ASSIGNMENT FRONT SHEET

Qualification	**BTEC Level 5 HND Diploma in Computing**		
Unit number and title	**WEBG301 – Project Web**		
Submission date	12/07/2022	**Date Received 1st submission**	
Re-submission Date		**Date Received 2nd submission**	
Group	**Student ID & Name**	**Final Score**	**Student's signature**
	1. GCC200001 BUI NGOC VINH KHANH		KHANH
	2. GCC200179 NGUYEN PHUONG DUY		DUY
	3. GCC200197 HUYNH PHAN THAI		THAI
Student declaration I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice.			
Class	GCC0902	**Assessor name**	TRAN THI KIM KHANH

OBSERVATION RECORD

Student name:	**NGUYEN PHUONG DUY**		
Description of activity undertaken			
I am team leader 4. Our project is to design a weblog. I am in charge of coding the main functions of the weblog such as login, registration, add, update, delete user information. Attached is the website interface design			
Assessment criteria			
How the activity meets the requirements of the assessment criteria			
Student signature:	**DUY**	**Date:**	**12/7/2022**
Assessor name:			
Assessor signature:		**Date:**	

Student name:	**BUI NGOC VINH KHANH**		
Description of activity undertaken			
I am a member of team 4. Our project is to design a weblog. I am in charge of coding the main functions of the weblog such as login, registration, add, update, delete friends and posts and user interfaces. Attached is the website interface design.			
Assessment criteria			
How the activity meets the requirements of the assessment criteria			
Student signature:	**KHANH**	**Date:**	**12/7/2022**
Assessor name:			
Assessor signature:		**Date:**	

Student name:	**HUYNH PHAN THAI**		
Description of activity undertaken			
I am a member of team 4. Our project is to design a weblog. I am in charge of website design and report creation			
Assessment criteria			
How the activity meets the requirements of the assessment criteria			
Student signature:	**THAI**	**Date:**	**12/7/2022**
Assessor name:			
Assessor signature:		**Date:**	

❒ **Summative Feedback:**		❒ **Resubmission Feedback:**
Grade:	**Assessor Signature:**	**Date:**
Internal Verifier's Comments:		
IV Signature:		

Table of Contents

OBSERVATION RECORD 2
Group Report Structure 7
Chapter 1 – Users' requirements 7
Chapter 2 – System Design 7
Chapter 3 – Implementation 14
Chapter 4 – Conclusion 27
References 28

Group Report Structure

Chapter 1 – Users' requirements

In the age of technology 4.0. The need to receive information, entertainment, and relaxation has increased to the pinnacle. To keep up with the times. Our company KTD has launched a weblog to help people easily exchange information, make friends, view posts quickly. With fast speed and high efficiency, our company's weblog has surpassed many other weblogs. However, our weblog is completely free and completely confidential user information.

Chapter 2 – System Design

2.1 About Symfony

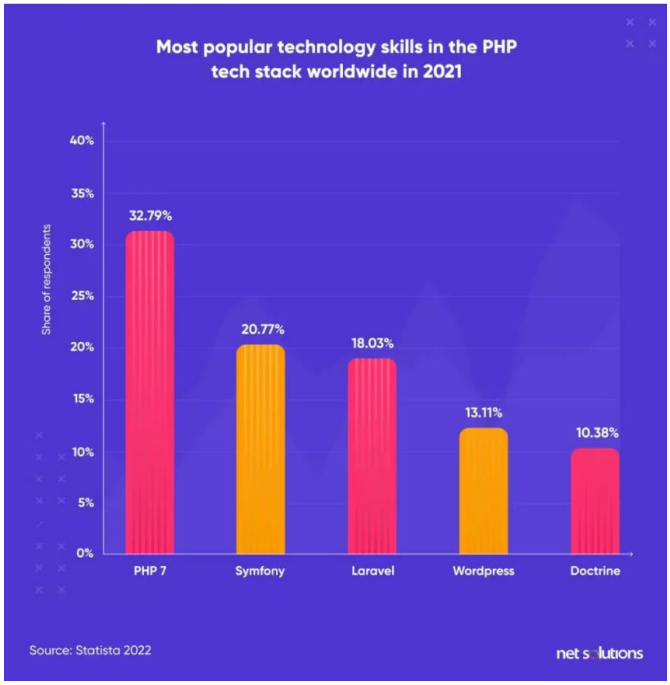
Symfony is an open-source PHP web app framework for developers seeking a simple and elegant toolkit for building full-featured web applications. It works with an independent library and the PHP Unit.

Symfony is heavily influenced by the web application frameworks Ruby on Rails, Django, and Spring. Many open-source projects, including Composer, Drupal, and phpBB, use its components.

Choosing the right web development framework is one of the vital steps toward building a web solution. PHP has evolved rapidly and is presently being used extensively for producing simple and complex web applications.

Today, 79% of all the websites in the world are created with the PHP programming language. A web application can be built either in pure PHP or one of its frameworks.

Symfony has grown exceptionally in recent years. The graph below shows that over 20% of businesses are using theSymfony web framework to web applications in PHP.



a. What is the Symfony Framework?

"Symfony is a set of PHP Components, a Web Application framework, a philosophy, and a community — all working together in harmony.

Generally speaking, the Symfony framework is a flexible framework that does away with cumbersome coding, thus saving development time. The philosophy behind Symfony is to help create software that streamlines the entire product development process for the developers.

Symfony has all the characteristics that are expected from a modern framework. This blog lists some features that make Symfony framework one of the best PHP frameworks.

b. Is Symfony framework frontend or backend?

Symfony is a feature-rich back-end framework that is used to build complex applications. Many developers still prefer the lightweight Silex micro-framework or the Symfony MicroKernel, and Symfony is a widely used application framework among open-source developers.

c. Why Symfony: 5 Reasons to Choose the Framework for PHP Development

- **High Flexibility**

The Symfony PHP framework is a well-organized, feature-rich PHP framework whose architecture paves the way for developers to build sustainable web applications in the easiest way possible which further enhances the users' experience.

This PHP framework is incredible because of its two most striking technological benefits: Bundles and Components.

The Bundle is similar to a plugin. The primary benefit of Bundles is that they are decoupled, which implies they can be reused and reconfigured further for many applications to bring down the overall development cost.

Components help reduce routine tasks as they can be used independently by adding your custom modules without hurting the architecture. All-in-all, 30 helpful Symfony components facilitate the web development process. The components of the Symfony framework can also be used exclusively in other frameworks (for instance, Laravel) or simple PHP solutions.

relationship-between-application

Bundles and Components help to eliminate the inflexible dependencies in the architecture by giving you the liberty to get reused. Fewer the dependencies, the easier it becomes to introduce changes sans the risk of breaking other parts of the system.Thus, you can adapt the solution to any requirements and user scenarios to create a highly flexible application.

- **Customization**

The Symfony framework is packed with exclusive custom features and functionalities for developers and businesses. With Symfony as your PHP framework, you can make your web application as user-oriented as you aim for. The advanced OOPS service architecture of Symfony allows to scale up projects.

Symfony offers the following types of customization:

Full-stack – Build a complex product with multiple functionalities

Brick by brick – You can create your custom framework if you wish to build an application with specific features and selective functions.

- **Refined MVC Architecture**

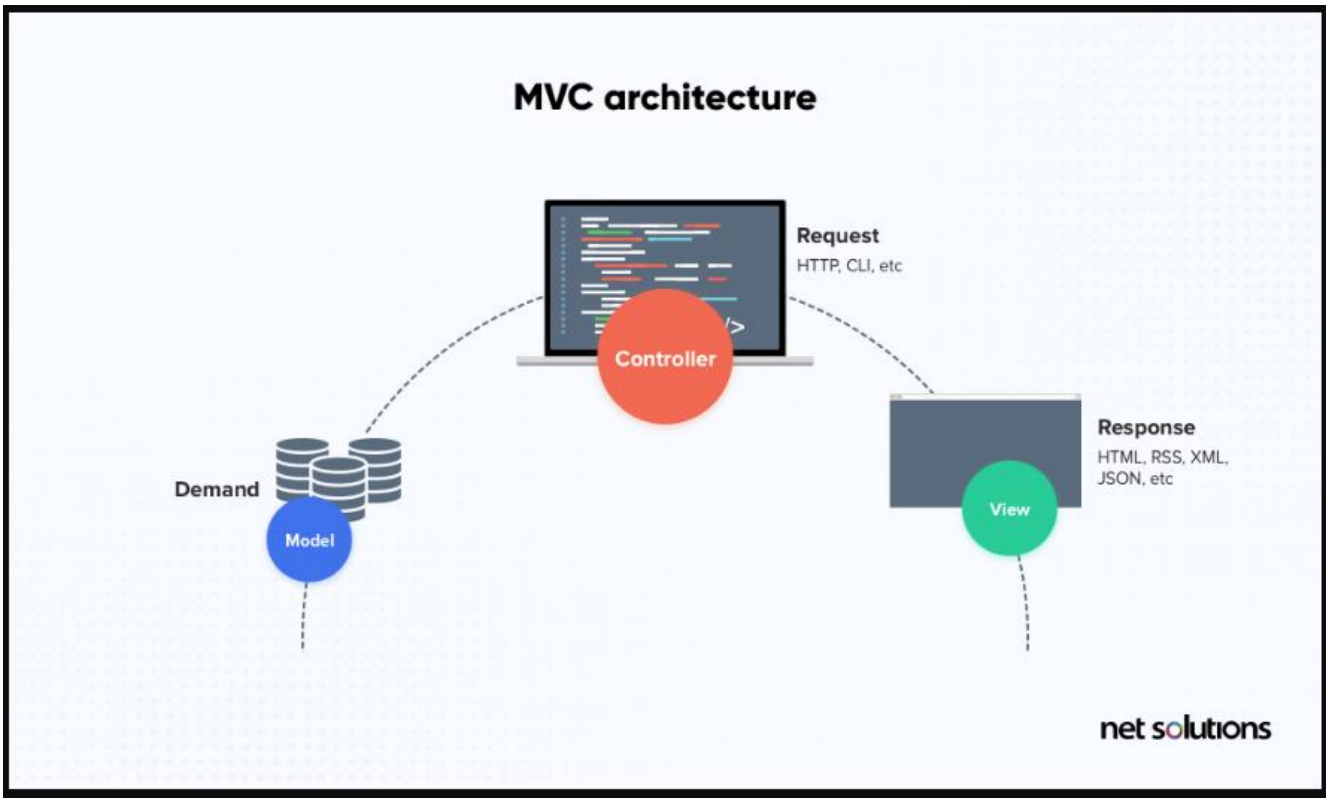
The MVC architecture of the Symfony framework is ideal for systematic and scalable web development projects. Utilizing MVC as the core of its web development, the Symfony framework makes sure that the project has an organized file structure distributed into Model, View, and Controller.

The Model – focuses on the business logic of the web application.

The View – provides the specific model into a web page visible to the user.

The Controller – as per the user actions, prompts alterations to the Model or View when needed.

The true benefit of using MVC with Symfony is that the developers get the liberty to easily separate the Model (business logic) and View (user presentation), allowing for better maintainability.



- **Large community**

This is one of the critical aspects that define the framework's survivability and stands good. Symfony's official website declares 3000+ contributors with a passionate group of over 600,000 developers from more than 120 countries – the number is several times bigger than that of other PHP framework communities. Moreover, when writing this article, the Symfony framework has 20,758 stars and 1,304 watchers and has been forked 6,901 times on Github.

- **Symfony Architecture: Decoding Various Components of the PHP Framework**

Symfony components are independent PHP libraries that provide unique functionality in any PHP application. Currently, the Symfony framework contains more than 30 high-quality components. Let's take a look at how Symfony components are used:

d. Symfony Framework Components

- Finder

The Finder component uses an intuitive fluent interface to find files and directories based on various criteria (name, file size, modification time, and so on). The component includes a plethora of methods for defining search criteria. Because they all implement a fluent interface, they can all be chained.

- Filesystem

The Filesystem component offers platform-independent utility for file system operations and file/directory path manipulation. It includes essential filesystem utilities.

- ClassLoader

If your project classes adhere to standard PHP conventions, this component will automatically load them. This component has been considered obsolete since Symfony 3.3; instead, use Composer class loading.

- DependencyInjection

The DependencyInjection component implements a PSR-11 compatible service container, allowing you to benchmark and centralize the way elements are built into your application.

- EventDispatcher

The EventDispatcher component provides tools for your application components to communicate with one another by dispatching and listening to events. To make all of this possible and your projects genuinely extensible, the Symfony EventDispatcher component implements the Mediator and Observer design patterns.

- Serializer

The Serializer component converts objects into a specific format (XML, JSON, YAML, etc.) and vice versa.

- ExpressionLanguage

The ExpressionLanguage component includes an engine for compiling and evaluating expressions and is a one-liner that returns a value.

- Workflow

The workflow component allows you to define a process or a life cycle for your object in an object-oriented manner. Each step is called a place, and you also define transitions, which describe the action taken to move from one place to another.

HttpFoundation

The HttpFoundation component defines the HTTP specification's object-oriented layer. An object-oriented layer replaces the default PHP global variables and functions in the Symfony HttpFoundation component.

- Form

Forms can be created, processed, and reused using the Form component.The Form component is a tool that can assist you in resolving the issue of allowing end-users to interact with and modify data in your application.

- HttpKernel

Using the EventDispatcher component, the HttpKernel component provides a structured process for converting a Request into a Response. It's versatile enough to be used to build a full-stack framework (Symfony), a micro-framework (Silex), or a refined CMS system (Drupal).

- Routing

When your application receives a request, it invokes a controller action to produce a response. The routing configuration specifies which action should be performed for each incoming URL, and it also offers the ability to generate SEO-friendly URLs.

e. Symfony Framework Bundle

A Symfony bundle is a collection of files and folders that have been organized in a specific way. The bundles are designed to be reusable across multiple applications, and the main application is packaged as a bundle, commonly referred to as AppBundle.

A bundle can be packaged specifically for an application, for example, AdminBundle (admin section), BlogBundle (site's blog), etc. Such bundles cannot be shared across applications. Instead, we can model a specific part of the application, such as blogs, as a generic bundle that can be copied from one application to another to reuse the blog functionality.

Controller − All controllers need to be placed here.

DependencyInjection − All dependency injection-related code and configuration need to be placed here.

Resources/config − Bundle-related configurations are placed here.

Resources/view − Bundle-related view templates are placed here.

Resources/public − Bundle-related stylesheets, JavaScripts, images, etc., are placed here.

Tests − Bundle-related unit test files are placed here.

f. Symfony Framework Examples

Some popular sites that are built with the Symfony PHP framework:

- Spotify

Spotify works with Symfony to handle its users' accounts; according to statistics, the streaming platform had 286 million active users in 2020. Adopting open-source software powered by PHP and Symfony is helping businesses boost their solution and provide new features.

- Dailymotion

Like Spotify, Dailymotion is a streaming service; only it offers access to videos instead of music. The Dailymotion company created its website in PHP, and after a few years, they decided to abandon the framework they created in favor of Symfony.

- Trivago

The Trivago company owns another Symfony-based booking application that serves millions of users and allows them to compare hotel accommodation prices. By creating numerous functionalities on the page, Symfony enables Internet users to have the best user experience possible when searching and comparing offers.

- Drupal Console

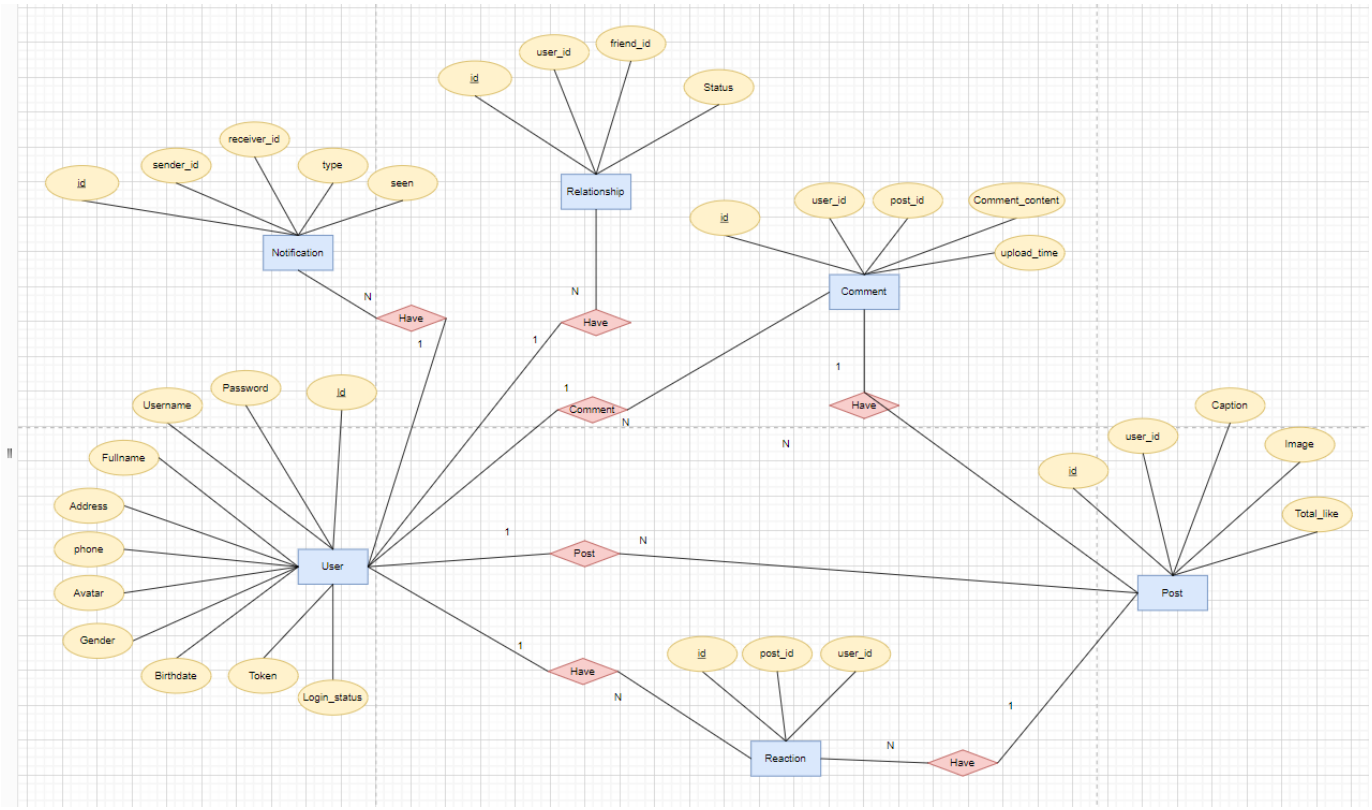
Drupal Console is a Symfony Console-based component used for debugging and creating event lists. The event Dispatcher can provide access to all application events due to a dependency injection container. (netsolutions, 2022).

2.2. Git/GitHub

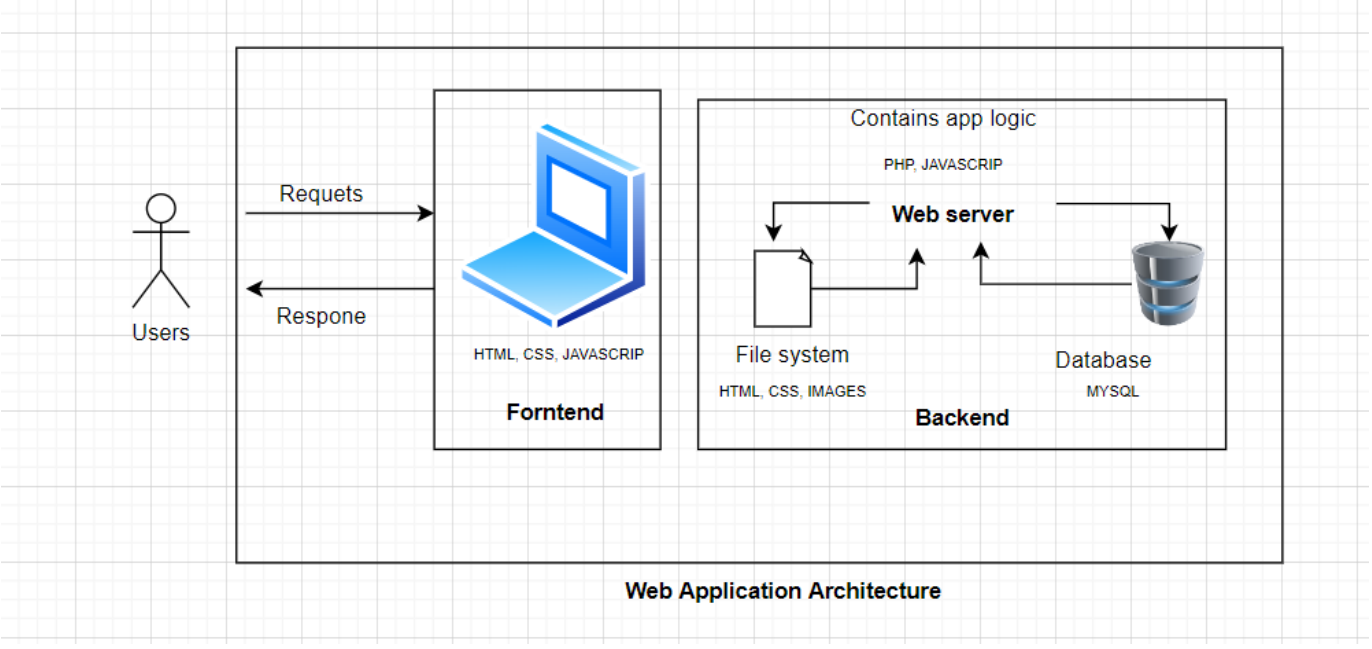
2.3 Use Case Diagram

2.4 Entity Relationship Diagram

2.5 Sitemap



2.6 System Architecture Diagram



Chapter 3 – Implementation

3.1 Sample Source Code

```
<?php
  namespace App\Controller;

  use App\Entity\Notification;
  use App\Entity\Relationship;
  use App\Repository\NotificationRepository;
  use App\Repository\ReactionRepository;
  use App\Repository\RelationshipRepository;
  use App\Repository\UserRepository;
  use Doctrine\Persistence\ManagerRegistry;
  use phpDocumentor\Reflection\Types\This;
  use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
  use Symfony\Component\HttpFoundation\JsonResponse;
  use Symfony\Component\HttpFoundation\Request;
  use Symfony\Component\Routing\Annotation\Route;
  use Symfony\Component\Security\Core\Security;

  class RelationShipController extends AbstractController
  {
    /**
     * @Route ("/friendList/{userId}", name="app_friend_list", methods={"GET"})
     */
    public function friendListAction($userId, NotificationRepository $notificationRepository,
UserRepository $userRepository, RelationshipRepository $relationshipRepository)
    {
      //get information navbar
      $inforNavBar = $userRepository->getUserInforNavBar($this->getUser()->getId());

      //get total notification of like and comment
      $liekNotification = $notificationRepository->getLikeFromOtherUser($this->getUser()->getId());
      $commentNotification = $notificationRepository->getCommentFromOtherUser($this->getUser()-
>getId());
      $totalLikeAndCommentNotification = $liekNotification[0]['total_like'] +
$commentNotification[0]['total_comment'];

      //get notification of invite friend
      $inviteFriend = $notificationRepository->getInvitefriend($this->getUser()->getId());

      //get detail notification
```

```
        $likeAndCommentDetail = $notificationRepository->getCommentAndLikeDetailFromOtherUser($this->getUser()->getId());
        $inviteFriendDetail = $notificationRepository->getInviteFriendDetail($this->getUser()->getId());
        $friendList = $relationshipRepository->getFriendList($userId);

        return $this->render('profile/profileFriendList.html.twig',[
            'inforNavBar' => $inforNavBar,
            'countlikeAndComment' => $totalLikeAndCommentNotification,
            'countInviteFriend' => $inviteFriend,
            'likeAndCommentDetail' => $likeAndCommentDetail,
            'inviteFriendDetail' => $inviteFriendDetail,
            'friendList' => $friendList
        ]);
//          return new JsonResponse(['friendList' => $friendList]);

    }

    /**
     * @Route ("/sendInviteFriend", name="api_add_friend", methods={"PUT"})
     */
    public function sendInviteFriendAPI(Request $request, UserRepository $userRepository, RelationshipRepository $relationshipRepository, ManagerRegistry $managerRegistry)
    {
        $request = $this->tranform($request);
        $friendId = $request->get('userId');
        $sendToFriend = $userRepository->find($friendId);

        if($sendToFriend)
        {
            $database = $managerRegistry->getManager();

            $this->saveRelationShip($sendToFriend, $database);
            $this->saveInviteFriend($sendToFriend, $database);

            $countStatusRelationShip = $relationshipRepository->checkRelatonshipStatus($this->getUser()->getId(), $friendId);

            if($countStatusRelationShip[0]['friendStatus'] == 2)
            {
                //update friend status sender
                $relationshipRepository->updateStatus($this->getUser()->getId(), $friendId);

                return new JsonResponse([
```

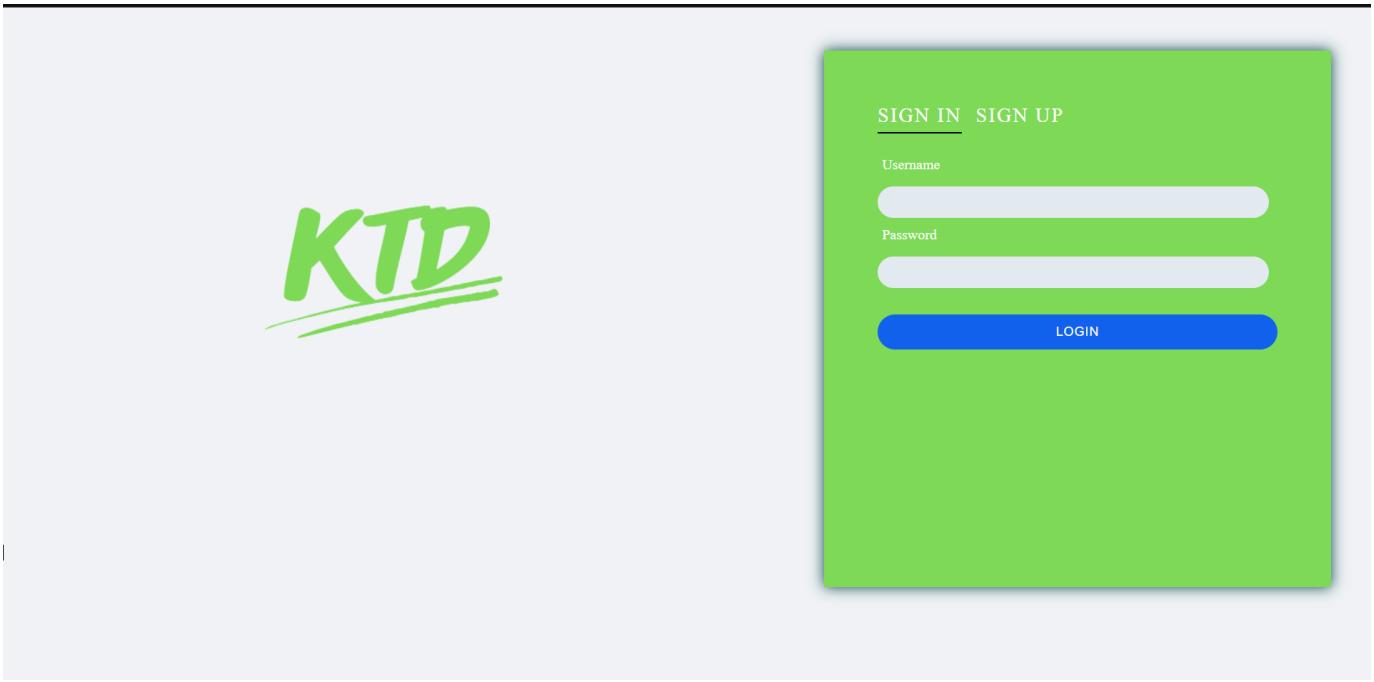
```
            'status_code' => 200,
            'Message' => 'Accept friend to user with id: '.$friendId
        ]);
    }

    return new JsonResponse([
        'status_code' => 200,
        'Message' => 'Has send invite friend to user with id: '.$friendId
    ]);
  }
  else
  {
    return new JsonResponse([
        'status_code' => 400,
        'Message' => 'Not found user with id: '.$friendId
    ]);
  }

}
public function saveRelationShip($sendToFriend, $database)
{
  $relationShip = new Relationship();
  $relationShip->setUser($this->getUser());
  $relationShip->setFriend($sendToFriend);
  $relationShip->setStatus('0');

  $database->persist($relationShip);
  $database->flush();
}
public function saveInviteFriend($sendToFriend, $database)
{
  $inviteFriendNotifical = new Notification();
  $inviteFriendNotifical->setSender($this->getUser());
  $inviteFriendNotifical->setReceiver($sendToFriend);
  $inviteFriendNotifical->setType('invite');
  $inviteFriendNotifical->setSeen('no');

  $database->persist($inviteFriendNotifical);
  $database->flush();
}

/**
 * @Route ("/acceptFriend", name="api_accept_friend", methods={"PUT"})
```

```
     */
    public function acceptFriendAPI(Request $request, UserRepository $userRepository,
RelationshipRepository $relationshipRepository, ManagerRegistry $managerRegistry)
    {
        $request = $this->tranform($request);
        $senderId = $request->get('senderId');
        $sender = $userRepository->find($senderId);

        if($sender)
        {
            // save new friend
            $relationShip = new Relationship();
            $relationShip->setUser($this->getUser());
            $relationShip->setFriend($sender);
            $relationShip->setStatus('1');

            $database = $managerRegistry->getManager();
            $database->persist($relationShip);
            $database->flush();

            $countStatusRelationShip = $relationshipRepository->checkRelatonshipStatus($this-
>getUser()->getId(), $senderId);

            if($countStatusRelationShip[0]['friendStatus'] == 2)
            {
                //update friend status sender
                $relationshipRepository->updateStatus($this->getUser()->getId(), $senderId);
            }

            return new JsonResponse([
                'status_code' => 200,
                'Message' => 'Accept friend to user with id: '.$senderId
            ]);
        }
        else
        {
            return new JsonResponse([
                'status_code' => 400,
                'Message' => 'Not found user with id: '.$senderId
            ]);
        }

    }
```

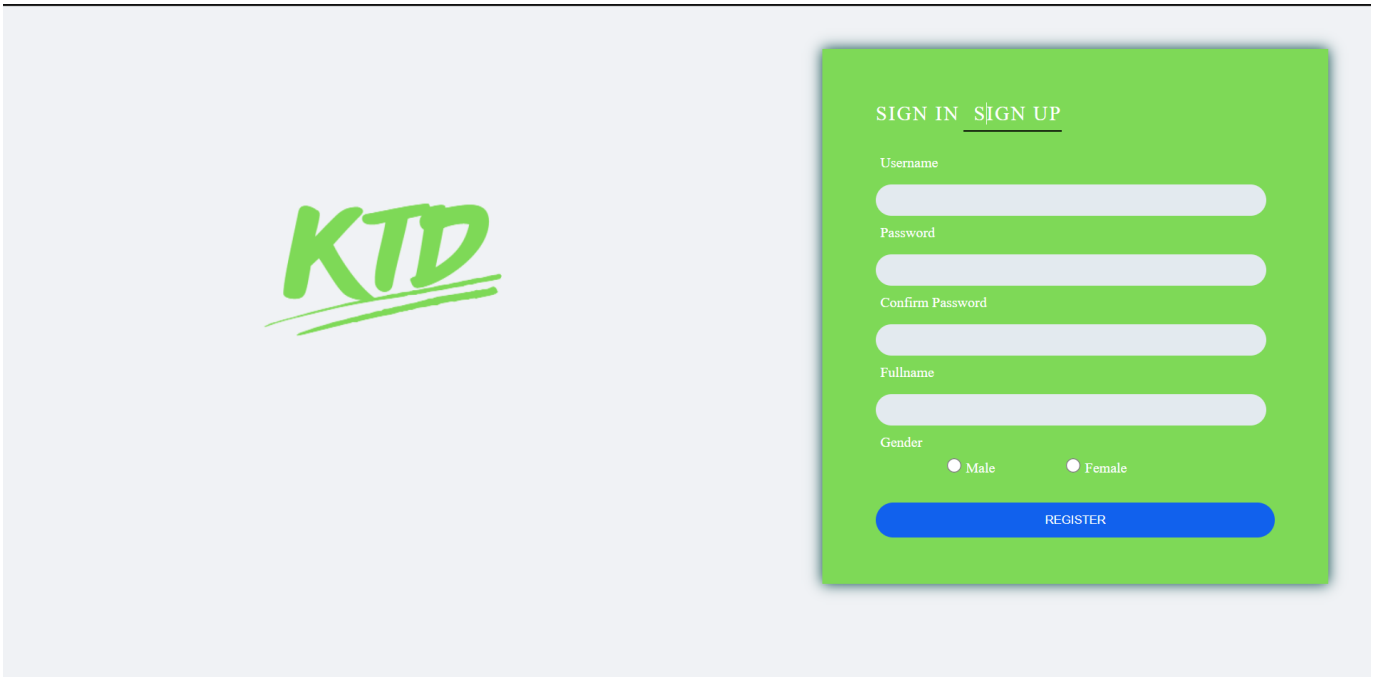
```
    /**
     * @Route ("/unFriend", name="api_unFriend", methods={"DELETE"})
     */
    public function unFriendAPI(Request $request, RelationshipRepository $relationshipRepository)
    {
        $request = $this->tranform($request);
        $idWantToUnfriend = $request->get('friendId');

        $unFriendResult = $relationshipRepository->unfriend($this->getUser()->getId(),
$idWantToUnfriend);

        if($unFriendResult)
        {
            return new JsonResponse(['status_code' => 200, 'Message' => 'Success unfriend with user id:
'.$idWantToUnfriend]);
        }
        else
        {
            return new JsonResponse(['status_code' => 400, 'Message' => 'Fail unfriend with user id:
'.$idWantToUnfriend]);
        }

    }

    public function tranform($request){
        $data = json_decode($request->getContent(), true);
        if($data === null){
            return $request;
        }
        $request->request->replace($data);
        return $request;
    }
}
```

3.2 Images of final Application

a. Login interface

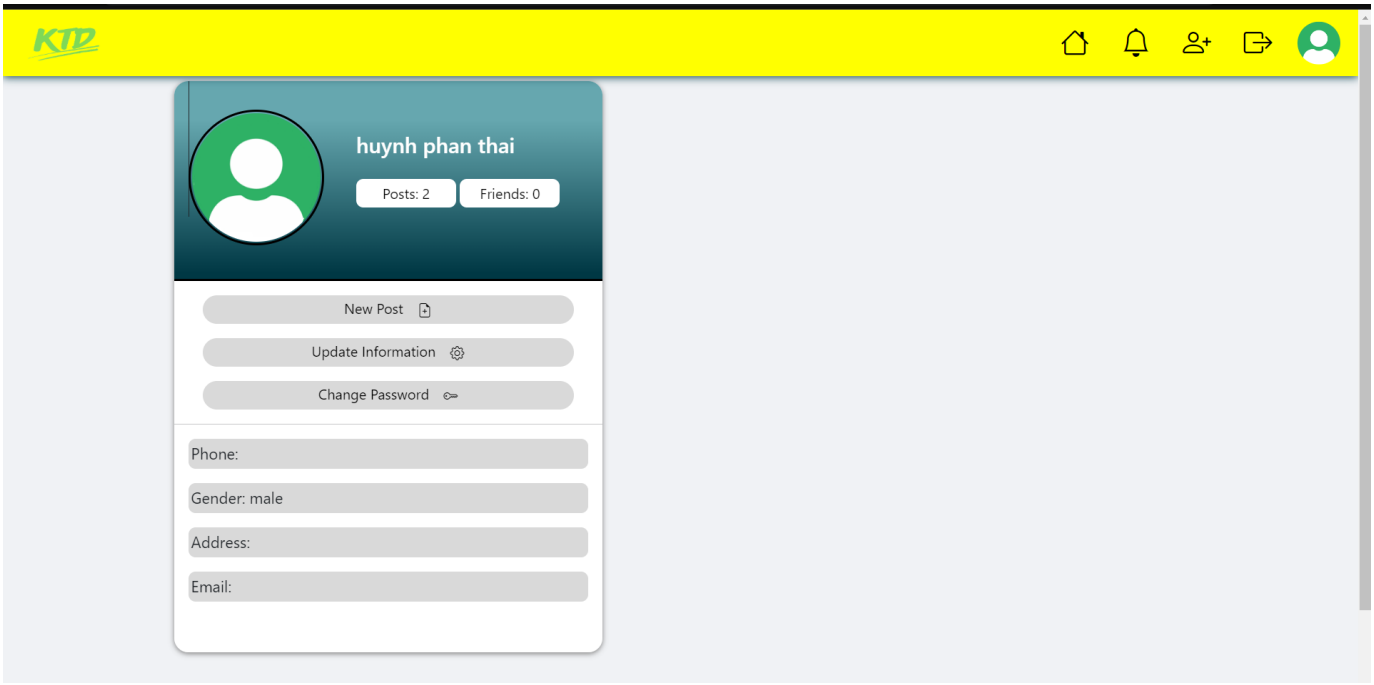


b. Registration interface



c. Home interface

d. Personal interface

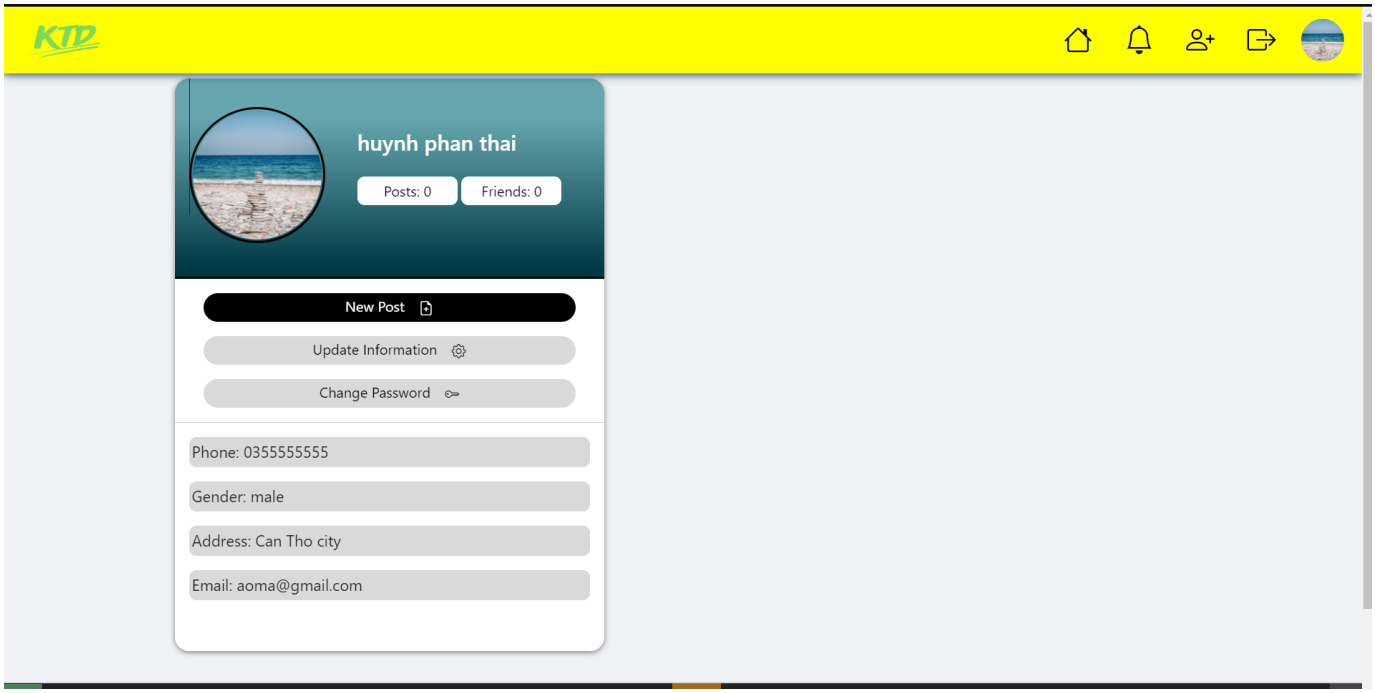


***The process of adding, editing, deleting posts**

User clicks on the avatar. The user will then be redirected to the profile page.

- **Adding**

The user then clicks "New post"



Next, the user adds an image by clicking on the "Add image" box and adding content to the data boxes.

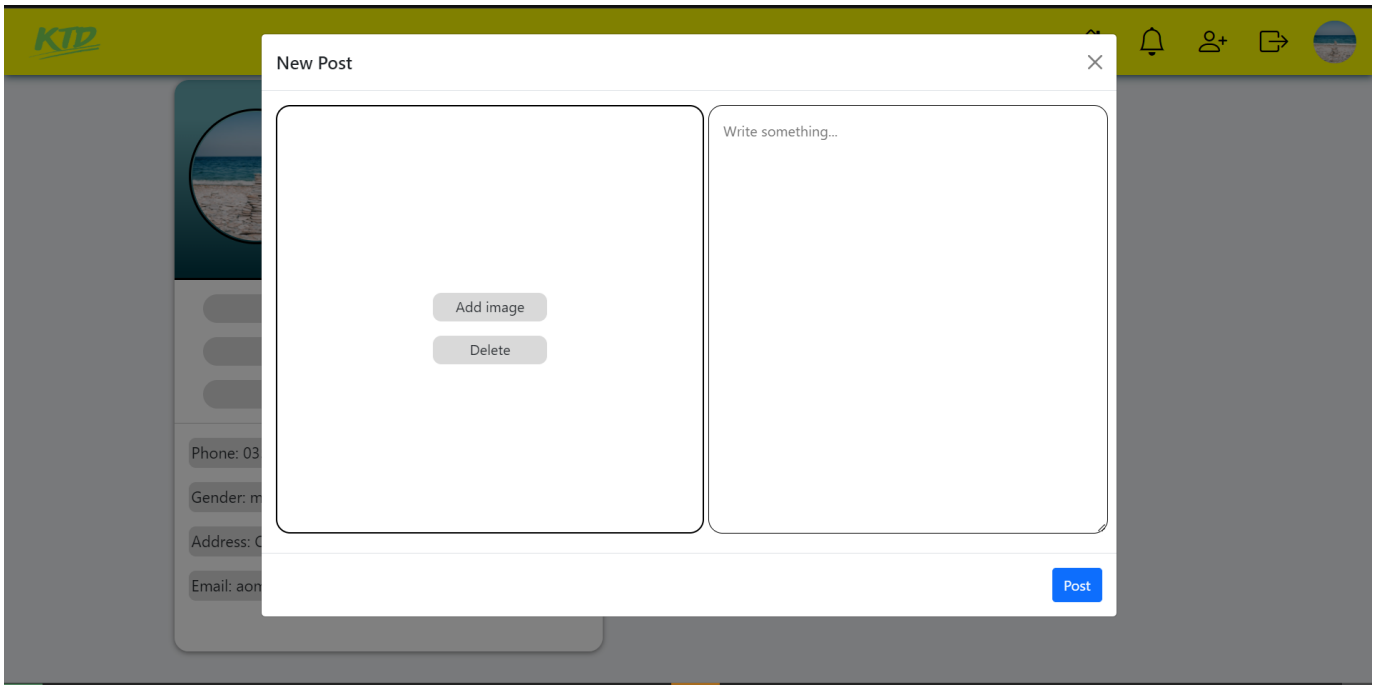


Figure 1 Adding (1)

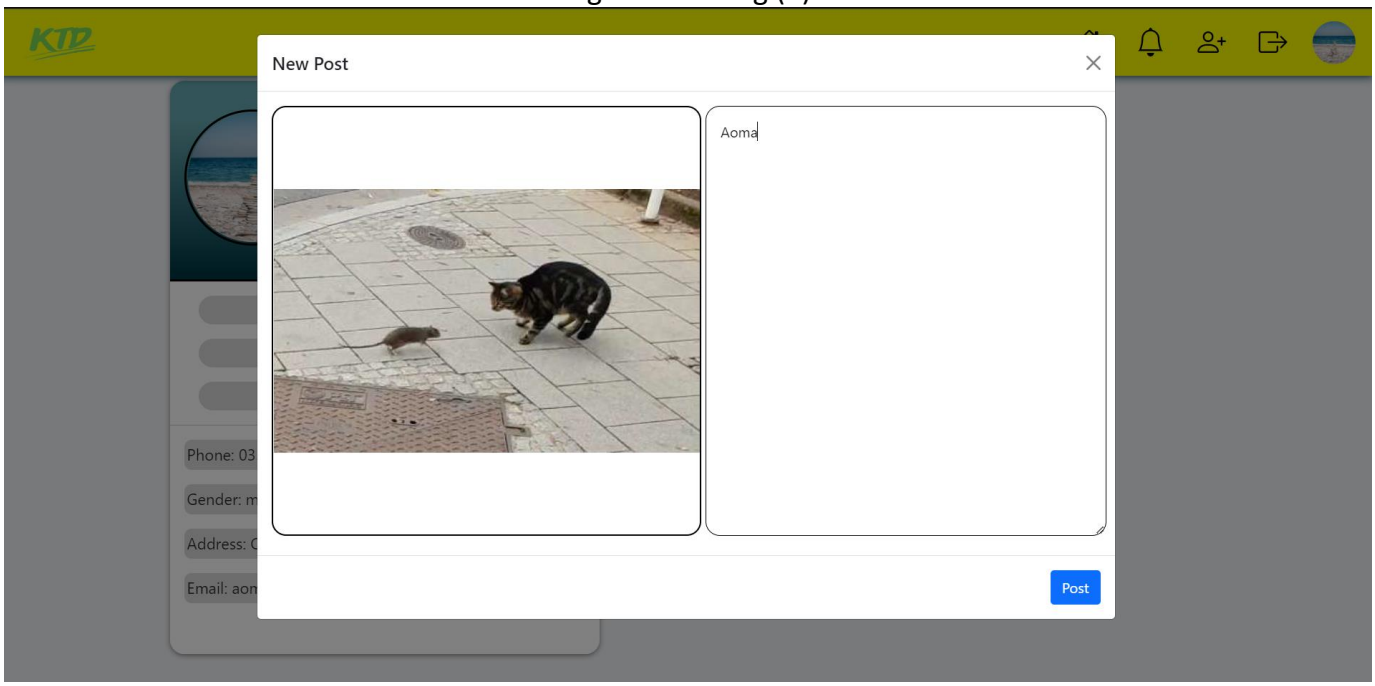


Figure 2 Adding (2)

After adding the data, click on the Post button. The post will be displayed on the profile page

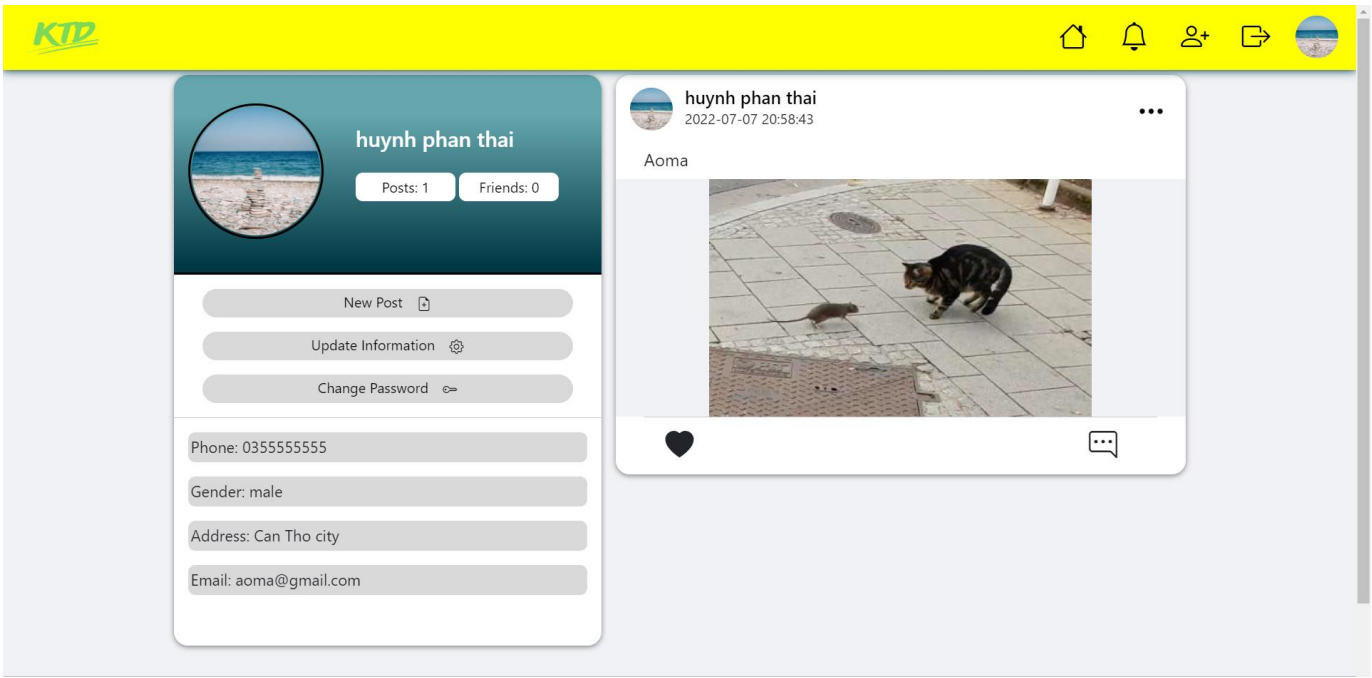


Figure 3 Adding (3)

- **Updating**

The user clicks the "..." sign at the post that the user wants to update. Then click on the edit button

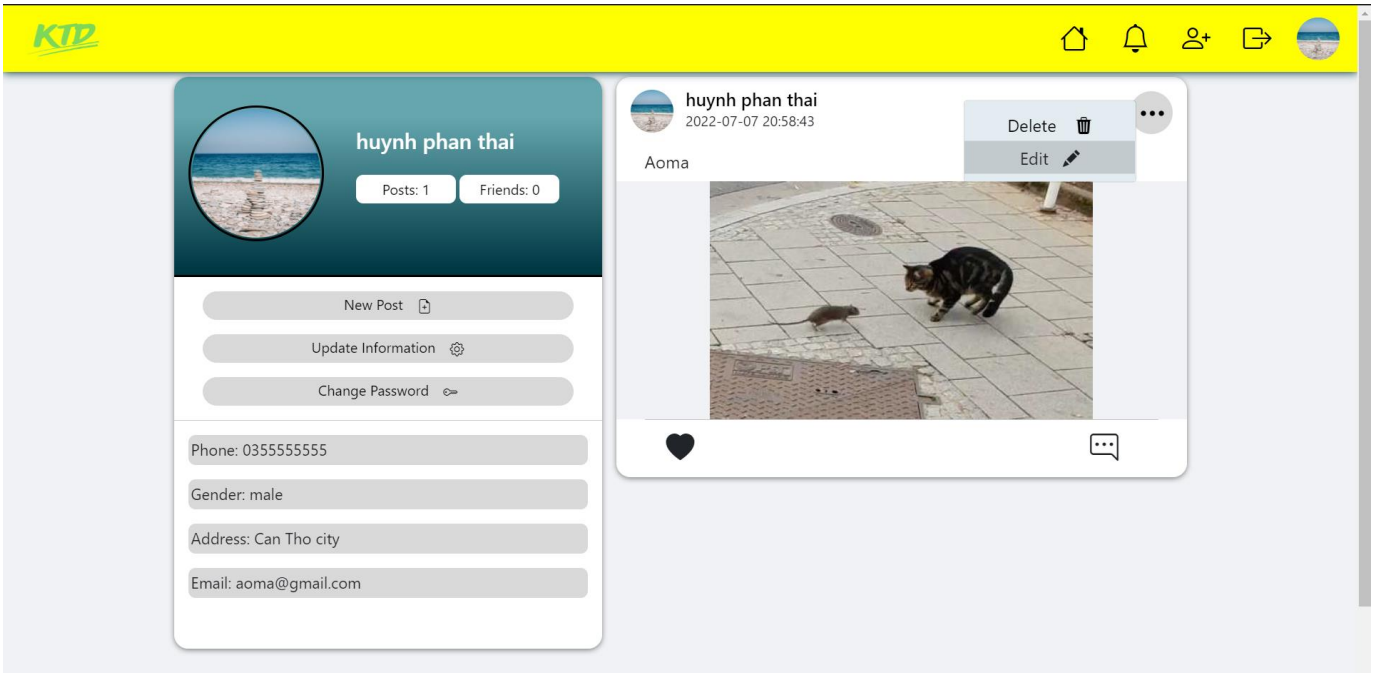


Figure 4 Updating (1)

Then the user enters the information he wants to update and clicks the Update post button. Post information will be updated.

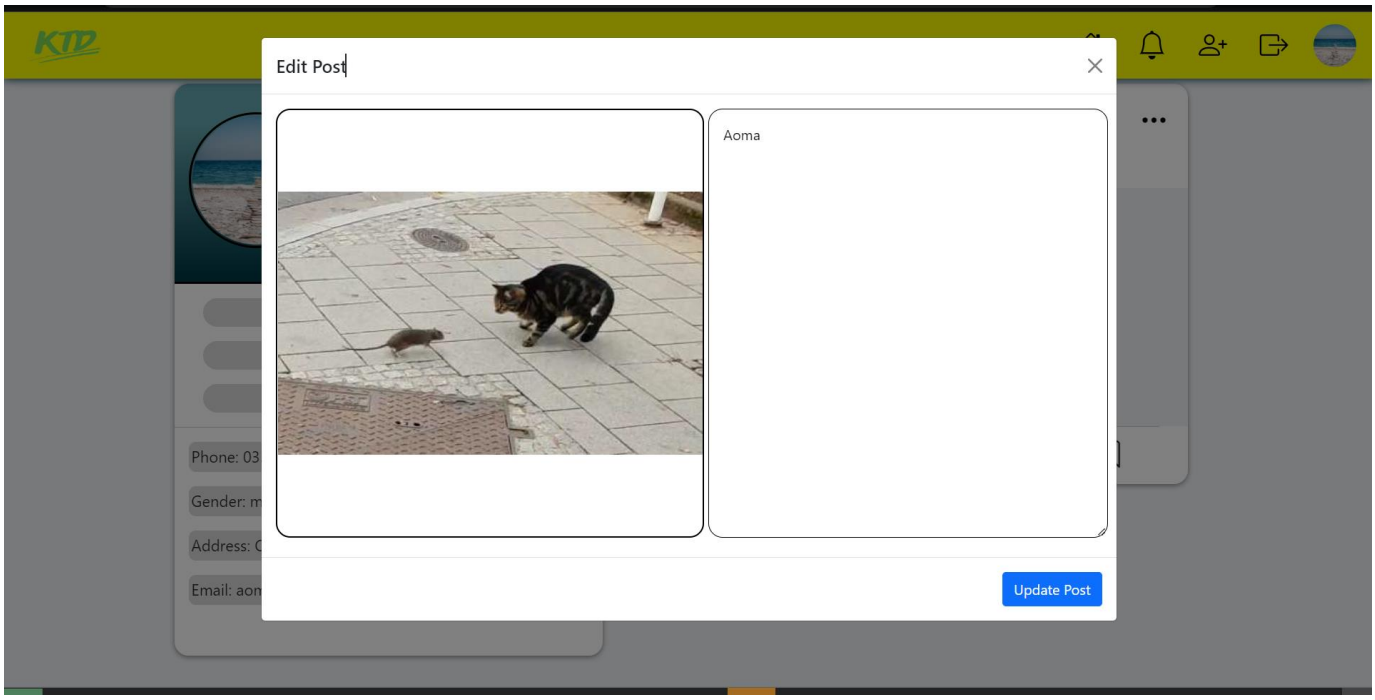


Figure 5 Updating (2)

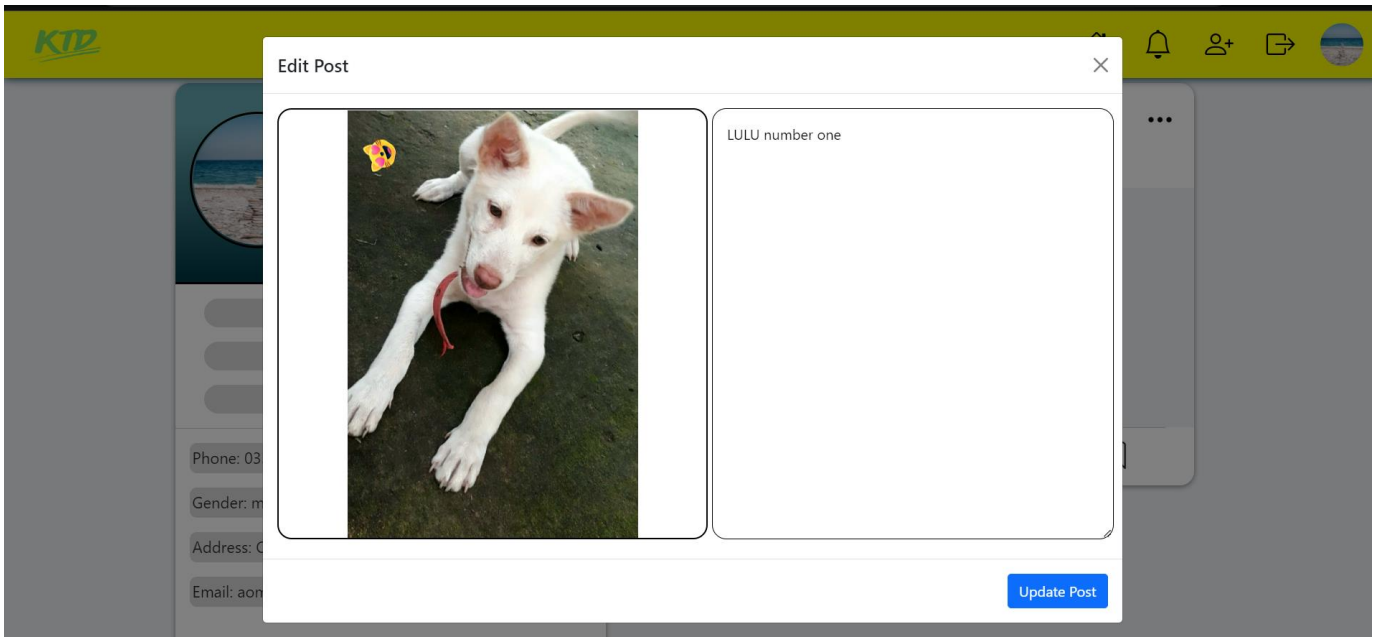


Figure 6 Updating (3)

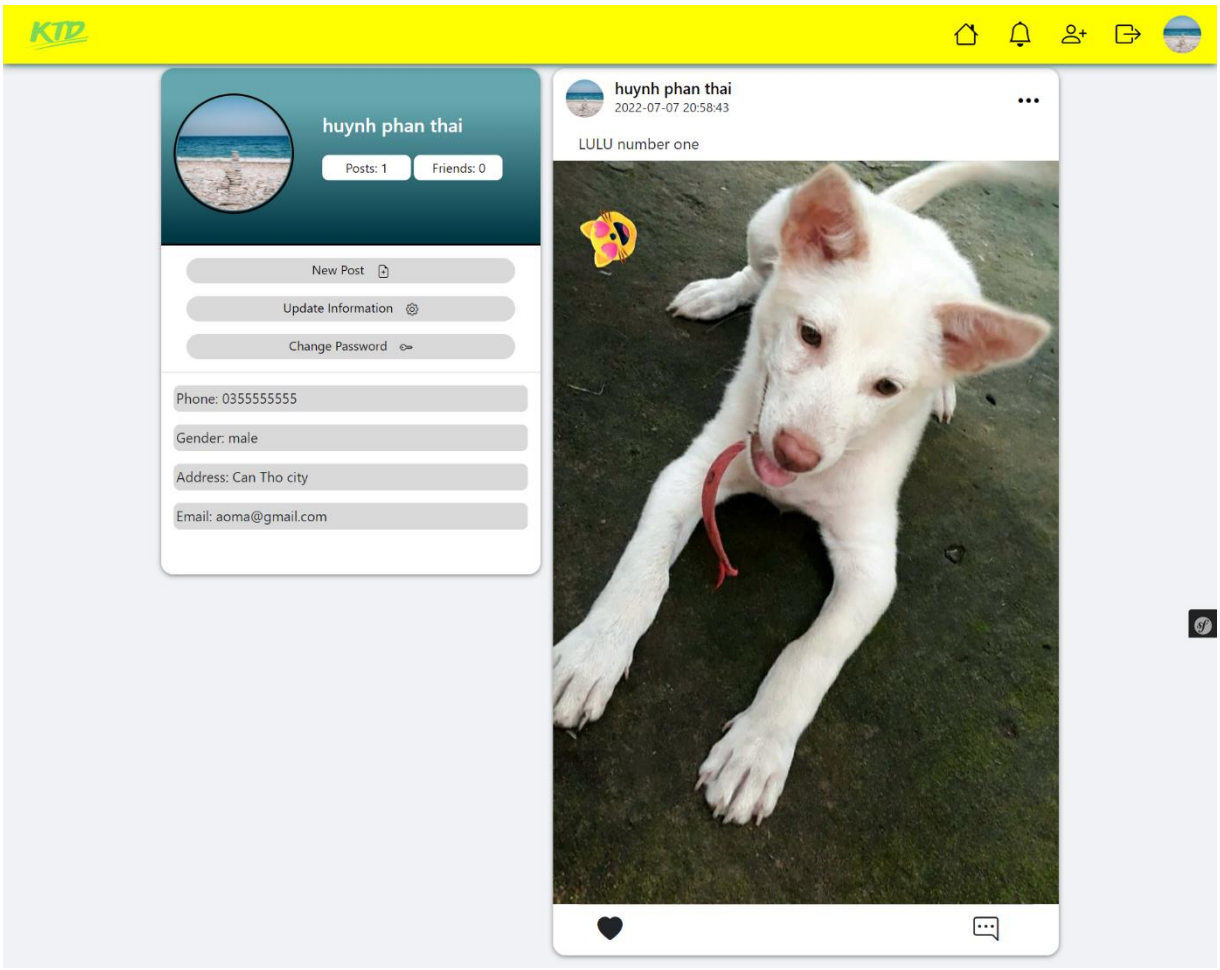


Figure 7 Updating (4)

Deleting

Users click on the "..." button at the post they want to delete and click the Delete button. The post will then be removed from the profile.

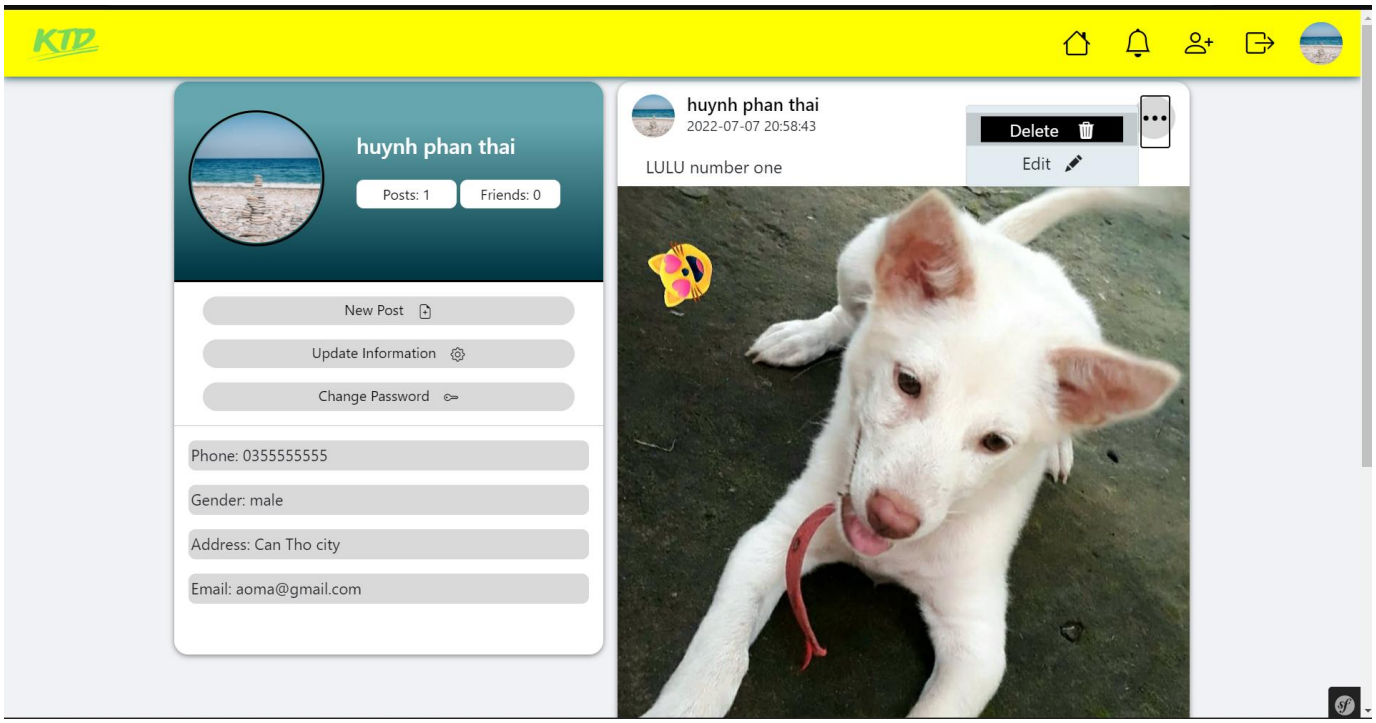


Figure 8 Deleting (1)

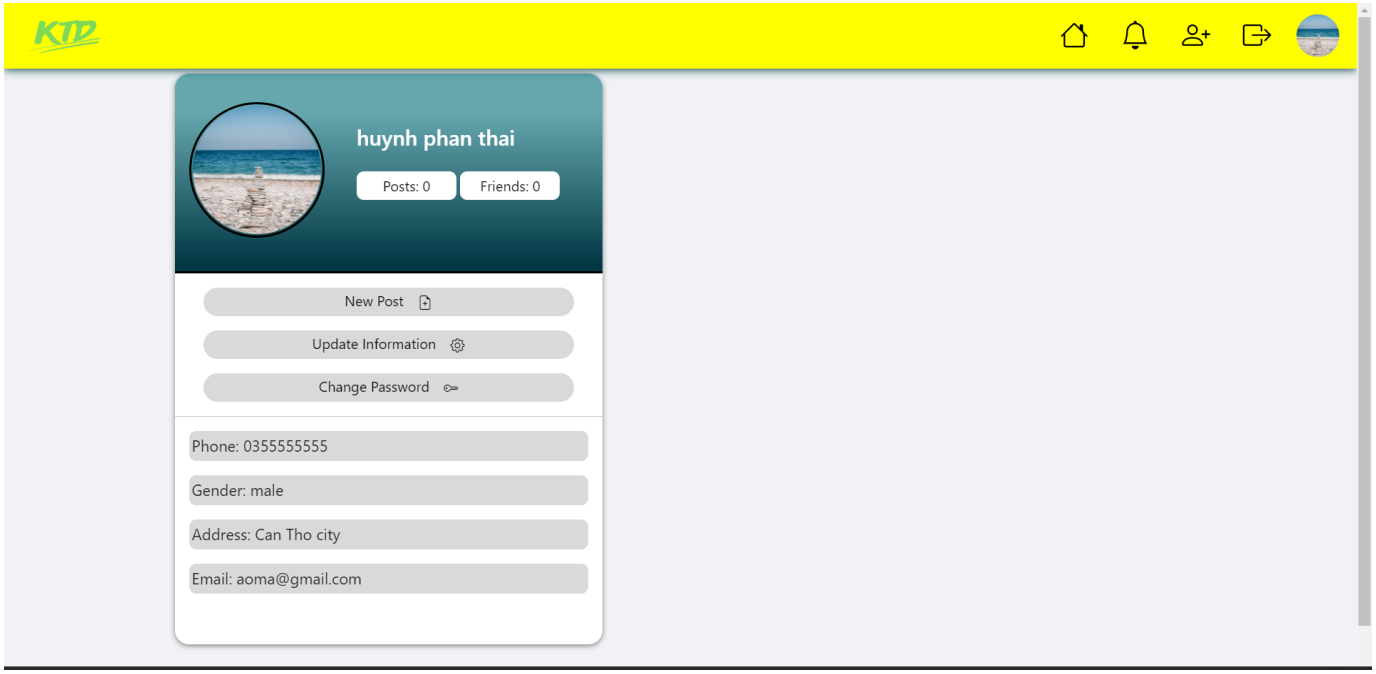


Figure 9 Deleting (2)

3.3 GitHub Repository evidences

https://github.com/blogktddd/blog_ktd.git

Chapter 4 – Conclusion

4.1 What went well

Sau khi hoàn thành dự án. Tôi đã đúc kết được không ít kinh nghiệm. Trong đó có các kinh nghiệm về symfony, git/github, etc.

Dự án của chúng tôi đã kết nối mọi người với nhau. Đáp ứng được nhu cầu thông tin giải trí. Weblog là một thiết kế hoàn hảo đối với mọi người. Từ các chức năng đến giao diện, tốc độ cao và an toàn toàn tuyệt đối.

4.2 What did no go well

While completing the project. There are many mistakes that need to be fixed.

Sometimes we can't do anything for a whole day

There are many errors that arise immediately after fixing an error.

The design is not harmonious, not suitable for a weblog

The team is often late in completing the project

4.3 Lessons learned and further improvements

Learn how symfony works, git/github

Learn how to work as a team: divide tasks from simple to complex

How to persevere with difficulties during project completion

Learn to be creative in designing the look and feel of a weblog

Improvements

More functions will be developed in the future such as messaging, calling, reporting community violation posts. Complex functions like mini game live stream will be updated.

References

flm. (2022, 29 6). *flm*. Retrieved from flm: https://flm.greenwich.edu.vn/gui/role/student/SyllabusDetails?sylID=2515

Githubdoc. (2022, 6 29). *github*. Retrieved from github: https://docs.github.com/en/get-started/using-git/about-git#github-and-the-command-line

netsolutions. (2022, 6 29). *netsolutions*. Retrieved from netsolutions: https://www.netsolutions.com/insights/symfony-framework-features/