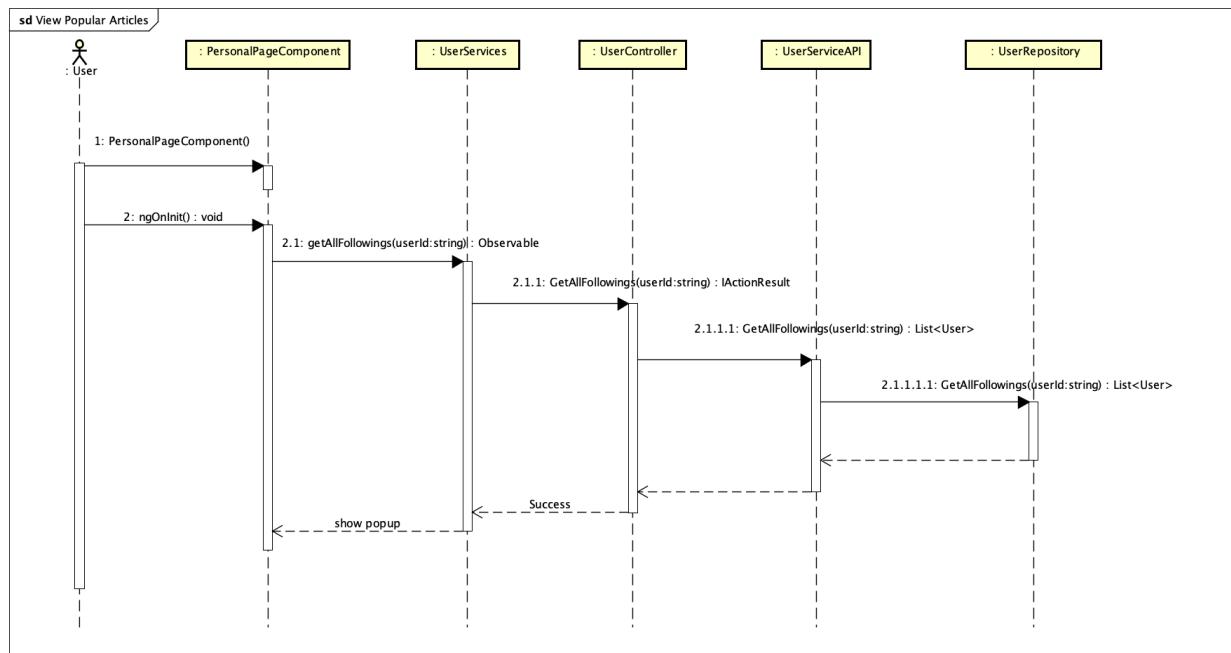


Figure 4-169: View user's following follower Sequence Diagram

4.3.4.54 View user's profile

Screen design

Trang chủ Chuyển đi Tim bạn đồng hành + ⚡ Search

PhongNV

Bài viết Chuyển đi Tim bạn đồng hành

Thông tin LỌC

Điểm tích lũy 6
Giới tính: Nam
Đến từ: Vinh Phuc
3 người theo dõi
1 đang theo dõi

Tuỳ chỉnh

Chỉnh sửa trang cá nhân
Bài viết đã lưu
Danh sách chặn

Giấc mơ Nhật Bản tuổi thơ của bạn, giờ đã khác như nào?
Thế hệ 8x, 9x Việt Nam từng sống trong những ước mơ tuổi thơ về đất nước Nhật Bản...

PhongNV 10/7/2019

Figure 4-170: View user's profile screen

Class Diagram

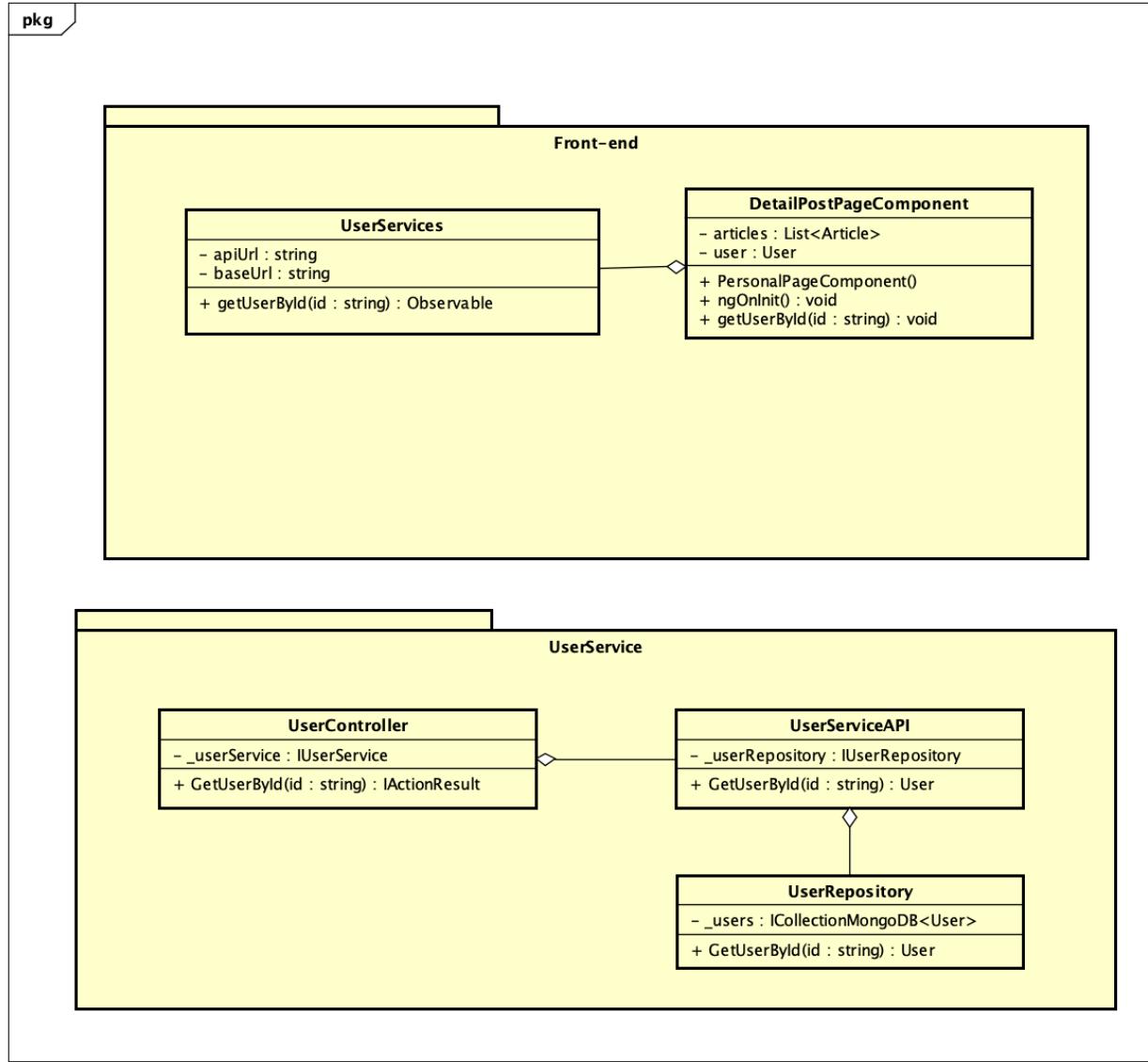


Figure 4-171: User's profile

Class Specification

PersonalPageComponent

PersonalPageComponent			
Physical address	src\app\pages\list-post-page\personal-page.component.ts		
Base Class	Class		
Attributes			
No	Name	Return Type	Description
1	articles	List<Article>	
2	user	List<User>	
Operations			
No	Name	Return Type	Description

1	PersonalPageComponent	void	Contructor
2	ngOnInit	void	Life circle hook in Angular
3	getUserById	void	Get user by id

UserServices

UserServices			
Physical address	src\app\core\services\post-service\user.service.ts		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	apiUrl	String	
2	baseUrl	String	
Operations			
No	Name	Return Type	Description
1	getUserById	Observable	Call api get data from server

UserController

UserController			
Physical address	Src\Services\PostService\PostService\Controllers\UserController.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	_userService	IUserService	
Operations			
No	Name	Return Type	Description
1	GetUserById	IActionResult	Get user detail return data to front-end.

UserServiceAPI

UserServiceAPI			
Physical address	Src\Services\PostService\PostService\services\UserServiceAPI.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	_userRepository	IUserRepository	
Operations			
No	Name	Return Type	Description

1	GetUserById	Object	Get user detail
---	-------------	--------	-----------------

UserRepository

UserRepository			
Physical address	Src\Services\PostService\PostService\Repositories\UserRepository.cs		
Base Class	Class		
Attributes			
No	Name	Type	Description
1	_users	ICollectionMongoDB<User>	
Operations			
No	Name	Return Type	Description
1	GetUserById	Object	Get user detail from database

Sequence Diagram

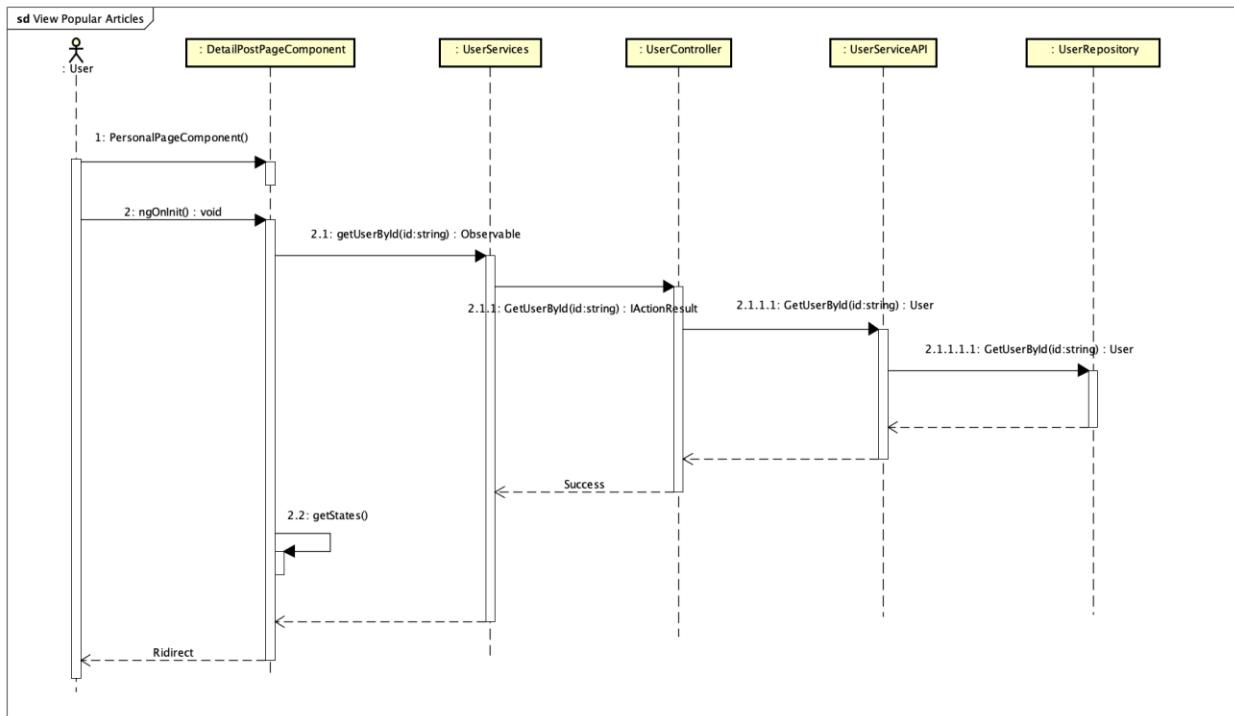


Figure 4-172: View user's profile

Chapter 5: Software Testing Documentation

5.1 Introduction

5.1.1 Purpose

This chapter's primary goal is to create testing plans and execute the defects detection and prevention procedures, which may cause software malfunctioning. Another objective of this chapter is to provide details about the software quality and to ensure that the end result meets the business and user requirements.

5.1.2 Scope of testing

➤ Stages of testing:

There are 4 phases in the Testing Process: Unit testing, Integration testing, System testing and Acceptance testing.

ID	Test Stages	Description
1	Unit testing	Unit Tests are performed by developers to ensure that individual units of source code function as intended. Unit tests are primarily used to test complicated algorithms and automatically test important functions.
2	Integration testing	Integration Tests are performed by testers, this is a software testing method in which individual software modules are combined and tested as a group. We also know defects between the modules. After that, developer will fix the system suitably.
3	System testing	System Tests are performed by testers. In this test, tester will test of a complete, fully integrated software product and determine whether or not the system meets the requirements.
4	Acceptance testing	Testers create check list include some rules between team members and customers. If customers don't have any requirements, we will build acceptance test base on the purpose of project

➤ Type of testing

The following types of testing are performed:

- Functional testing
- User interface testing

➤ Range of testing

Team performs all functions defined in the SRS based on the approved version.

5.2 Test plan

5.2.1 Testing tools and environment

5.2.1.1 Testing tools

- **Chrome Developer Tools:** To view logs, inspect elements.



Figure 5-1 Chrome Dev Tools

- **Trello and Backlog:** manage bugs, to log bugs, resolve bugs and close bug
- **Microsoft Excel:** To manage test cases



Figure 5-2 Microsoft Excel

- **Postman:** A tool to test API endpoints and returned results.



Figure 5-3 Postman

5.2.1.2 Testing environment

Type of testing	Software	Hardware
Unit test	Visual Studio	Personal computer for developing with the minimum configuration: - Windows 10 Pro 64-bit. - Intel® Core™ i5 5200 CPU. - Installed memory (RAM): 8.00GB - Visual Studio 2017 or 2019

Integration Test	Google chrome	Personal computer for developing with the minimum configuration: - Windows 10 Pro 64-bit. - Intel® Core™ i5 5200 CPU. - Installed memory (RAM): 8.00GB
System Test	Google chrome	Personal computer for developing with the minimum configuration: - Windows 10 Pro 64-bit. - Intel® Core™ i5 5200 CPU. - Installed memory (RAM): 8.00GB
Acceptance Test	Google chrome	Personal computer for developing with the minimum configuration: - Windows 10 Pro 64-bit. - Intel® Core™ i5 5200 CPU. - Installed memory (RAM): 8.00GB

5.2.2 Resources and responsibilities

This table shows the staffing assumptions for the project.

ID	Resources	Responsibilities
1	Project Manager	<ul style="list-style-type: none"> • Responsible for Project Schedules and overall success of the project. • Review Test Plan and Test report.
2	Test Leader	<ul style="list-style-type: none"> • Manage Test resource and assign test tasks • Create Test Plan • Review Test case • Create Test report
3	Tester	<ul style="list-style-type: none"> • Preforming the actual system testing. • Create Test Cases. • Execute Test. • Report test result. • Log bugs
4	Developer	<ul style="list-style-type: none"> • Create and execute unit tests • Fix bugs.

5.2.3 Test strategy

5.2.3.1 Test model

- TripSharing deploys a contemporary of traditional software development models is "V-Model":
 - At V-model, corresponding to a test phase is a software development phase, testing in the V-model is done in parallel with the software development cycle.
 - Each level of the development lifecycle is verified before moving on to the next level.
 - This helps in identifying errors very early in the lifecycle and minimizes potential future defects appearing in the code later in the lifecycle.

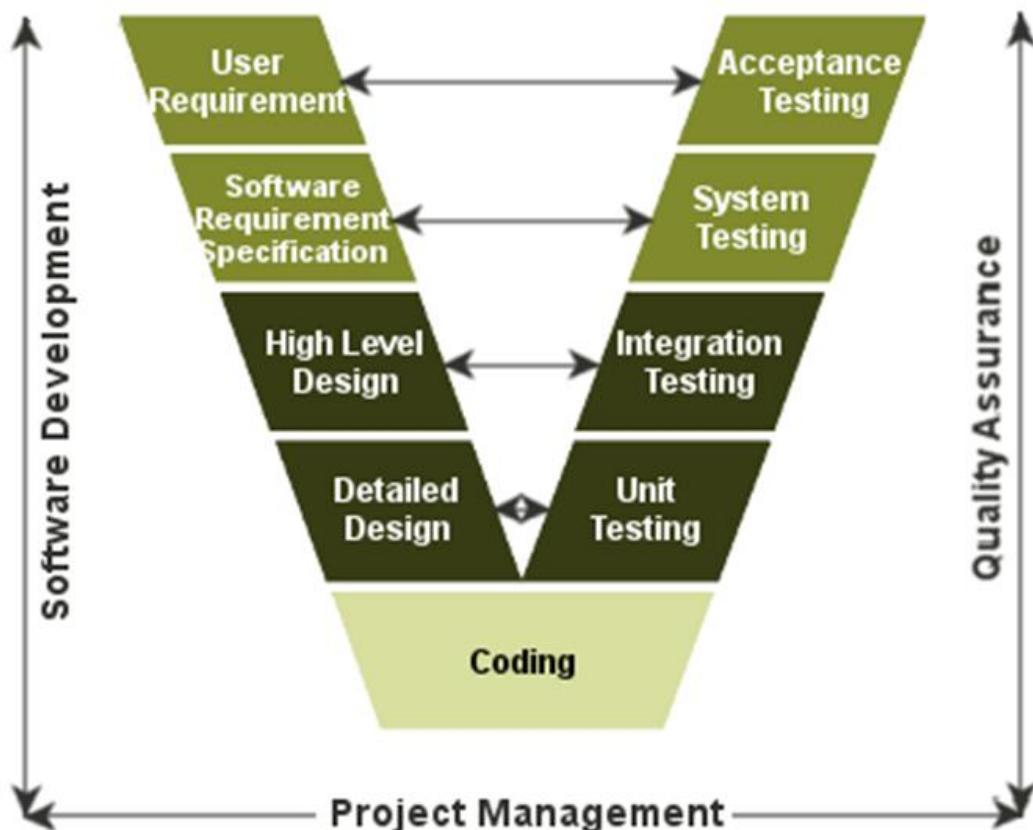


Figure 5-4 Architecture of V-Model

5.2.3.2 Test types

5.2.3.2.1 Functional testing

Functional testing is a formal type of testing performed by testers. Functional testing focuses on testing each function of the software application operates in conformance with the requirement specification.

- **Test Objective:**
 - Testing the main functions of an application
 - It involves basic usability testing of the system. It checks whether a user can freely navigate through the screens without any difficulties.

- Checks the accessibility of the system for the user
 - Usage of testing techniques to check for error conditions. It checks whether suitable error messages are displayed.
- **Technique**
 - Each and every functionality of the system is tested by providing appropriate input, verifying the output and comparing the actual results with the expected results.
- **Completion Criteria**
 - All planned tests have been executed.
 - All identified defects have been resolved and closed

5.2.3.2.2 User interface testing

User Interface testing is a testing technique used to identify the presence of defects in a product/software under test by using Graphical user interface [GUI].

- **Test Objective**
 - Ensure that the GUI provides the user with an appropriate access and navigation through the functions of the target-of-test.
 - GUI testing ensures that the objects within the GUI function as expected and conform to requirement.
- **Technique**
 - GUI test will be performed fully on all screens
 - Verification of the overall look and feel of the TripSharing system including initial position, text size, color, initial button, tab order, label, screen sizes and text font is readable.
- **Completion Criteria**
 - Verifying that the application responds correctly to events such as clicking on the button.
 - GUI testing would also confirm that appearance elements such as fonts and images conform to design specifications.

5.2.3.3 Test schedule

Table below is the Test Schedule for TripSharing Project

Test Schedule	Start Date	End Date
<i>Phase 1:</i>		
Unit Testing	06/06/2019	20/07/2019
Integration Testing	10/06/2019	30/06/2019
System Testing	15/06/2019	10/07/2019
<i>Phase 2:</i>	17/06/2019	20/07/2019
Unit Testing	20/07/2019	25/08/2019
Integration Testing	23/07/2019	10/08/2019
System Testing	26/07/2019	15/08/2019
Acceptance Testing	28/07/2019	16/08/2019
	10/08/2019	20/08/2019

Table 5-1 Test schedule

5.2.4 Features to be tested

All features are listed in the use case list.

5.2.5 Features not to be tested

All features are listed in 1.5.1.3 above will not to be tested.

5.3 Test Approach

5.3.1 Unit testing

- Unit testing is a software testing method by which small units of source code are tested to determine whether they meet the requirements.
- Unit testing will be done by the developers and approved by the development team leader. The TripSharing development team embrace these features to gain the following advantages:
 - Reduce the number of bugs in production code.
 - Saving development time.
 - Ensure the functions of the software are in accordance with customer requirements
 - Identify errors very early in the development process and minimize potential future errors that may appear

5.3.1.1 Unit testing framework

- We use NUnit is a unit testing framework for .NET, it is the most used framework for writing unit test cases and JetBrains dotCover is a .NET unit test runner and code coverage tool.
- Mock function are simulated functions that mimic the behavior of real functions in controlled ways.
- Unit testing scripts are created manually and saved to ProjectName.Tests directory of TripSharing API services.

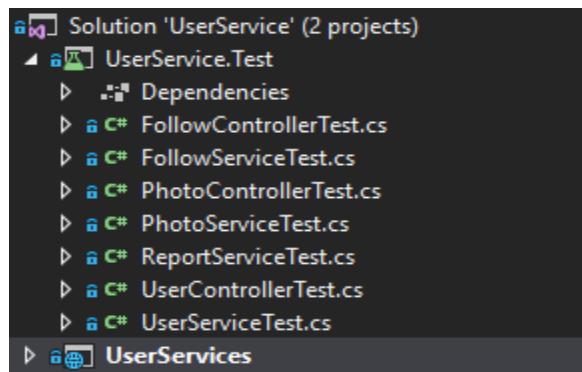


Figure 5-5 Test directory structure

- Unit tests focus on individual functions in a class and are created as in the picture.

```

[TestCase]
0 references | 0 exceptions
public void TestGetRecommendArticlesReturnOkObjectResult()
{
    // mock ClaimsIdentity Object
    ClaimsIdentity claims = new ClaimsIdentity(new Claim[]
    {
        new Claim(ClaimTypes.Name, "PhongTvs"),
        new Claim(ClaimTypes.Role, "member"),
        new Claim("user_id", "5d247a04eff1030d7c5209a1")
    });
    var contextMock = new Mock<HttpContext>();
    contextMock.Setup(x => x.User).Returns(new ClaimsPrincipal(claims));
    List<Article> articles = new List<Article>();
    articles.Add(article);
    I Enumerable<Article> _iEnumerableArticle = articles;
    // mock data of GetRecommendArticles() function return _iEnumerableArticle
    mockArticleService.Setup(x => x.GetRecommendArticles(It.IsAny<PostFilter>(),
        It.IsAny<UserInfo>(), It.IsAny<int>())).Returns(_iEnumerableArticle);
    var _articleController = new ArticleController(mockArticleService.Object, mockPostService.Object);
    _articleController.ControllerContext.HttpContext = contextMock.Object;
    // get actual value of GetRecommendArticles function from ArticleController
    IActionResult getRecommendArticles = _articleController.GetRecommendArticles(postFilter, 4);
    var type = getRecommendArticles.GetType();
    Assert.AreEqual(type.Name, "OkObjectResult");
}

```

Figure 5-6: Unit test case sample

This factory function used to get recommend articles

- Coverage report:

Symbol	Coverage (%)	Uncovered/Total Stmts.
▲ Total	96%	19/423
▲ ChatService	89%	19/167
▷ ChatService.Models	60%	16/40
▷ ChatService.Controllers	96%	3/82
▷ ChatService.Services	100%	0/45
▲ ChatService.Test	100%	0/256
▲ ChatService.Test	100%	0/256
▷ ChatServiceTest	100%	0/99
▷ ChatControllerTest	100%	0/157

Figure 5-7: Unit test chat service coverage

Symbol	Coverage (%)	Uncovered/TotalStmts.
▲ Total	87%	74/566
└ IdentityProvider		
└ IdentityProvider.Services	75%	74/296
└ IdentityProvider.Models	68%	55/172
└ IdentityProvider.Controllers	69%	16/52
└ IdentityProvider.Test	96%	3/72
└ IdentityProvider.Test	100%	0/270
└ AccountServiceTest	100%	0/270
└ AccountControllerTest	100%	0/123

Figure 5-8 Unit test identity provider service coverage

Symbol	Coverage (%)	Uncovered/TotalStmts.
▲ Total	95%	6/111
└ NotificationService		
└ NotificationService.Models	86%	6/44
└ NotificationService.Services	57%	6/14
└ NotificationService.Controllers	100%	0/11
└ NotifyService.Test	100%	0/19
└ NotifyService.Test	100%	0/67
└ NotificationServiceTest	100%	0/67
└ NotificationControllerTest	100%	0/29

Figure 5-9 Unit test notification service coverage

Symbol	Coverage (%)	Uncovered/Total Stmt.
▲ Total	91%	220/2467
└ PostService	80%	220/1078
└ PostService.Models	56%	100/226
└ PostService.Services	80%	77/391
└ PostService.Controllers	91%	43/461
└ PostService.Test	100%	0/1389
└ UserService.Test	100%	0/129
└ BookmarkServiceTest	100%	0/48
└ BookmarkControllerTest	100%	0/81
└ PostService.Test	100%	0/1260
└ AuthorServiceTest	100%	0/17
└ UploadFileControllerTest	100%	0/20
└ LikeControllerTest	100%	0/28
└ LikeServiceTest	100%	0/38
└ TopicServiceTest	100%	0/39
└ ReportServiceTest	100%	0/42
└ CommentServiceTest	100%	0/52
└ PostServiceTest	100%	0/56

Figure 5-10 Unit test post service coverage

Symbol	Coverage (%)	Uncovered/Total Stmt.
▲ Total	92%	69/896
└ UserServices	81%	69/371
└ UserServices.Models	60%	31/78
└ UserServices.Services	78%	28/130
└ UserServices.Controllers	94%	10/163
└ UserService.Test	100%	0/525
└ UserService.Test	100%	0/525
└ PhotoServiceTest	100%	0/19
└ PhotoControllerTest	100%	0/39
└ ReportServiceTest	100%	0/43
└ FollowServiceTest	100%	0/62
└ UserServiceTest	100%	0/74
└ FollowControllerTest	100%	0/106
└ UserControllerTest	100%	0/182

Figure 5-11 Unit test user service coverage

5.3.2 Integration testing and System test

- Integration testing is a software testing method in which individual software modules are combined and tested as a group.
- System testing is testing conducted on a complete integrated system to evaluate the system's compliance with its specified requirements.
- Detailed test cases will be described in TripSharing_TestCase_Final folder. As a standard definition, TripSharing defines that a test case is:
 - A set of test data and their expected results. A test case validates one or more system requirements and generates a result is pass or fail, from there can verify if the system works as expected or not.
 - A good test case should follow two basic aspects, the Contents and the Style. Test cases should be developed for each use case scenario
 - TripSharing Project System testing will not focus on common logic of system like length of text but focus on behavior of website and aims to validate that all software module dependencies are functionally correct, and the communications is maintained between separate modules.

5.3.4 Acceptance Test

Acceptance testing is a level of the software testing process where a system is tested for acceptability. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery. But our project will use the checklist as a substitute for acceptance testing.

The content of the checklist is shown in the table below:

ID	Checklists		
		Yes	No
Functional			
CL-001	User can sign up a new account	✓	
CL-002	User can sign in to TripSharing.net by Facebook account	✓	
CL-003	User can sign in to TripSharing.net by Google account	✓	
CL-004	User can sign in to TripSharing.net by normal account	✓	
CL-005	User can change the new password for account	✓	
CL-006	User can reset the password	✓	
CL-007	User can update profile and update information	✓	
CL-008	User can choose more interested topics	✓	
CL-009	User can create a new article	✓	
CL-010	User can create a new virtual trip	✓	
CL-011	User can create a new finding companion post	✓	
CL-012	Users are allowed to delete their own posts	✓	
CL-013	Users are allowed to edit their own posts	✓	
CL-014	User can join to a companion group that other member created	✓	
CL-015	User can leave to a companion group that other member created	✓	
CL-016	User can share one or more posts from TripSharing.net to Facebook application	✓	

CL-017	If an article violates the website's rules, user can report it to the administrator.	✓	
CL-018	If a comment violates the website's rules, users can report it to the administrator.	✓	
CL-019	If a member violates the website's rules, users can report them to the administrator.	✓	
CL-020	User can bookmark one or more posts and add them to the bookmarks list.	✓	
CL-021	User can view bookmarks list	✓	
CL-022	User can remove one or more posts from bookmarks list.	✓	
CL-023	User can add a new comment below the content of any post	✓	
CL-024	User can edit their own comments	✓	
CL-025	User can delete their own comments	✓	
CL-026	User can like any comment	✓	
CL-027	User can unlike any comment	✓	
CL-028	User can follow other member	✓	
CL-029	User can unfollow other member	✓	
CL-030	User can like one or more posts	✓	
CL-031	User can unlike one or more posts	✓	
CL-032	User can create a group chat	✓	
CL-033	User can remove a member from group chat	✓	
CL-034	User can send or receive messages and system will notify for user when they have new messages	✓	
CL-035	Administrator can add a new travel topic to topics list	✓	
CL-036	Administrator can remove one or more travel topic to topics list	✓	
CL-037	Administrator can edit a travel topic	✓	
CL-038	User can search 3 posts type: article, virtual trip and finding companion	✓	
CL-039	User can search a member from members list of system	✓	
CL-040	User can view recommended articles on home page	✓	
CL-041	User can view popular articles on home page	✓	
CL-042	User can view recently articles on home page	✓	
CL-043	User can view a post detail with category: article, virtual trip and finding companion	✓	
CL-044	User can view a other member's profile	✓	
CL-045	User can view a other member's followings/followers number	✓	
CL-046	User can view a other member's created posts	✓	
CL-047	Administrator can view all user's account	✓	
CL-048	Administrator can view all accounts that users reported	✓	
CL-049	Administrator can ban an account	✓	
CL-050	Administrator can unban an account		
CL-051	Administrator can view all posts reported by users	✓	
CL-052	Administrator can remove a post reported by users	✓	
CL-053	Administrator can restore a post that previously removed	✓	

CL-054	Administrator can view all comments that is reported by other members	✓	
CL-055	Administrator can remove a comment that is reported by other members	✓	
CL-056	Administrator can restore a comment that previously removed	✓	
CL-057	Administrator can view statistic of users	✓	
CL-058	Administrator can view statistic of posts	✓	
CL-059	Check application logout functionality.	✓	
GUI and Usability			
CL-060	All mandatory fields are validated.	✓	
CL-061	Validation error messages are displayed properly below the field.	✓	
CL-062	All error messages are displayed in red color.	✓	
CL-063	Delete functionality for any record on page are asked for confirmation.	✓	
CL-064	All like, comment number values are formatted properly.	✓	
CL-065	Application crash or unavailable pages are redirected to error page.	✓	
CL-066	The screen is well organized and easy to interact with.	✓	
CL-067	All fields on page (e.g. text box, radio options, dropdown lists) should be aligned properly.	✓	
CL-068	System display notification message when meet trouble, error.	✓	
CL-069	All images displayed on the website must be of good quality	✓	
Database			
CL-070	All data must save in database when submit data successful.	✓	
CL-071	The id field in the data base is required, cannot be left blank.	✓	
CL-072	Field length shown to user on page and in database schema should be same.	✓	
CL-073	Database fields are designed with correct data type and data length.	✓	
CL-074	The relationship between the tables must be clear and true to the customer's requirements	✓	

5.3.5 Defect Log

- Trip Sharing project used <http://www.trello.com> in phase 1 and <http://www.backlog.com> in phase 2 to manager tasks and defects.
- Every member of TripSharing project creates an account backlog and Trello to take part in activities: control bugs, fix bugs, re-test bugs and close bug. Bug will be log by tester or developer in develop progress.

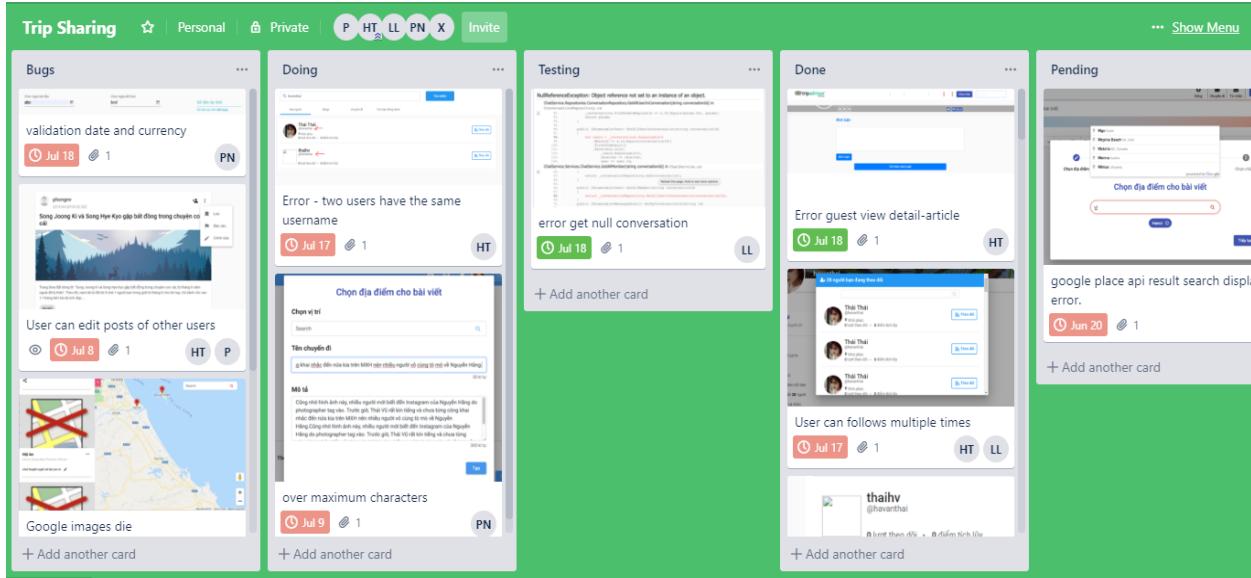


Figure 5-12 Control task and bug with Trello

Issue Type	Key	Subject	Assignee	Status	Priority	Created	Due date	Updated	Registered by	Attachment
Bug	TRIPSHARING-33	Login	Nguyen Van Phong	Open	↑	Aug. 13, 2019	Aug. 15, 2019	Aug. 13, 2019	Le Truong	
Bug	TRIPSHARING-34	Facebook Login	Ha Van Thai	Open	↑	Aug. 13, 2019	Aug. 15, 2019	Aug. 13, 2019	Le Truong	
Bug	TRIPSHARING-31	follow	Ha Van Thai	Open	→	Jul. 27, 2019	Jul. 30, 2019	Jul. 30, 2019	Nguyen Van Phong	
Bug	TRIPSHARING-28	Link on profile	Tran Van Phong	Resolved	↓	Jul. 26, 2019	Jul. 28, 2019	Jul. 28, 2019	Le Truong	
Bug	TRIPSHARING-27	Block	Nguyen Van Phong	Resolved	→	Jul. 26, 2019	Jul. 28, 2019	Jul. 28, 2019	Le Truong	
Bug	TRIPSHARING-24	Can not click on bookmark	Lý Phúc Linh	In Progress	→	Jul. 26, 2019	Jul. 28, 2019	Jul. 27, 2019	Le Truong	
Bug	TRIPSHARING-25	Loading trip	Nguyen Van Phong	In Progress	→	Jul. 26, 2019	Jul. 28, 2019	Jul. 27, 2019	Le Truong	
Bug	TRIPSHARING-18	Contribution point	Ha Van Thai	In Progress	→	Jul. 25, 2019	Jul. 27, 2019	Jul. 26, 2019	Le Truong	
Bug	TRIPSHARING-19	Virtual Trip without a cover is broken	Tran Van Phong	Resolved	→	Jul. 25, 2019	Jul. 27, 2019	Jul. 26, 2019	Le Truong	
Bug	TRIPSHARING-7	Guest can't see joined people in finding-companion-post	Nguyen Van Phong	Resolved	→	Jul. 23, 2019	Jul. 25, 2019	Jul. 25, 2019	Ha Van Thai	
Bug	TRIPSHARING-5	Login	Tran Van Phong	Resolved	→	Jul. 23, 2019	Jul. 25, 2019	Jul. 25, 2019	Le Truong	
Bug	TRIPSHARING-14	Article post	Tran Van Phong	Open	↑	Jul. 24, 2019	Jul. 26, 2019	Jul. 24, 2019	Le Truong	
Bug	TRIPSHARING-12	Login	Lý Phúc Linh	Resolved	↑	Jul. 24, 2019	Jul. 26, 2019	Jul. 24, 2019	Le Truong	

Figure 5-13 Control task and bug with Backlog

5.4 Test Report

5.4.1 Unit test report

The contents of the Unit Test Case Report are shown in the table below:

Service Name	Controller/Service Name	Pass	Fail	Not available	Number of Test Case
Chat Service	Chat Controller	15	0	0	15
	Chat Service	10	0	0	10
	Total	25	0	0	25
Identity Provider	Account Controller	16	0	0	16
	Account Service	15	0	0	15
	Total	31	0	0	31

Notification Service	Notification Controller	2	0	0	2
	Notification Service	2	0	0	2
	Total	4	0	0	4
Post Service	Article Controller	11	0	0	11
	Article Service	8	0	0	8
	Author Service	2	0	0	2
	Bookmark Controller	6	0	0	6
	Bookmark Service	5	0	0	5
	Comment Controller	5	0	0	5
	Commnet Service	5	0	0	5
	Companion Controller	13	0	0	13
	Companion Post Service	15	0	0	15
	Like Controller	2	0	0	2
	Like Service	4	0	0	4
	Post Controller	6	0	0	6
	Post Service	6	0	0	6
	Report Controller	6	0	0	6
	Report Service	5	0	0	5
	Topic Controller	8	0	0	8
	Topic Service	5	0	0	5
	Upload File Controller	2	0	0	2
	Upload File Service	1	0	0	1
UserService	Virtual Trip Controller	7	0	0	7
	Virtual Trip Service	7	0	0	7
	Total	133	0	0	133
	Follow Controller	10	0	0	10
	Follow Service	7	0	0	7
	Photo Controller	3	0	0	3
	Photo Service	2	0	0	2
	Report Service	5	0	0	5
Email Service	User Controller	19	0	0	19
	User Service	10	0	0	10
	Total	56	0	0	56
	Email Service	0	0	0	0
	Total	0	0	0	0
Total of Test Case		249	0	0	249

Table 5-2: Unit test case report

Service Name	Number of test cases				Final	
	Phase 1		Phase 2			
	Pass	Fail	Pass	Fail		
Chat Service	25	0	25	0	25	
Identity Provider	31	0	31	0	31	
Notify Service	4	0	4	0	4	

Post Service	113	0	133	0	133
User Service	42	0	56	0	56
Email Service	0	0	0	0	0
Total of test cases	215	0	249	0	249

Table 5-3: Unit test report

Service Name	Coverage	
	Phase 1	Phase 2
Chat Service	89%	89%
Identity Provider	75%	75%
Notify Service	86%	86%
Post Service	77%	80%
User Service	78%	81%
Email Service	0%	0%
Overall	76%	78%

Table 5-4: Unit test coverage report

5.4.2 Integration test report

Module code	Pass	Fail	Not available	Number of test case
Sign Up	17	0	0	17
Sign In	14	0	0	14
Forgot password	6	0	0	6
Sign Out	2	0	0	2
Change Password	16	0	0	16
Update Profile	14	0	0	14
Choose Interested Topic	14	0	0	14
View user's profile	2	0	0	2
Create a post	60	0	0	60
Edit a post	59	0	0	59
Like Unlike a post	7	0	0	7
Delete a post	11	0	0	11
Comment to a post	6	0	0	6
Edit a comment	5	0	0	5
Delete a comment	5	0	0	5
Like Unlike a comment	8	0	0	8
View a post detail	14	0	0	14
Bookmark a post	6	0	0	6
View all bookmarks	2	0	0	2
Remove a bookmark	7	0	0	7

Follow Unfollow a user	13	0	0	13
ViewPost	6	0	0	6
View user's follower following	3	0	0	3
View user's created posts	6	0	0	6
TravelTopic	14	0	0	14
Search	11	0	0	11
Join Leave to a companion group	5	0	0	5
Share a post	4	0	0	4
Send receive messages	9	0	0	9
View all users' account	3	0	0	3
View Statistic	5	0	0	5
View Reported	3	0	0	3
Reported	17	0	0	17
Ban a user	4	0	0	4
Unban a user	4	0	0	4
Remove a reported post	4	0	0	4
Restore a removed post	4	0	0	4
Remove a reported comment	4	0	0	4
Restore a removed comment	4	0	0	4
Total of Test Case	398	0	0	398

Table 5-6 Integration test case report

Integration test	Phase 1		Phase 2		Final
	Pass	Fail	Pass	Fail	
Total of test case	346	0	398	0	398

Table 5-7 Integration test report

5.4.3 System test report

The contents of the System Test Case Report are shown in the table below:

Module code	Pass	Fail	Not available	Number of test case
Guest	25	0	0	25
Member	95	0	0	95
User	30	0	0	30

Admin	35	0	0	35
Total of Test Case	185	0	0	185

Table 5-8 System test case report

We execute test with 2 stages with 2 phases of process model, to finish project.

The contents of the Test Report are shown in the table below:

Service Name	Number of test cases				Final	
	Phase 1		Phase 2			
	Pass	Fail	Pass	Fail		
Guest	25	0	25	0	25	
Member	89	0	95	0	95	
User	30	0	30	0	30	
Admin	21	0	35	0	35	
Total of test cases	165	0	185	0	185	

Table 5-9 System test report

Chapter 6: User manual

6.1 Deployment guidelines

6.1.1 Environment for development

- ❖ Operating system: Window 10 Pro
- ❖ Code IDE: Visual Studio 2017
- ❖ Code Editor: Visual Studio Code
- ❖ Node.js: 10.15.3
- ❖ Docker: 18.09.2
- ❖ Plugins:
 - TSLint
 - Dotcover

6.1.1.1 Setup development environment

6.1.1.1.1 Install Visual Studio 2017

- Go to <https://visualstudio.microsoft.com/downloads/>

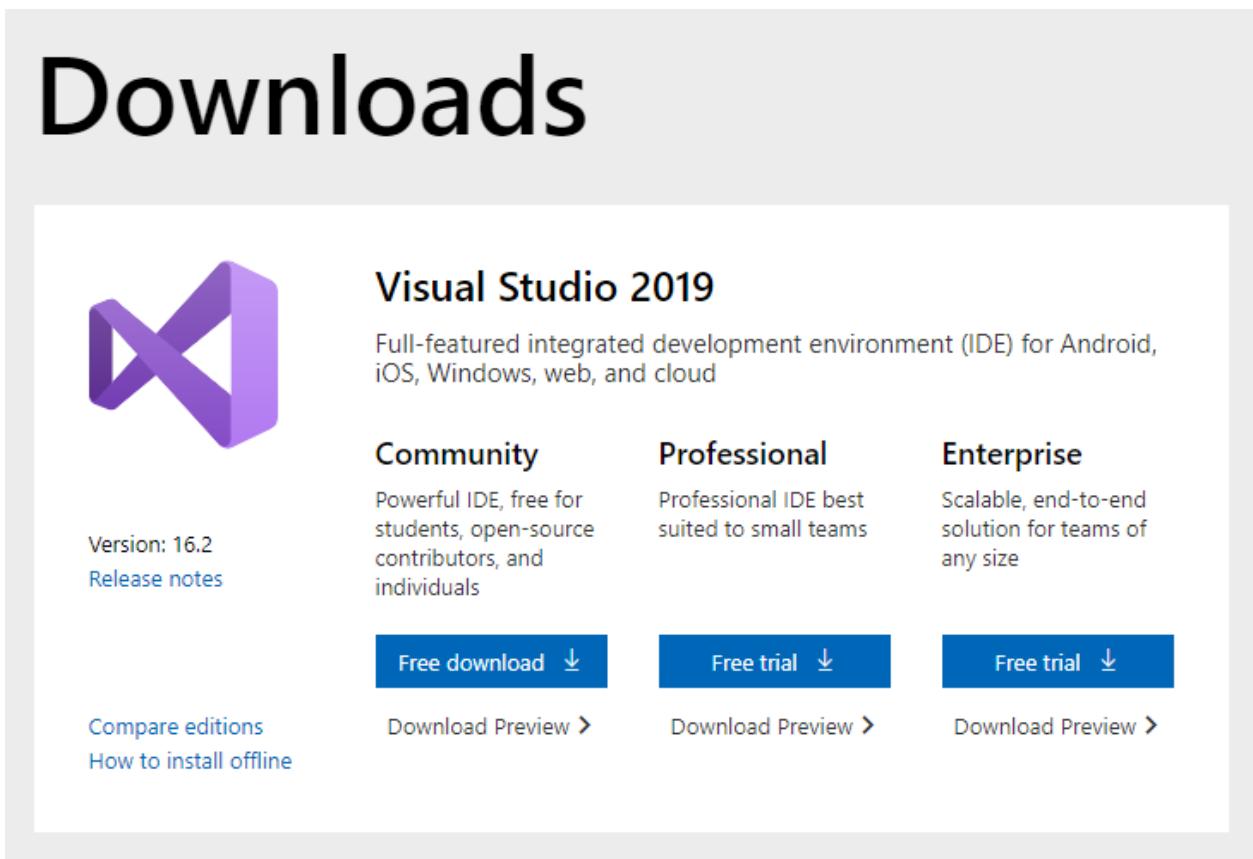


Figure 6-1: Visual Studio download page

- Currently, If the last version is not the Visual Studio 2017, scroll to the bottom of the page then click “Older versions” button. Otherwise, Click “Free download” button
- In older version page, Choose the right version (2017) and click “Download” button.

Expand All Collapse All

› 2017

Visual Studio 2017 and other Products

To download any product from the following list, click the download button and log in with your Visual Studio Subscription account when prompted. If you don't have a Visual Studio Subscription, you can create one for free by clicking on "Create a new Microsoft account" on the login page.

Visual Studio Community 2017; Visual Studio Professional 2017; Visual Studio Enterprise 2017;
Visual Studio 2017 for Mac
Visual Studio Test Professional 2017
Visual Studio Team Explorer 2017
Agents for Visual Studio 2017
Feedback Client for Visual Studio 2017
IntelliTrace Standalone Collector for Visual Studio 2017
Performance Tools for Visual Studio 2017
Remote Tools for Visual Studio 2017

[Download](#)

› 2015

› 2013

› 2012

› 2010

Figure 6-2: Visual Studio download old version page

- Run the downloaded executable file, follow the instructions.
- Start the application by opening the shortcut on desktop or from start menu.

6.1.1.1.2 Install Visual Studio Code

- Go to <https://code.visualstudio.com/> and click on “Download” button.

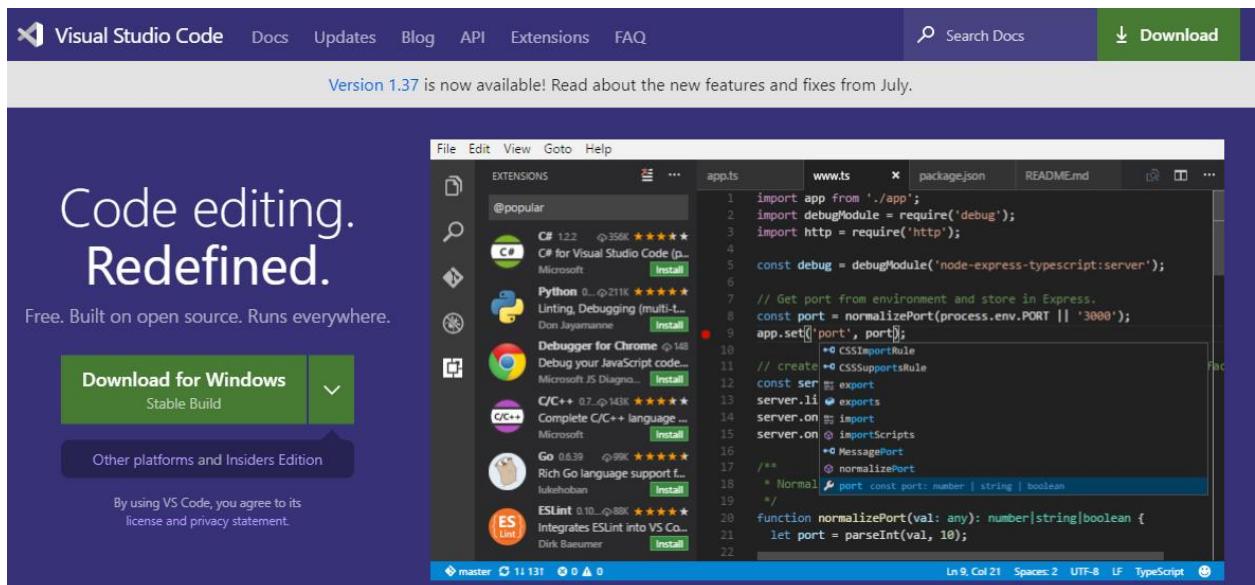


Figure 6-3: Visual Studio Code download page

- Run the downloaded executable file, follow the instructions.
- Start the application by opening the shortcut on desktop or from start menu.

6.1.1.3 Install Node.js

- Go to <https://nodejs.org/en/download/> and click “Windows Installer” to download the installation file.



Downloads

Latest LTS Version: **10.16.3** (includes npm 6.9.0)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.



Figure 6-4: Node.js download page

- Run the downloaded executable file, follow the instructions.
- To test if Node.js is installed successfully, open Command Prompt then enter “node -v”.

```
Microsoft Windows [Version 10.0.17763.678]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Thai>node -v
v10.15.3

C:\Users\Thai>
```

Figure 6-5: Show node's version

6.1.1.4 Install Docker

- Go to <https://hub.docker.com/editions/community/docker-ce-desktop-windows> and click “Get Docker” button.

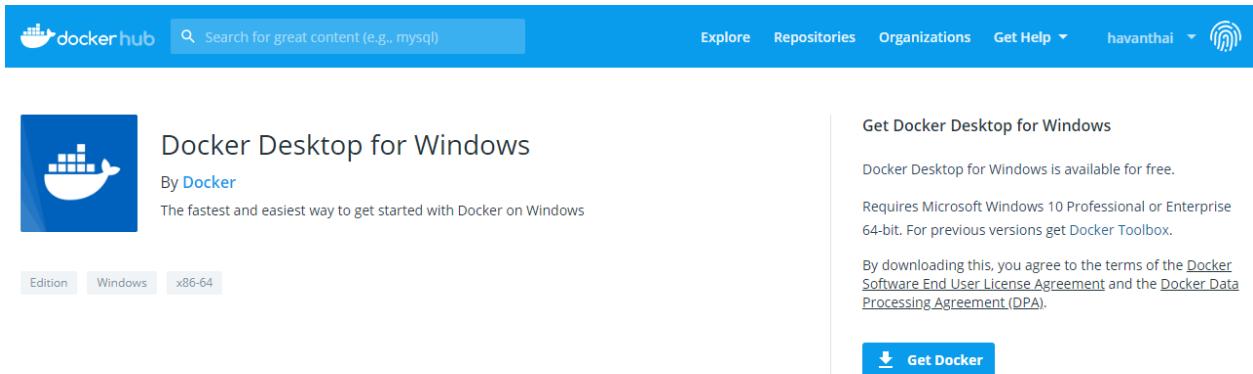


Figure 6-6: Docker download page

- Run the downloaded executable file, follow the instructions. Be sure to keep using Linux container when prompted to switch to Windows container.
- Start the application by open the short cut “Docker Desktop” on the desktop.

6.1.1.2 Run unit test and coverage

- ❖ Run unit test:
 - Open the solution which is needed to run test with visual studio.
 - On the menu bar, click **Test > Run > All Tests**

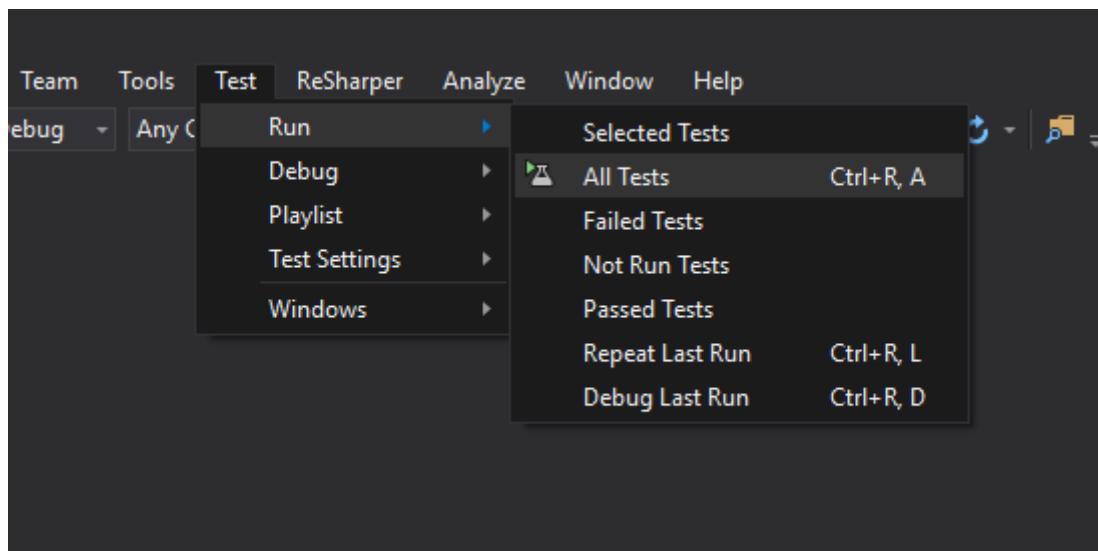


Figure 6-7: Run unit test

- Unit test result will be displayed in Test Explorer.

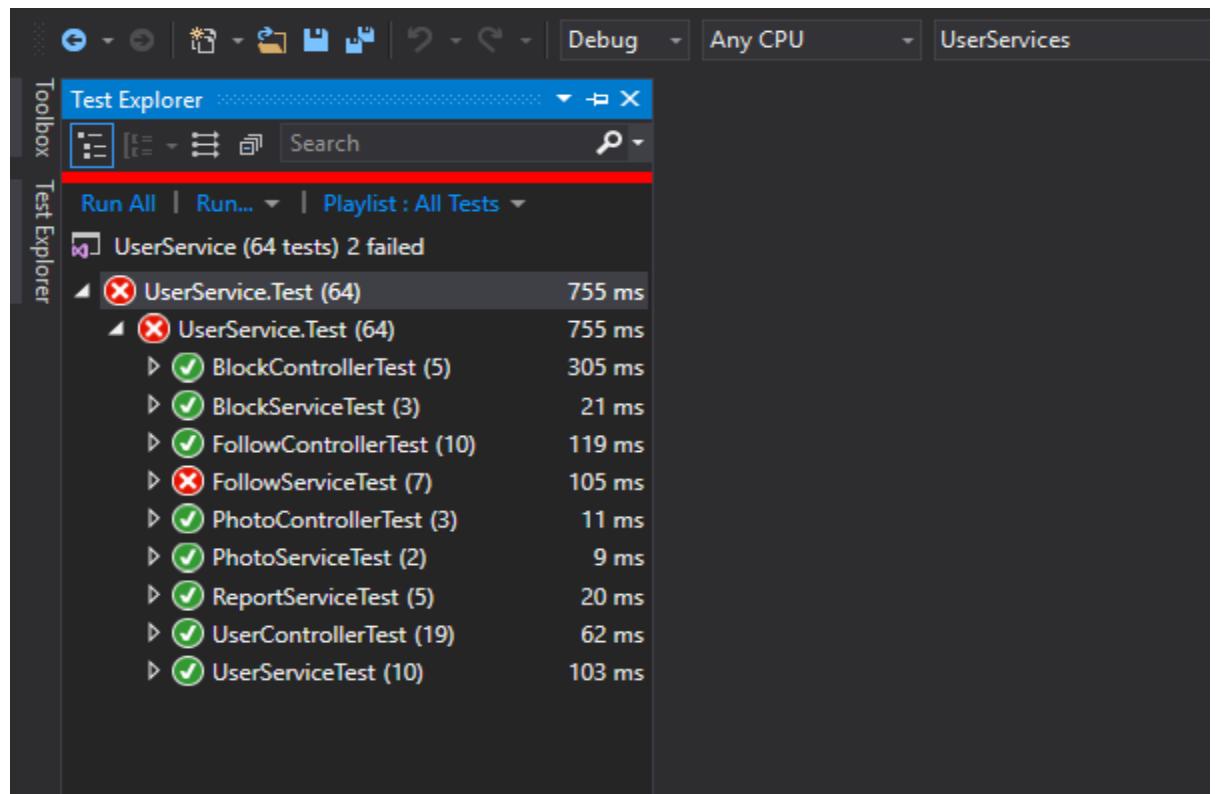


Figure 6-8: Unit test result

- ❖ Run coverage:
 - In Solution Explorer, right-click on solution name, then select **Cover Unit Tests**

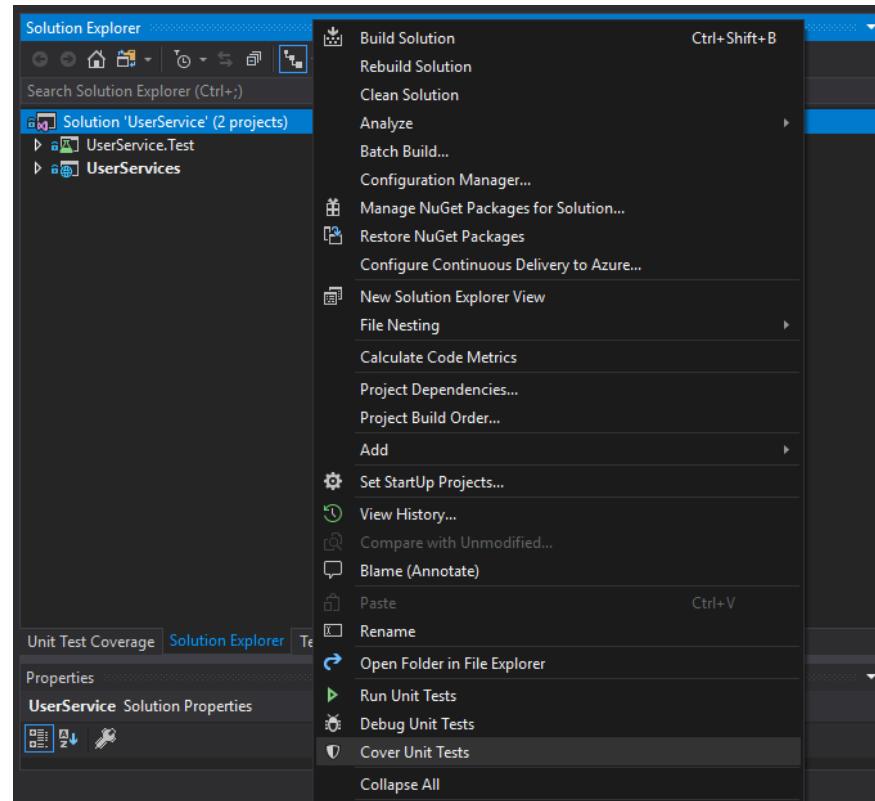


Figure 6-9: Run test coverage

- Wait for the process finish, test coverage result will be displayed in **Unit Test Coverage** tab.

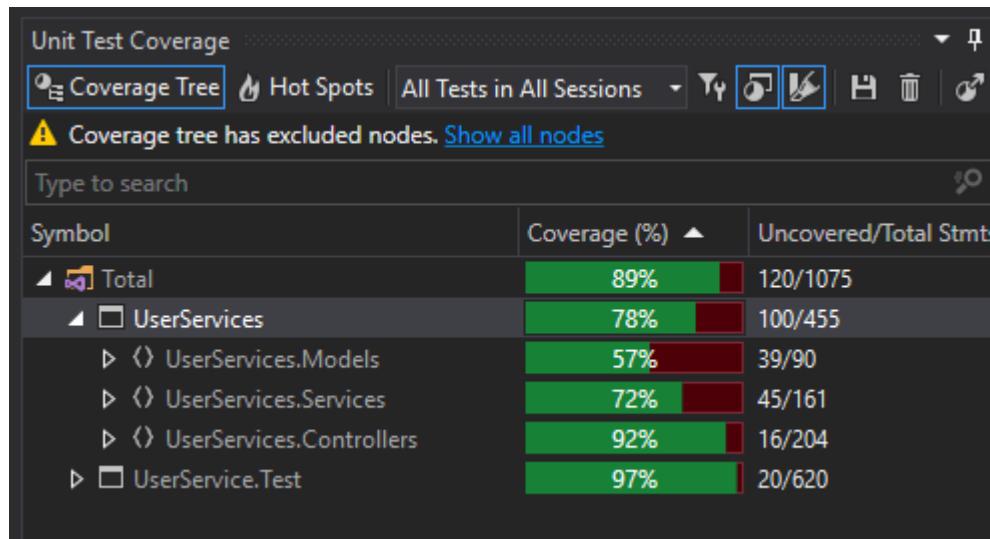


Figure 6-10: Test coverage result

6.1.1.3 Local network setup

- Run “Notepad” as Administrator.
- From File menu, click “Open”.
- Navigate to <Windows Drive>\Windows\system32\drivers\etc\.
- Choose “All Files (*.*)” from the dropdown to the right of the “File name” box.
- Choose “hosts” file and click “Open”.
- Add the following new lines at the end of the file:

```
127.0.0.1      dev-trip-sharing.net
```

6.1.2 Environment for deployment

Platform: Kubernetes Engine on Google Cloud Platform

6.1.2.1 Install Google Cloud SDK

- Download the Google Cloud SDK installer:
<https://dl.google.com/dl/cloudsdk/channels/rapid/GoogleCloudSDKInstaller.exe>
- Launch the installer and follow the prompts.
- After installation has completed, the installer presents several options:

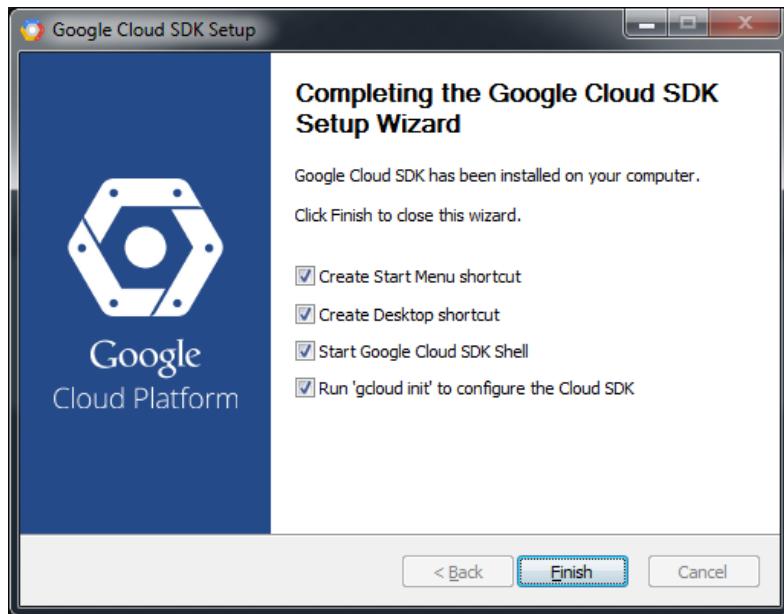


Figure 6-11: Google cloud SDK's installation completed window

- Make sure that the following are selected:
 - Start Google Cloud SDK Shell
 - Run 'gcloud init'
- The installer then starts a terminal window and runs the gcloud init command.

Reference: <https://cloud.google.com/sdk/docs/quickstart-windows>

6.1.2.2 Configure for Google Cloud SDK

- Run the following at a command prompt:

```
> gcloud init
```

- Accept the option to log in using your Google user account:

```
To continue, you must log in. Would you like to log in (Y/n)? Y
```

- In your browser, log in to your Google user account when prompted and click *Allow* to grant permission to access Google Cloud Platform resources.
- At the command prompt, select a Cloud Platform project from the list of those where you have *Owner*, *Editor* or *Viewer* permissions:

```
Pick cloud project to use:
```

```
[1] [my-project-1]
```

```
[2] [my-project-2]
```

```
...
```

```
Please enter your numeric choice:
```

If you only have one project, “gcloud init” selects it for you.

- If you have the Google Compute Engine API enabled, “gcloud init” allows you to choose a default Compute Engine zone:

```
Which compute zone would you like to use as project default?
```

```
[1] [asia-east1-a]
```

```
[2] [asia-east1-b]
```

```
...
```

```
[14] Do not use default zone
```

```
Please enter your numeric choice:
```

“gcloud init” confirms that you have complete the setup steps successfully:

```
gcloud has now been configured!
```

```
You can use [gcloud config] to change more gcloud settings.
```

```
Your active configuration is: [default]
```

Reference: <https://cloud.google.com/sdk/docs/quickstart-windows>

6.1.2.3 Setup Kubernetes Engine on Google Cloud

- Setting a default project

```
> gcloud config set project [PROJECT_ID]
```

Replace [PROJECT_ID] with your project ID.

- Setting a default compute zone

```
> gcloud config set compute/zone [COMPUTE_ZONE]
```

where [COMPUTE_ZONE] is the desired geographical compute zone

- Get authentication credentials for the cluster

```
> gcloud container clusters get-credentials [CLUSTER_NAME]
```

- Creating the Deployment

```
> kubectl create deployment <DEPLOYMENT-NAME> --image=asia.gcr.io/trip-sharing-cp/<SERVICE-NAME>
```

- Exposing the Deployment

```
kubectl expose deployment <DEPLOYMENT-NAME> --type LoadBalancer \ --port 80 --target-port 8080
```

Reference: <https://cloud.google.com/kubernetes-engine/docs/quickstart>

6.1.2.4 Build and push docker images

- Build the docker image

```
> docker build -t asia.gcr.io/trip-sharing-cp/<SERVICE-NAME> .
```

- Push the docker image

```
> docker push asia.gcr.io/trip-sharing-cp/<SERVICE-NAME>
```

6.1.2.5 Setup Google pub/sub

- Create a topic

```
> gcloud pubsub topics create mail-sending-topic
```

- Create a subscription

```
> gcloud pubsub subscriptions create --topic mail-sending-topic mail-service-sub
```

6.1.2.6 Setup SendGrid

- Go to <https://signup.sendgrid.com/> and register a SendGrid account.



Let's Get Started

Review your plan and create an account.

Username •

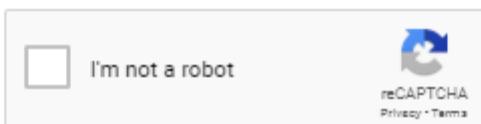
Password •

Must have more than 8 characters, including at least 1 letter and number.

Confirm Password •

Email Address •

You'll need access to this email address to verify your account.



I accept the [Terms of Service](#) and have read the [Services Privacy Policy](#)

[Create Account](#)

Figure 6-12: Sign up SendGrid

- Go to <https://app.sendgrid.com> and sign in
- From the menu bar on the left, click **Settings > API Keys** to navigate to API Keys management page
- Click **Create API Key** at the top right corner and follow instructions to create a new API Key

Create API Key

The screenshot shows a user interface for creating an API key. At the top, there is a field labeled "API Key Name" with a red asterisk indicating it is required. Below this is a section titled "API Key Permissions" with another red asterisk. Three options are listed:

- Full Access** (selected): Allows the API key to access GET, PATCH, PUT, DELETE, and POST endpoints for all parts of your account, excluding billing and Email Address Validation.
- Restricted Access**: Customize levels of access for all parts of your account, excluding billing and Email Address Validation.
- Billing Access**: Allows the API key to access billing endpoints for the account. (This is especially useful for Enterprise or Partner customers looking for more advanced account management.)

At the bottom right are two buttons: "Cancel" and "Create & View".

Figure 6-13: Create SendGrid API key

6.2 User guidelines

- ❖ Our system work well on Google Chrome browser for desktop, so we suggest using it to access our system.
- ❖ User visits <https://trip-sharing.net> and follows our guidelines bellow.

6.2.1 User sign up

- On any page, user click on “Đăng nhập” button.
- Sign in popup will be displayed, user click on “Đăng kí” link at the bottom of the popup.

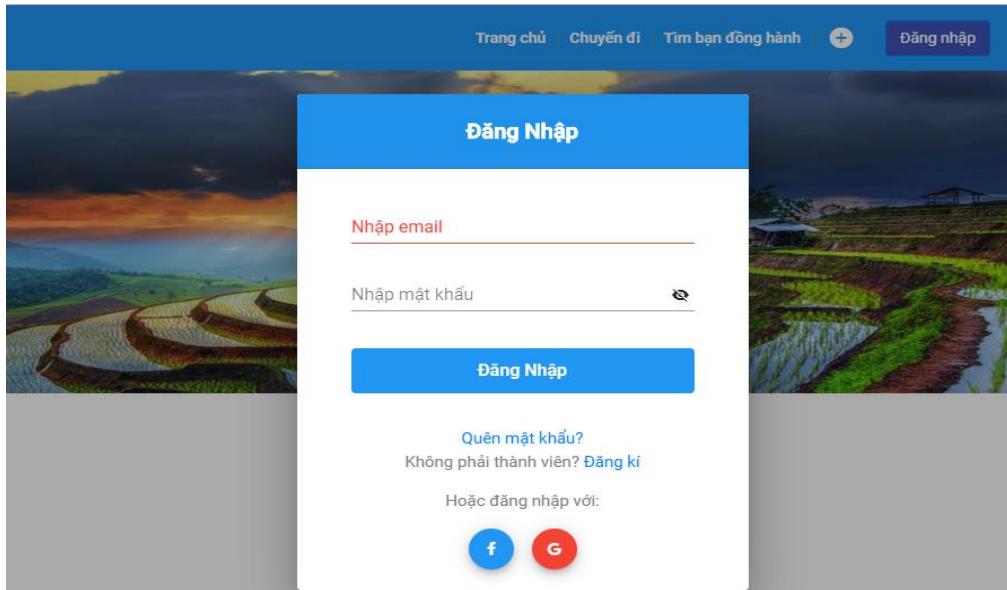


Figure 6-14: Sign in form

- Browser will redirect to sign up page. User fill in all needed field and click “Đăng kí” button.

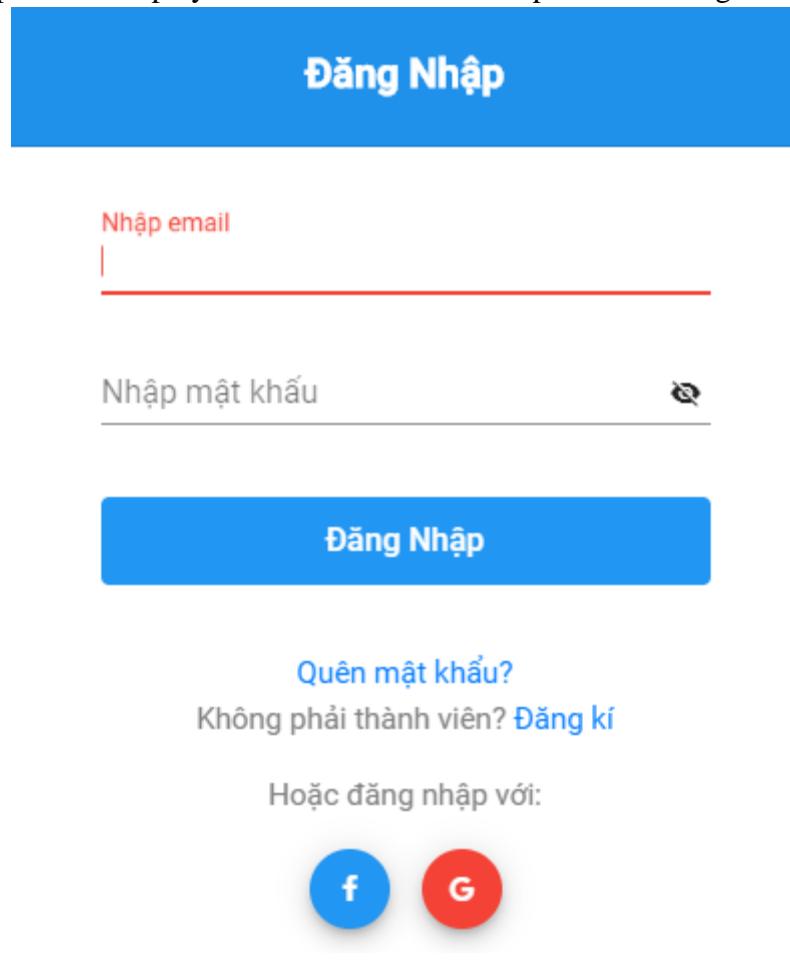
A screenshot of a web browser showing a sign-up form titled "Đăng Kí" (Sign Up) over a background image of rice terraces. The form includes fields for "Nhập email" (Enter email), "Nhập mật khẩu" (Enter password), and "Nhập mật lại khẩu" (Enter password again). There is a checkbox for accepting terms and conditions: "Tôi đồng ý với các điều khoản của Trip Sharing" (I agree with the terms and conditions of Trip Sharing). A "Đăng Kí" (Sign Up) button is present, along with social login options for Facebook and Google.

Figure 6-15: Sign up form

- System will send to user an email to confirm. User click on link on the email to complete.

6.2.2 User sign in

- On any page, user click on “Đăng nhập” button.
- Sign in popup will be displayed. User fill in email and password to sign in.



The image shows a sign-in form with a blue header containing the text "Đăng Nhập". Below the header are two input fields: one for "Nhập email" (Email input) and one for "Nhập mật khẩu" (Password input). To the right of the password input field is a small icon of a person. A large blue button at the bottom of the form also contains the text "Đăng Nhập". Below the form, there are links for "Quên mật khẩu?" (Forgot password?) and "Không phải thành viên? Đăng kí" (Not a member? Register). There is also a link for "Hoặc đăng nhập với:" (Or log in with:) followed by two social media icons: a blue circle with a white 'f' for Facebook and a red circle with a white 'G' for Google.

Figure 6-16: Sign in form

- User now signed in the system and redirect to homepage.

6.2.3 User sign out

- On any page, user click on image profile on the header.
- System will display a list of options, then user click on “Đăng xuất” option.

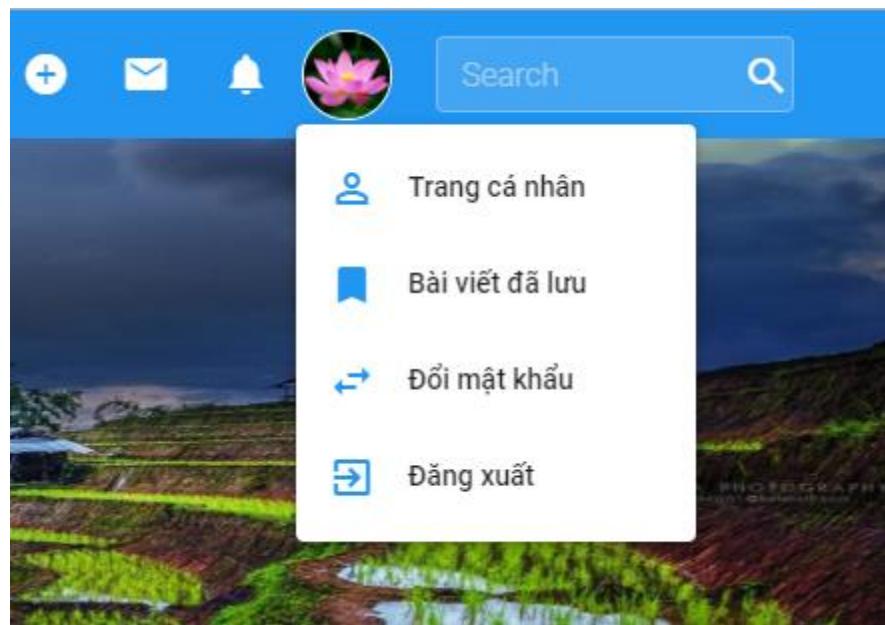


Figure 6-17: User's menu

- User now signed out of the system.

6.2.4 User update profile

- On any page, user click on image profile on the header.
- System will display a list of options, then user click on “Trang cá nhân”.
- Browser will redirect to user’s personal page.
- User click on “Chỉnh sửa trang cá nhân” at the left side of the page.

Figure 6-18: Update profile screen

- A popup of user information will be display, then user edit the field user want to change and click on “Tiếp tục” button.

- System will display update interest topics popup. User change the interest topics then click on “Cập nhật” button.

6.2.5 User searches posts

- ❖ *Search posts by location:*
 - From home page, user enter the location to the search input.
 - System will suggest list of location base on Google Maps. User select one location.

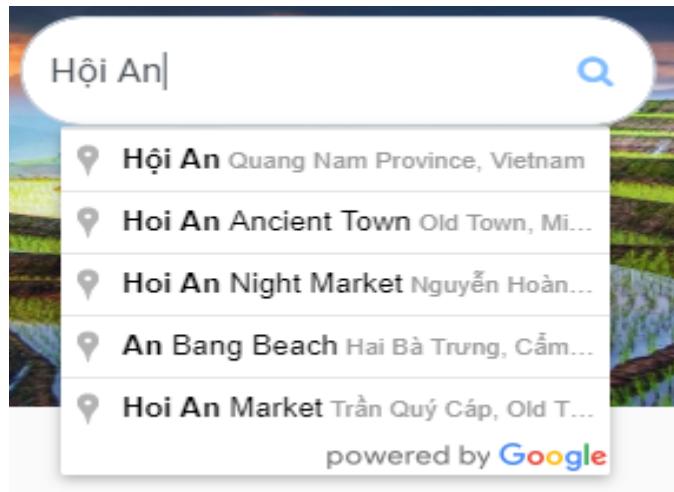


Figure 6-19: Search by Google Map location

- Browser will redirect to search-result page.
- ❖ *Search posts by keyword:*
 - From any page, user enter search keyword into search field at the top-right of the page then press “Enter”.

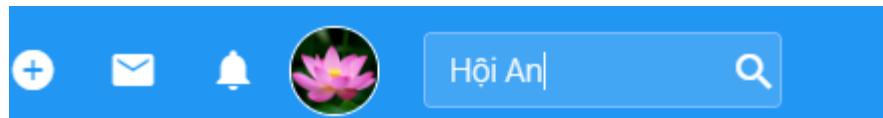


Figure 6-20: Search by keyword

- Browser will redirect to search-result page

6.2.6 User searches other users

- From any page, user enter search keyword into search field at the top-right of the page then press “Enter”.
- Browser will redirect to search-result page
- User switch to “Mọi người” tab to see searched users.

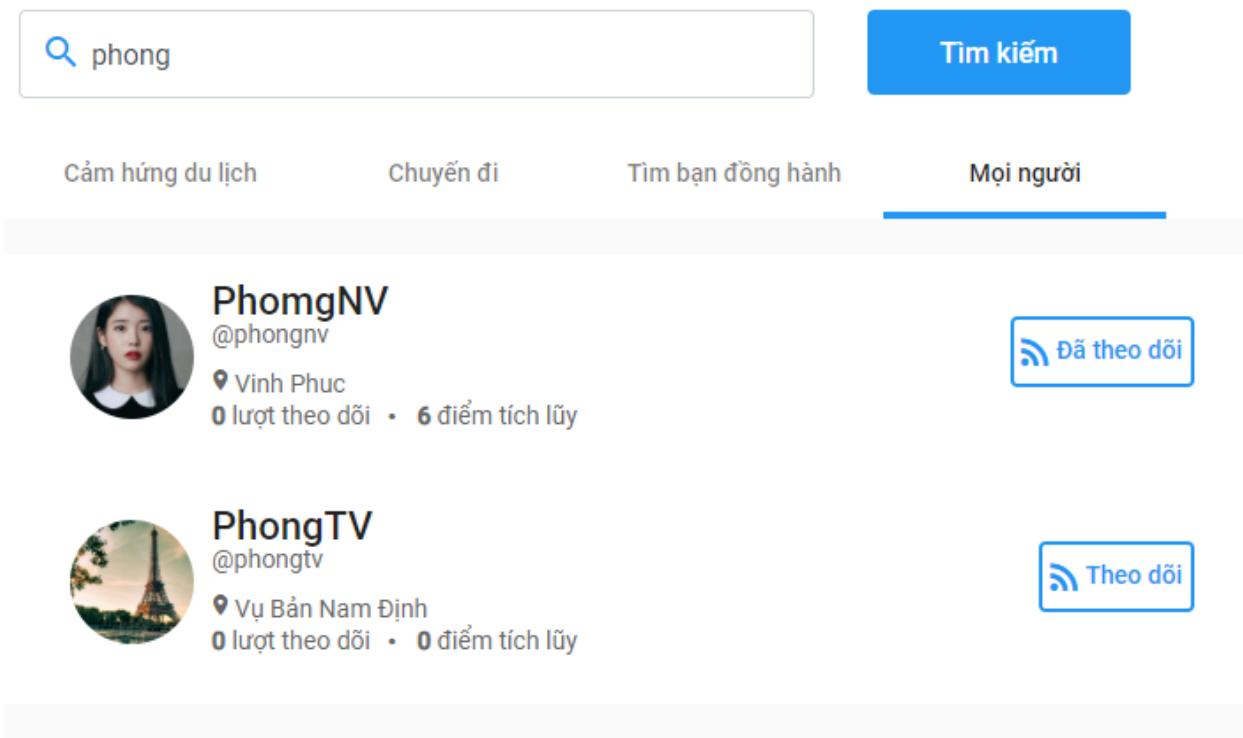


Figure 6-21: Search a user

6.2.7 User creates posts

- From any page, click on “+” button on the header. A list of post type will be displayed.
- User choose post type user want to create.

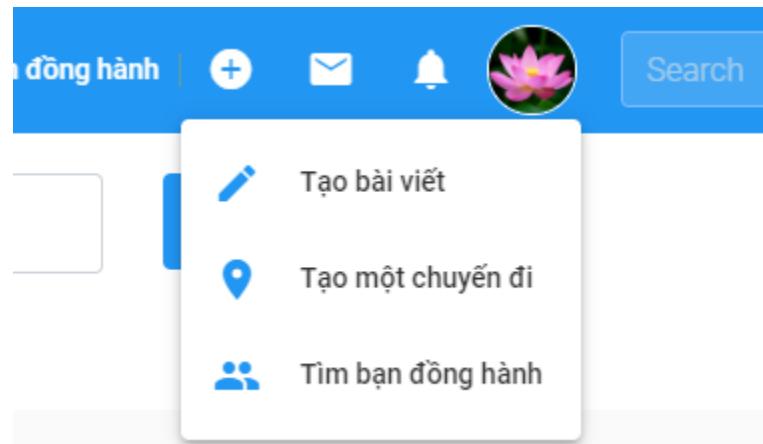
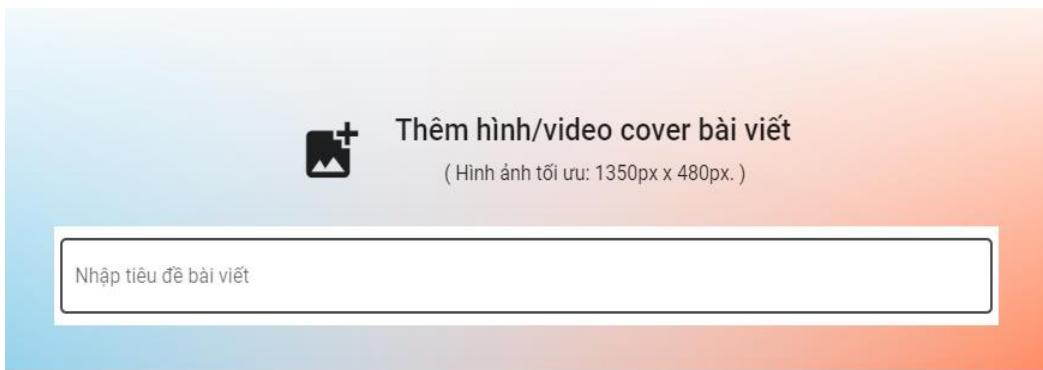


Figure 6-22: Create-post options

- Browser will redirect to create-post page.
 - ❖ Create an article



1. Địa điểm chuyển đi

Nhập địa điểm bạn muốn đến 

2. Viết bài cho chuyến đi



Figure 6-23: Create an article page

❖ *Create a virtual trip*

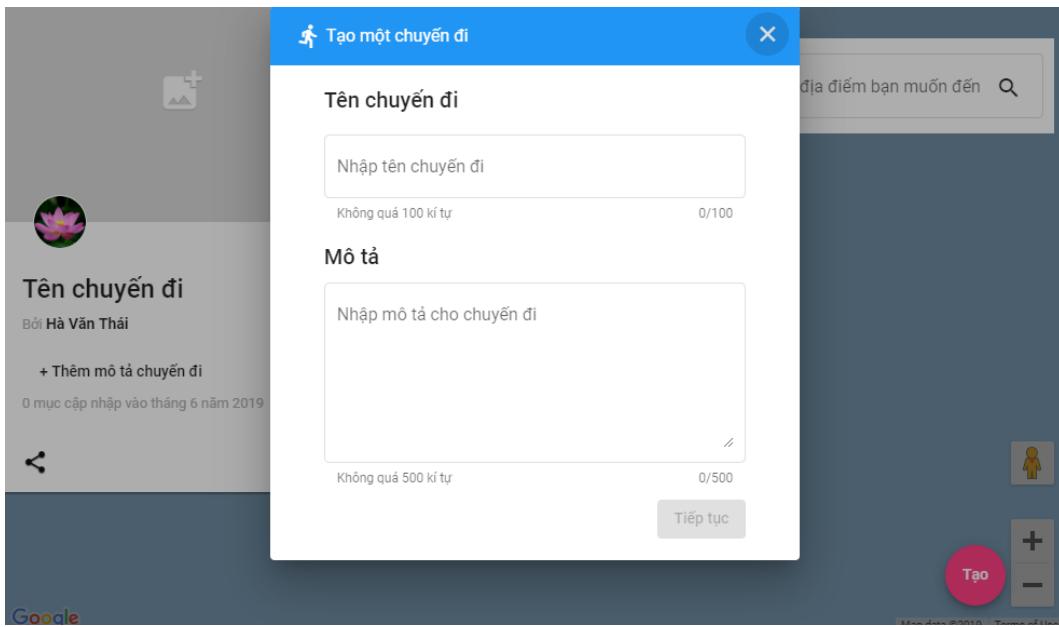


Figure 6-24: Create a virtual trip page

❖ *Create a finding-companion post*

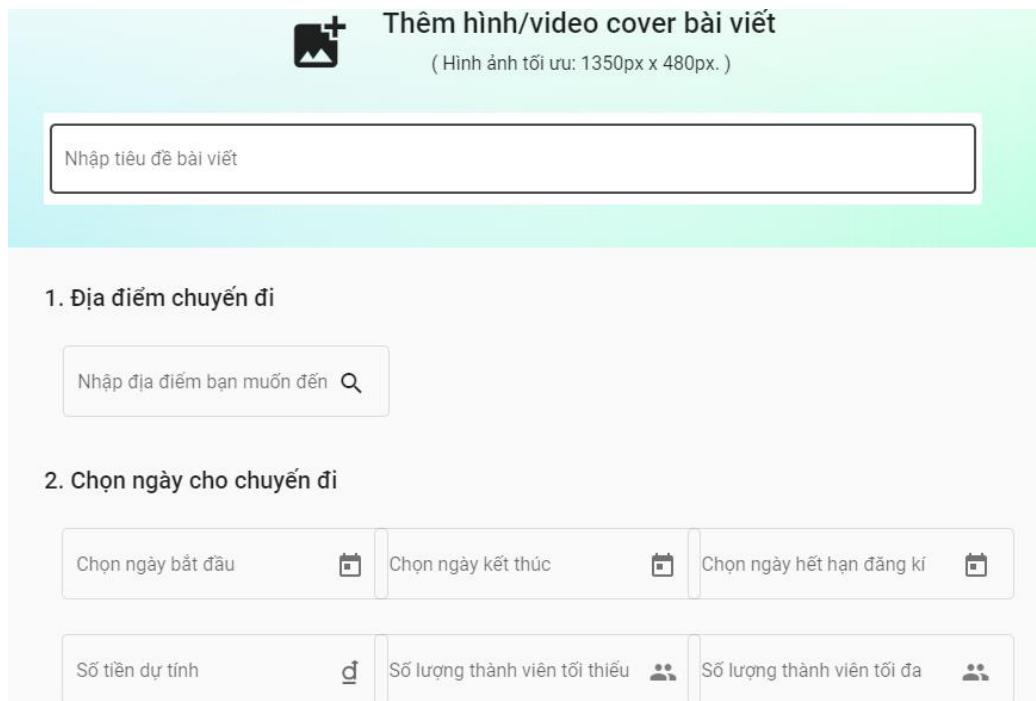


Figure 6-25: Create a finding-companion post page

6.2.8 User chats with other users

- User go to the personal page of the user that he/she want to chat with.
- User click on “Gửi tin nhắn” button.

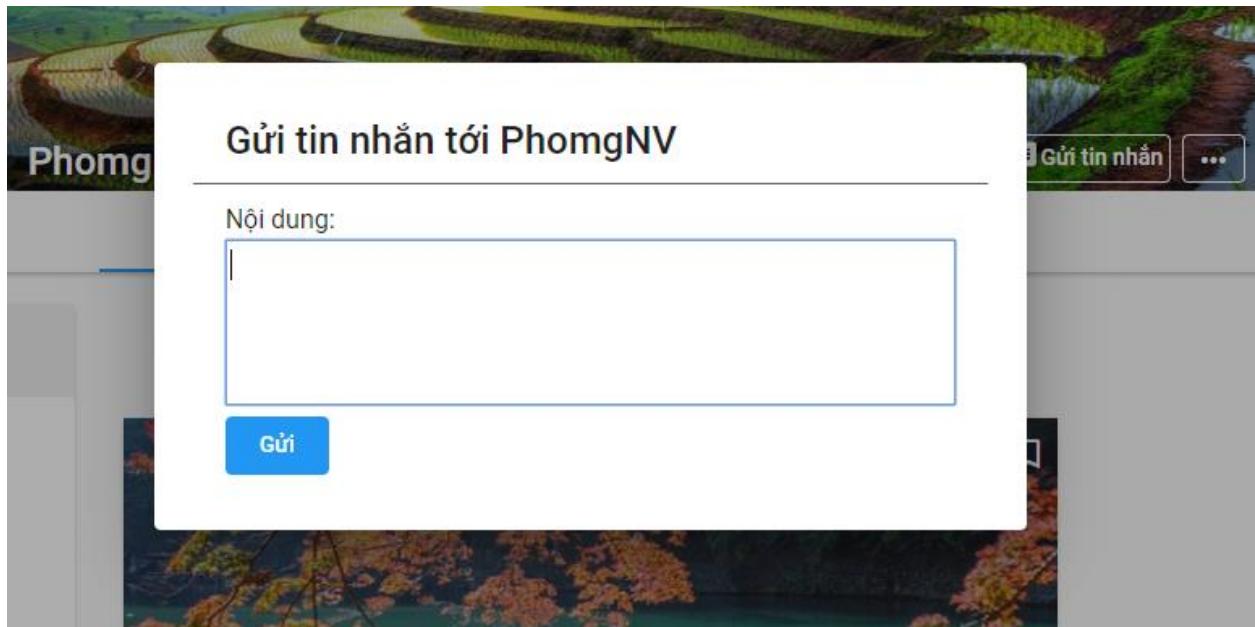


Figure 6-26: Send a message to another

- A popup will be displayed. User enter message then click on “Gửi” button.

- From now on, a conversation between two users is initiated. They can quickly see received message and send new message by click on message icon on the header and select conversation.

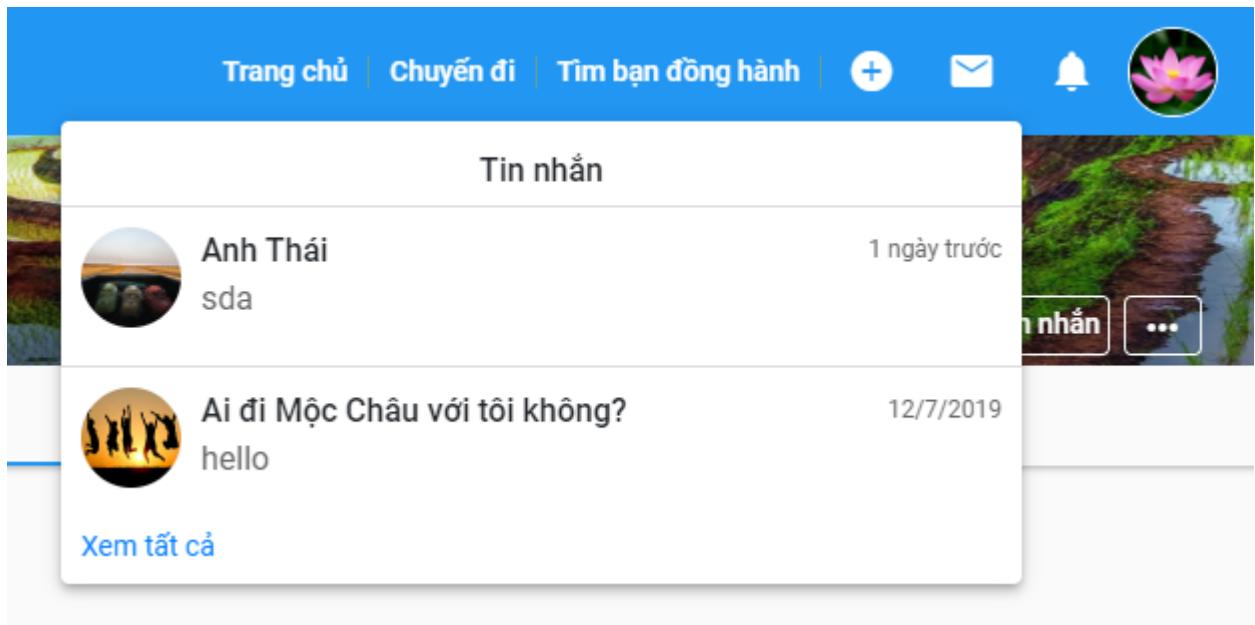


Figure 6-27: List of conversations

6.2.9 User joins a group

- User go to the finding-companion post page which he/she wants to join.
- At the right side of the page, click on “Tham gia” button.

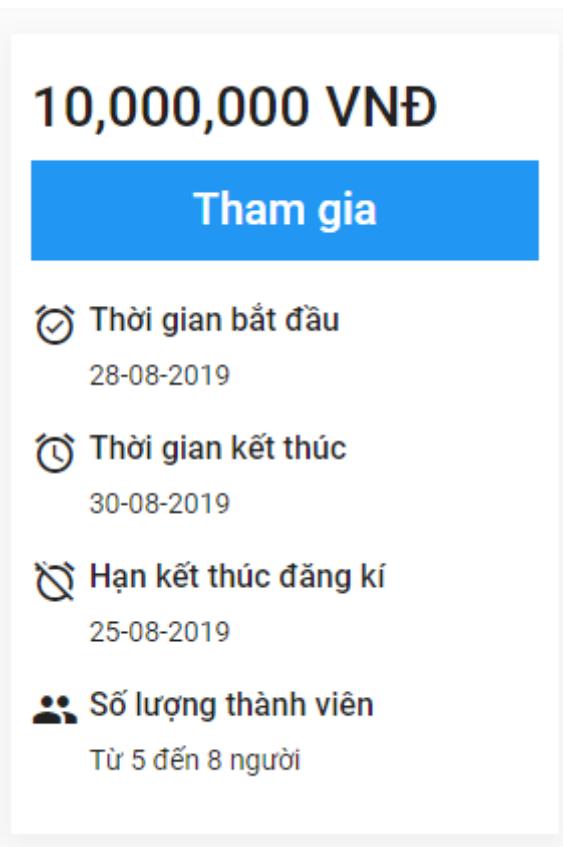


Figure 6-28: Finding-companion post information box

- System will send a request to the post's author
- If the author accepts your join request, system will notify to you. Now you can see the group chat in the list of your conversations.

6.2.10 User follows other users and views followings/followers

- On any page, user click on image profile on the header.
- System will display a list of options, then user click on “Trang cá nhân”.
- Browser will redirect to user's personal page.
- User click on “n – Người đang theo dõi bạn” to see followers or “n – Người bạn đang theo dõi” to see followings.

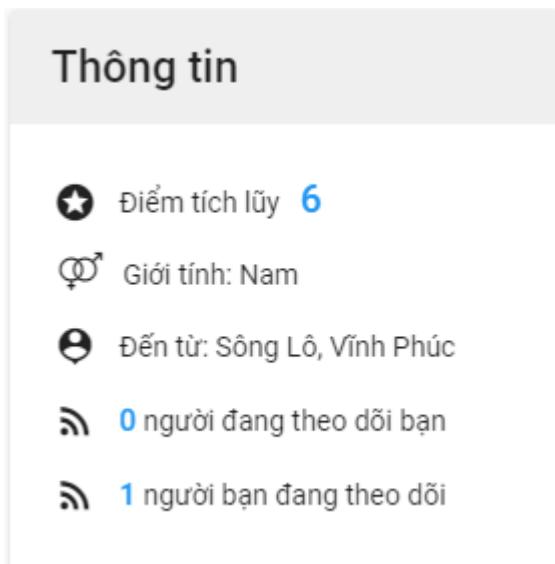


Figure 6-29: User profile

- System will display a popup with list of followings/followers

6.2.11 User bookmarks posts and view bookmarked posts

- On any page that display list of posts, click on bookmark icon at the top-right of post item.



Figure 6-30: A post item

- Or on post detail page, click on bookmark icon at the left side of the page.

lìm trong rừng phong lá đỏ, bên những ng
tới nước Nhật Bản một lần trong đời.



Những đứa trẻ ngày đó lớn lên, khát khao
Bản mê đắm người trẻ bởi thế giới công n
truyền thông văn hóa được gìn giữ suốt c

Figure 6-31: Post interaction in detail post page