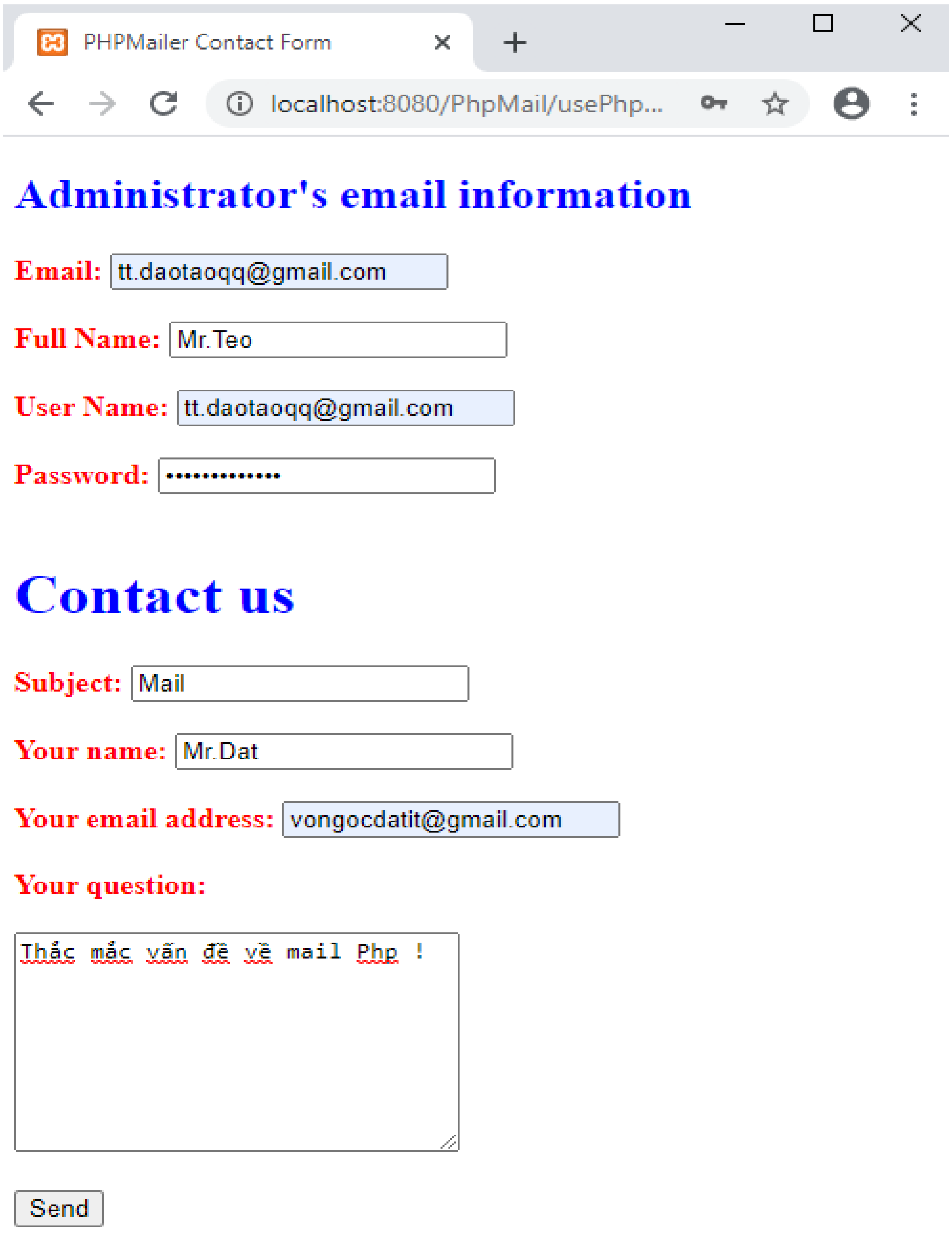
Bài 01. Xây dựng ứng dụng web, sử dụng thư viện **PHPMailer** với giao diện như sau, để gửi mail phản hồi thông tin cho quản trị website.



Administrator's email information

Email:	tt.daotaoqq@gmail.com
Full Name:	Mr.Teo
User Name:	tt.daotaoqq@gmail.com
Password:	••••••••••••••

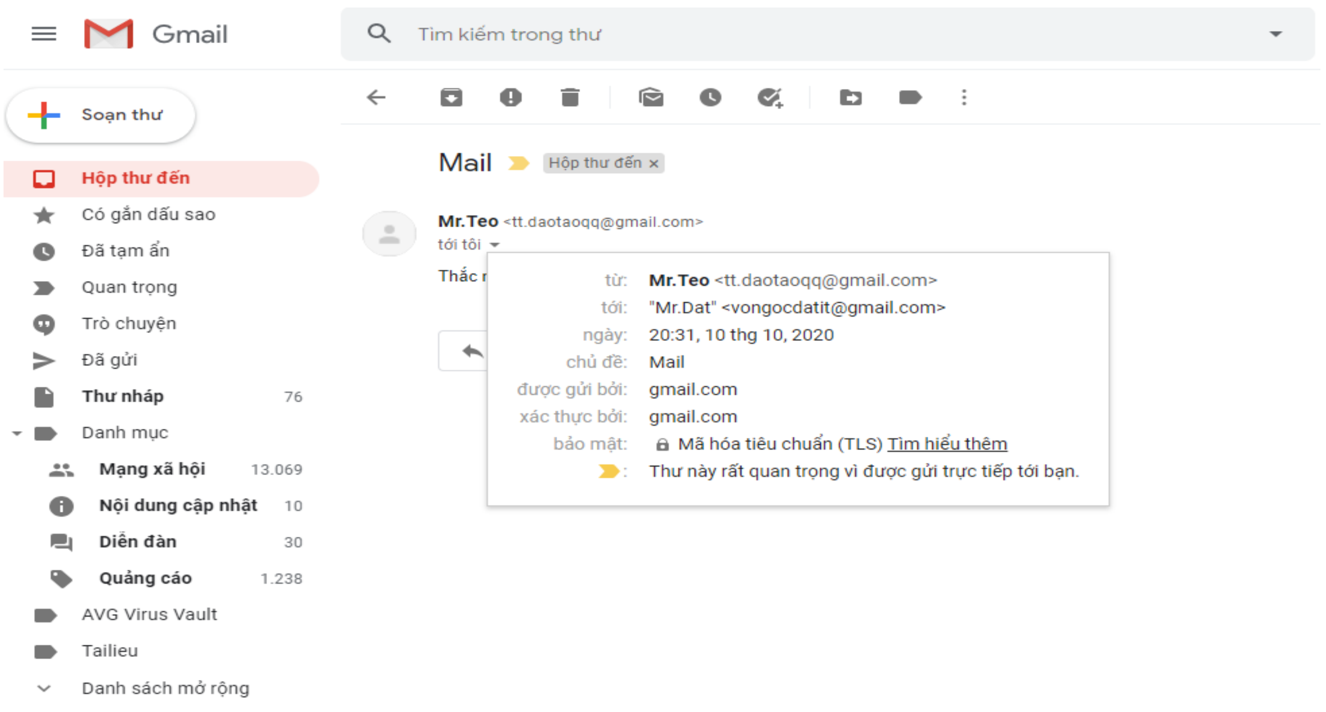
Contact us

Subject:	Mail
Your name:	Mr.Dat
Your email address:	vongocdatit@gmail.com

Your question:

Thắc mắc vấn đề về mail Php !

+ Giao diện nội dung mail do người dùng đã gửi:



từ: Mr.Teo <tt.daotaoqq@gmail.com>
tới: "Mr.Dat" <vongocdatit@gmail.com>
ngày: 20:31, 10 thg 10, 2020
chủ đề: Mail
được gửi bởi: gmail.com
xác thực bởi: gmail.com
bảo mật: Mã hóa tiêu chuẩn (TLS) Tìm hiểu thêm
: Thư này rất quan trọng vì được gửi trực tiếp tới bạn.

<u>Gợi ý:</u>

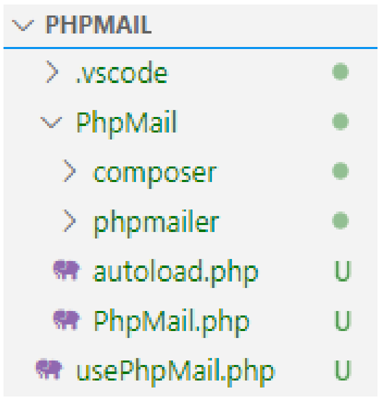
- **B1.** Tải thư viện **PHPMailer**, https://github.com/PHPMailer/PHPMailer

- **B2.** Tạo một tài khoản gmail mới. Sau đó thiết lập cho phép giao thức SMTP Google gửi mail. Settings -> **Forwarding and POP/IMAP** -> In the POP Download section, select the **"Enable POP for all mail"** option -> Save Changes button.

- **B3.** Truy cập vào đường dẫn sau: https://myaccount.google.com/u/0/lesssecureapps?pli=1

=> Thiết lập quyền truy cập tài khoản mail không an toàn (Chú ý: phải thực hiện bước này, nếu không việc gửi mail sẽ bị chặn, do chế độ an toàn của google)

- **B4**. Tạo cấu trúc ứng dụng web như sau:



```
PHPMAIL
  > .vscode
  v PhpMail
    > composer
    > phpmailer
    autoload.php        U
    PhpMail.php         U
  usePhpMail.php        U
```

- **B5.** Tạo file **PhpMail.php**, viết mã định nghĩa **class PhpMail** chứa các phương thức **send_email()** và **valid_email()** sử dụng thư viện **PHPMailer** để gửi mail.

```
<?php
  // Import PHPMailer classes into the global namespace
  // These must be at the top of your script, not inside a function
  use PHPMailer\PHPMailer\PHPMailer;
  use PHPMailer\PHPMailer\SMTP;
  use PHPMailer\PHPMailer\Exception;
  // Load Composer's autoloader
  require 'PhpMail/autoload.php';
  class PhpMail
  {
    function send_email($username, $password, $to_address, $to_name, $from_address,
                   $from_name, $subject, $body, $is_body_html = true, &$msg) {
      if(!$this->valid_email($to_address))
        throw new Exception('This To address is invalid: ' . htmlspecialchars($to_address));
      if(!$this->valid_email($to_address))
        throw new Exception('This From address is invalid: ' . htmlspecialchars($from_address));
      //Server settings
      //Instantiation and passing `true` enables exceptions
      $mail = new PHPMailer(true);
      $mail->isSMTP();  // Send using SMTP
      $mail->Host       = 'smtp.gmail.com'; // Set the SMTP server to send through
      $mail->SMTPAuth   = true;   // Enable SMTP authentication
      $mail->SMTPSecure = 'tls'; // Enable TLS encryption
      $mail->Port       = 587;     // TCP port to connect
      $mail->Username   = $username; // SMTP username
      $mail->Password   = $password; // SMTP password
      //Recipients
      $mail->setFrom($to_address, $to_name);
      $mail->addAddress($from_address, $from_name); // Add a recipient
```

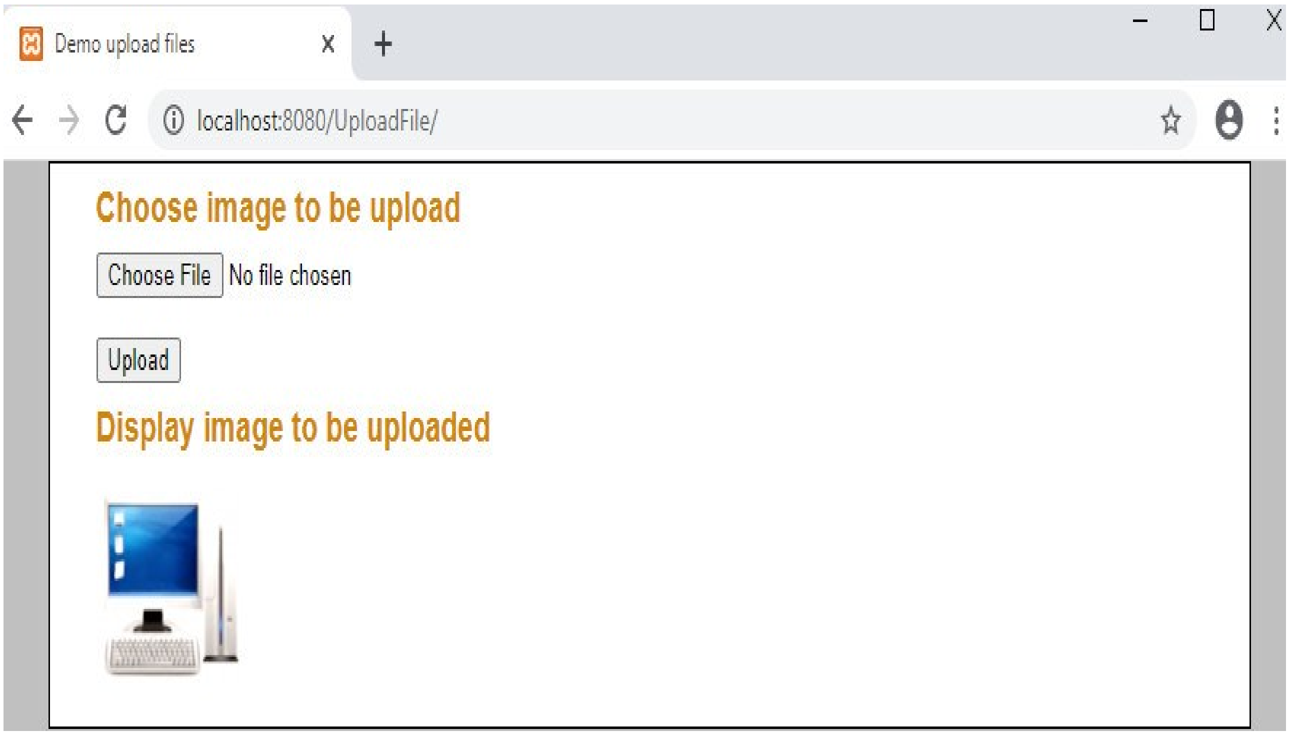
```
        //Content
        $mail->isHTML($is_body_html); // Set email format to HTML
        $mail->Subject = $subject;
        $mail->Body    = $body;
        if (!$mail->send()) {
          $msg .= 'Error sending email: ' . htmlspecialchars($mail->ErrorInfo);
        } else {
          $msg .= 'Message sent!';
        }
      }
      function valid_email($email) {
        if (filter_var($email, FILTER_VALIDATE_EMAIL) === false) {
        return false;
        } else {
        return true;
        }
      }
    }
?>
```

- **B6.** Tạo file **usePhpMail.php** thiết kế giao diện ứng dụng như hình ở phần đầu bài tập và viết mã cho phép gửi mail.

```
<?php
require_once("PhpMail/PhpMail.php");
  $action = filter_input(INPUT_POST,'action',FILTER_DEFAULT);
  if(!empty($action) && $action == "Send")
  {
    $to_email = filter_input(INPUT_POST,'to_email',FILTER_VALIDATE_EMAIL);
    $to_name = filter_input(INPUT_POST,'to_name');
    $username = filter_input(INPUT_POST,'username');
    $password = filter_input(INPUT_POST,'password');
    $subject = filter_input(INPUT_POST,'subject');
    $from_name = filter_input(INPUT_POST,'from_name');
    $from_email = filter_input(INPUT_POST,'from_email',FILTER_VALIDATE_EMAIL);
    $body = filter_input(INPUT_POST,'body');
```

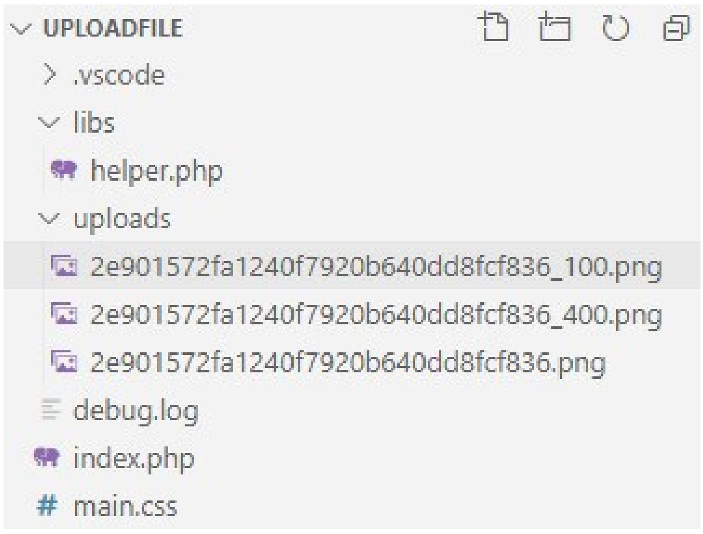
```
        $mail = new PHPMail();
        if(!empty($subject) && !empty($from_name) && !empty($from_email) && !empty($body)
          && !empty($to_email) && !empty($to_name) && !empty($username) && !empty($password))
        {
           $msg = "";
           $mail-
>send_email($username,$password,$to_email,$to_name,$from_email,$from_name,$subject,$body,true,$msg);
        }
    }
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>PHPMailer Contact Form</title>
    <style>
        h1,h2{color:blue}
        label{color:red;font-weight: bold;}
    </style>
</head>
<body>
<h2>Administrator's email information</h2>
<form action="" method="post">
   <label for="to_email">Email: <input type="text" name="to_email" id="to_email" value="<?php echo !empty
($to_email)?$to_email:""; ?>"></label><br><br>
   <label for="to_name">Full Name: <input type="text" name="to_name" id="to_name" value="<?php echo !e
mpty($to_name)?$to_name:""; ?>"></label><br><br>
   <label for="username">User Name: <input type="text" name="username" id="username" value="<?php echo
!empty($username)?$username:""; ?>"></label><br><br>
   <label for="password">Password: <input type="password" name="password" id="password" value="<?php e
cho !empty($password)?$password:""; ?>"></label><br><br>
<h1>Contact us</h1>
    <label for="subject">Subject: <input type="text" name="subject" id="subject"></label><br><br>
    <label for="name">Your name: <input type="text" name="from_name" id="from_name"></label><br><br>
    <label for="email">Your email address: <input type="from_email" name="from_email" id="from_email"></
label><br><br>
    <label for="body">Your question:</label><br><br>
    <textarea cols="30" rows="8" name="body" id="body" placeholder="Your question"></textarea><br><br>
    <input type="submit" name="action" value="Send">
</form>
<?php
  if(!empty($msg))
    echo "<h4>$msg</h4>";
?>
</body>
</html>
```

Bài 02. Xây dựng ứng dụng web. Cho phép người dùng chọn ảnh và upload file lên hệ thống ứng dụng web, với giao diện ứng dụng web như sau:


Demo upload files
localhost:8080/UploadFile/
Choose image to be upload
Choose File No file chosen
Upload
Display image to be uploaded

Gợi ý:

- **B1.** Tạo cấu trúc ứng dụng như sau:


```
UPLOADFILE
  > .vscode
  v libs
      helper.php
  v uploads
      2e901572fa1240f7920b640dd8fcf836_100.png
      2e901572fa1240f7920b640dd8fcf836_400.png
      2e901572fa1240f7920b640dd8fcf836.png
    debug.log
    index.php
    main.css
```

- **B2.** Viết code cho thư viên **libs\helper.php** để upload ảnh.

```php
<?php
class Helper
{
   public static function process_image($dir, $filename) {
      // Set up the variables
      $dir = $dir . DIRECTORY_SEPARATOR;
      $i = strrpos($filename, '.');
      $image_name = substr($filename, 0, $i);
      $ext = substr($filename, $i);
      // Set up the read path
      $image_path = $dir . DIRECTORY_SEPARATOR . $filename;
      // Set up the write paths
      $image_path_400 = $dir . $image_name . '_400' . $ext;
      $image_path_100 = $dir . $image_name . '_100' . $ext;
      // Create an image that's a maximum of 400x300 pixels
      self::resize_image($image_path, $image_path_400, 400, 300);
      // Create a thumbnail image that's a maximum of 100x100 pixels
      self::resize_image($image_path, $image_path_100, 100, 100);
   }
   /*******************************************
    * Resize image to 400x300 max
```

```
 *********************************************/
public static function resize_image($old_image_path, $new_image_path,
        $max_width, $max_height) {
    // Get image type
    $image_info = getimagesize($old_image_path);
    $image_type = $image_info[2];
    // Set up the function names
    switch($image_type) {
        case IMAGETYPE_JPEG:
            $image_from_file = 'imagecreatefromjpeg';
            $image_to_file = 'imagejpeg';
            break;
        case IMAGETYPE_GIF:
            $image_from_file = 'imagecreatefromgif';
            $image_to_file = 'imagegif';
            break;
        case IMAGETYPE_PNG:
            $image_from_file = 'imagecreatefrompng';
            $image_to_file = 'imagepng';
            break;
        default:
            echo 'File must be a JPEG, GIF, or PNG image.';
            exit;
    }
    // Get the old image and its height and width
    $old_image = $image_from_file($old_image_path);
    $old_width = imagesx($old_image);
    $old_height = imagesy($old_image);
    // Calculate height and width ratios
    $width_ratio = $old_width / $max_width;
    $height_ratio = $old_height / $max_height;
    // If image is larger than specified ratio, create the new image
    if ($width_ratio > 1 || $height_ratio > 1) {
        // Calculate height and width for the new image
        $ratio = max($width_ratio, $height_ratio);
        $new_height = round($old_height / $ratio);
        $new_width = round($old_width / $ratio);
        // Create the new image
        $new_image = imagecreatetruecolor($new_width, $new_height);
        // Set transparency according to image type
        if ($image_type == IMAGETYPE_GIF) {
          $alpha = imagecolorallocatealpha($new_image, 0, 0, 0, 127);
          imagecolortransparent($new_image, $alpha);
        }
        if ($image_type == IMAGETYPE_PNG || $image_type == IMAGETYPE_GIF) {
          imagealphablending($new_image, false);
          imagesavealpha($new_image, true);
        }
        // Copy old image to new image - this resizes the image
        $new_x = 0;
        $new_y = 0;
        $old_x = 0;
        $old_y = 0;
        imagecopyresampled($new_image, $old_image,
                    $new_x, $new_y, $old_x, $old_y,
                    $new_width, $new_height, $old_width, $old_height);
        // Write the new image to a new file
        $image_to_file($new_image, $new_image_path);
        // Free any memory associated with the new image
        imagedestroy($new_image);
      } else {
        // Write the old image to a new file
        $image_to_file($old_image, $new_image_path);
      }
      // Free any memory associated with the old image
      imagedestroy($old_image);
    }
```

```
  public static function upload_file($inputfile, &$imgfile)
  {
    if (!empty($_FILES[$inputfile])) {
      $check = getimagesize($_FILES[$inputfile]['tmp_name']);
      // Check if image file is a actual image or fake image and error upload file
      if (($_FILES[$inputfile]['error'] > 0) && $check === false) {
        return false;
      } else {
        $clientpath = $_FILES[$inputfile]['tmp_name'];
        $imgfile = $_FILES[$inputfile]['name'];
        $extension = "." . strtolower(pathinfo($imgfile,PATHINFO_EXTENSION));
        $allowed_extensions = array(".jpg", ".jpeg", ".png", ".gif");
        $imgnewfile = md5($imgfile) . $extension;
        if (in_array($extension, $allowed_extensions)) {
          move_uploaded_file($clientpath, 'uploads/' . $imgnewfile);
          //image size
          self::process_image('uploads/',$imgnewfile);
          $imgfile = $imgnewfile;
          return true;
        } else {
          echo "<script>alert('Phần mở rộng file không hợp lệ. Chỉ phần mở rộng jpg /jpeg/png/gif cho phép
upload file !');</script>";
          return false;
        }
      }
    } else
      return false;
  }
}
```

- **B3.** Viết code **index.php** để thực hiện chức năng upload file.

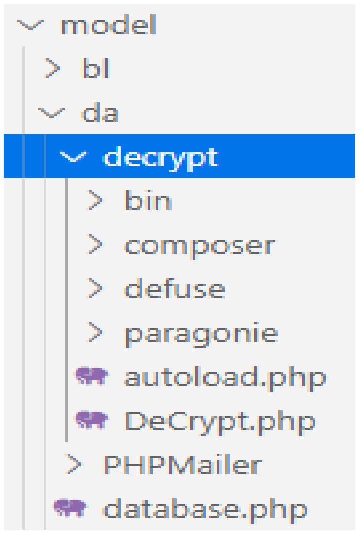
```
<?php
  require_once("libs/helper.php");
 //Cach 1. Upload file su dung phuong thuc move_uploaded_file()
 $action = filter_input(INPUT_POST,'action');
 if(!empty($action))
 {
   $img_file = "";
   Helper::upload_file('file1',$img_file);
   //Retrieve Image size is 400x400
   $i = strrpos($img_file, '.');
   $image_name = substr($img_file, 0, $i);
   $ext = substr($img_file, $i);
   $img_file = $image_name . '_100' . $ext;
 }
?>
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Demo upload files</title>
  <link rel="stylesheet" href="main.css">
</head>
<body>
    <h2>Choose image to be upload</h2>
    <form id="upload_form" action="." method="POST" enctype="multipart/form-data">
      <input type="hidden" name="action" value="upload">
      <input type="file" name="file1"><br><br>
      <input id="upload_button" type="submit" value="Upload">
    </form>

    <h2>Display image to be uploaded</h2>
    <p>
    <img src="uploads/<?php echo $img_file; ?>" alt="Image01">
    </p>
</body>
</html>
```

- **B4.** Nội dung file **main.css**

```
html {
    background-color: rgb(192, 192, 192);
}
body {
    font-family: Arial, Helvetica, sans-serif;
    width: 760px;
    margin: 0 auto;
    padding: 0 2em;
    background-color: white;
    border: 1px solid black;
}
header {
    border-bottom: 2px solid black;
    padding: .5em 0;
}
header h1 {
    color: black;
}
h1 {
    font-size: 150%;
    margin: .5em 0 .25em;
}
h2 {
    font-size: 120%;
    margin: .5em 0;
}
h1, h2 {
    color: rgb(208, 133, 4);
}
label {
    width: 8em;
    padding-right: .5em;
    padding-top: .25em;
    padding-bottom: .5em;
    text-align: right;
    float: left;
}
br {
    clear: both;
}
```

Bài 03. Cách sử dụng thư viện **Defuse** để mã hóa dữ liệu hai chiều và thư viện **PHPMailer** vào project.

- B1. Sao chép thư viện **Defuse** (trong thư mục ví dụ **Encryption**) và thư mục **PHPMailer** (trong thư mục ví dụ **ch22_register_email**) vào trong thư mục **model\da.**



```
model
  bl
  da
    decrypt
      bin
      composer
      defuse
      paragonie
      autoload.php
      DeCrypt.php
    PHPMailer
    database.php
```

- B2. Mở file **DeCrypt.php** cập nhập lại mã như sau:

```
<?php
   require_once("autoload.php");
   use Defuse\Crypto\Crypto;
   use Defuse\Crypto\Key;
   use Defuse\Crypto\Exception\WrongKeyOrModifiedCiphertextException;
  class DeCrypt
  {
   private static $key;
   function __construct() {
       self::$key = Key::createNewRandomKey();
   }
   public static function encrypt($data) {
      $encrypted_data = Crypto::encrypt($data, self::$key);
      return $encrypted_data;
   }
   public static function decrypt($encrypted_data) {
     try {
      $data = Crypto::decrypt($encrypted_data, self::$key);
      return $data;
     } catch (WrongKeyOrModifiedCiphertextException $ex) {
        throw new Exception($ex->getMessage());
     }
   }
  }
?>
```

- **B3.** Mở file **da\helper.php** viết một số đoạn mã bổ sung thêm một số thuộc tính và phương thức để sử dụng thư viện **Defuse** và **PHPMailer.**

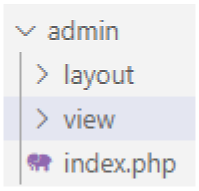
```
<?php
include_once('decrypt/DeCrypt.php');
include_once('PHPMailer/PHPMailerAutoload.php');
class Helper extends DeCrypt
{
   function __construct() {
     parent::__construct();
   }
   ……
public static function process_image($dir, $filename) {
      // Set up the variables
      $dir = $dir . DIRECTORY_SEPARATOR;
      $i = strrpos($filename, '.');
      $image_name = substr($filename, 0, $i);
      $ext = substr($filename, $i);
      // Set up the read path
      $image_path = $dir . DIRECTORY_SEPARATOR . $filename;
```

```
        // Set up the write paths
        $image_path_400 = $dir . $image_name . '_400' . $ext;
        $image_path_100 = $dir . $image_name . '_100' . $ext;
        // Create an image that's a maximum of 400x300 pixels
        self::resize_image($image_path, $image_path_400, 400, 300);
        // Create a thumbnail image that's a maximum of 100x100 pixels
        self::resize_image($image_path, $image_path_100, 100, 100);
    }
    /*******************************************
     * Resize image to 400x300 max
     *******************************************/
    public static function resize_image($old_image_path, $new_image_path,
            $max_width, $max_height) {
        // Get image type
        $image_info = getimagesize($old_image_path);
        $image_type = $image_info[2];
        // Set up the function names
        switch($image_type) {
            case IMAGETYPE_JPEG:
                $image_from_file = 'imagecreatefromjpeg';
                $image_to_file = 'imagejpeg';
                break;
            case IMAGETYPE_GIF:
                $image_from_file = 'imagecreatefromgif';
                $image_to_file = 'imagegif';
                break;
            case IMAGETYPE_PNG:
                $image_from_file = 'imagecreatefrompng';
                $image_to_file = 'imagepng';
                break;
            default:
                echo 'File must be a JPEG, GIF, or PNG image.';
                exit;
        }
```

```
// Get the old image and its height and width
$old_image = $image_from_file($old_image_path);
$old_width = imagesx($old_image);
$old_height = imagesy($old_image);
// Calculate height and width ratios
$width_ratio = $old_width / $max_width;
$height_ratio = $old_height / $max_height;
// If image is larger than specified ratio, create the new image
if ($width_ratio > 1 || $height_ratio > 1) {
    // Calculate height and width for the new image
    $ratio = max($width_ratio, $height_ratio);
    $new_height = round($old_height / $ratio);
    $new_width = round($old_width / $ratio);
    // Create the new image
    $new_image = imagecreatetruecolor($new_width, $new_height);
    // Set transparency according to image type
    if ($image_type == IMAGETYPE_GIF) {
        $alpha = imagecolorallocatealpha($new_image, 0, 0, 0, 127);
        imagecolortransparent($new_image, $alpha);
    }
    if ($image_type == IMAGETYPE_PNG || $image_type == IMAGETYPE_GIF) {
        imagealphablending($new_image, false);
        imagesavealpha($new_image, true);
    }
    // Copy old image to new image - this resizes the image
    $new_x = 0;
    $new_y = 0;
    $old_x = 0;
    $old_y = 0;
    imagecopyresampled($new_image, $old_image,
                    $new_x, $new_y, $old_x, $old_y,
                    $new_width, $new_height, $old_width, $old_height);
```

```
            // Write the new image to a new file
            $image_to_file($new_image, $new_image_path);
            // Free any memory associated with the new image
            imagedestroy($new_image);
        } else {
            // Write the old image to a new file
            $image_to_file($old_image, $new_image_path);
        }
        // Free any memory associated with the old image
        imagedestroy($old_image);
    }
    //Upload files
    public static function upload_file($inputfile, &$imgfile)
    {
        if (!empty($_FILES[$inputfile])) {
            $check = getimagesize($_FILES[$inputfile]['tmp_name']);
            // Check if image file is a actual image or fake image and error upload file
            if (($_FILES[$inputfile]['error'] > 0) && $check === false) {
                return false;
            } else {
                $clientpath = $_FILES[$inputfile]['tmp_name'];
                $imgfile = $_FILES[$inputfile]['name'];
                $extension = "." . strtolower(pathinfo($imgfile,PATHINFO_EXTENSION));
                $allowed_extensions = array(".jpg", ".jpeg", ".png", ".gif");
                $imgnewfile = md5($imgfile) . $extension;

                if (in_array($extension, $allowed_extensions)) {
                    move_uploaded_file($clientpath, 'uploads/' . $imgnewfile);
                    //image size
                    self::process_image('uploads/',$imgnewfile);
                    $imgfile = $imgnewfile;
                    return true;
                } else {
                    echo "<script>alert('Phần mở rộng file không hợp lệ. Chỉ phần mở rộng jpg /jpeg
/png/gif cho phép upload file !');</script>";
                    return false;
                }
            }
        } else
            return false;
    }
    //Send mail
    public static function send_email($to_address, $to_name, $from_address, $from_name,
                                      $subject, $body, $is_body_html = false) {
        if (!self::valid_email($to_address)) {
            throw new Exception('This To address is invalid: ' .
                                htmlspecialchars($to_address));
        }
        if (!self::valid_email($from_address)) {
            throw new Exception('This From address is invalid: ' .
                                htmlspecialchars($from_address));
        }
        $mail = new PHPMailer();
        // **** You must change the following to match your
        // **** SMTP server and account information.
        $mail->isSMTP();                          // Set mailer to use SMTP
        $mail->Host = 'smtp.gmail.com';           // Set SMTP server
        $mail->SMTPSecure = 'tls';                // Set encryption type
        $mail->Port = 587;                        // Set TCP port
        $mail->SMTPAuth = true;                   // Enable SMTP authentication
        $mail->Username = 'tt.daotaoqq@gmail.com'; // Set SMTP username
        $mail->Password = '';           // Set SMTP password
        // Set From address, To address, subject, and body
        $mail->setFrom($from_address, $from_name);
        $mail->addAddress($to_address, $to_name);
        $mail->Subject = $subject;
        $mail->Body = $body;                 // Body with HTML
        $mail->AltBody = strip_tags($body);   // Body without HTML
        if ($is_body_html) {
            $mail->isHTML(true);             // Enable HTML
        }
        if(!$mail->send()) {
            throw new Exception('Error sending email: ' .
                            htmlspecialchars($mail->ErrorInfo) );
        }
        else
            return true;
    }
    public static function valid_email($email) {
        if (filter_var($email, FILTER_VALIDATE_EMAIL) === false) {
            return false;
        } else {
            return true;
        }
    }
}
```

- B4. Mở file **controller : admin\index.php** thêm đoạn mã khởi tạo đối tượng thể hiện lớp **Helper**, để gọi phương thức **__construct()** tạo **Key** sử dụng cho thư viện mã hóa **Defuse.**



```
<?php
    .........
    $db = new Database();
    //Create key for encryption
    $h = new Helper();
?>
```

- **B5.** Viết code để test ứng dụng.

```
<?php
  print_r(Helper::encrypt('abc')) . "<hr>";
  print_r(Helper::decrypt(Helper::encrypt('abc')));
  $kq = Helper::send_email('vongocdatit@gmail.com','Vo Ngoc Dat','tt.dtqq@gmail.com','lap
top.com','Test Mail','Ban co nhan duoc mail khong',true);
  if($kq)
    echo "Gui mail thanh cong !";
?>
```

=> **Lưu ý:** để có thể gửi mail, phải khai báo thông tin **xác thực mail** cho phương thức **send_email()** của lớp **Helper**.

```
public static function send_email
{
……
$mail = new PHPMailer();
        // **** You must change the following to match your
        // **** SMTP server and account information.
        $mail->isSMTP();                          // Set mailer to use SMTP
        $mail->Host = 'smtp.gmail.com';           // Set SMTP server
        $mail->SMTPSecure = 'tls';                // Set encryption type
        $mail->Port = 587;                        // Set TCP port
        $mail->SMTPAuth = true;                   // Enable SMTP authentication
        $mail->Username = 'tt.daotaoqq@gmail.com'; // Set SMTP username
        $mail->Password = '';          // Set SMTP password
……
}
```