Unit –IV Creating and validating forms	4a Use the relevant form controls to get user's input. 4b Design web pages using multiple Forms for the given problem. 4c Apply the given validation rules on form. 4d Set/ modify/ delete cookies using cookies attributes. 4e Manage the given session using session variables.	4.1 Creating a webpage using GUI Components, Browser Role-GET and POST methods, Server Role 4.2 Form controls: text box, text area, radio button, check box, list, buttons 4.3 Working with multiple forms : - A web page having many forms - A form having multiple submit buttons. 4.4 Web page validation. 4.5 Cookies - Use of cookies, Attributes of cookies, create cookies, modify cookies value, and delete cookies. 4.6 Session - Use of session, Start session, get session variables, destroy session. 4.7 Sending E-mail.

12 marks

Introduction to Webpage

To run any code on server or to get connected with PHP on server we need to set up Web pages in a specific fashion.

Two methods "get" and "post" mentioned in form are commonly used to send data from HTML controls to PHP scripts on the server.

URL is used to specify location, which helps browsers understand where to send the data on the server mentioned in the "action: attribute of a form".

Two methods get and post mentioned in form are commonly used to send data from HTML controls to PHP script on server.

URL is used to specify the location which specifies where to send data on the server.

PHP script handles both displaying of HTML controls and then reading of data from HTML controls when the user clicks on the submit button.

4.1 Creating a Webpage using GUI components

A document that contains blank fields, that the user can fill the data or user can select the data is known as form.Generally the data will be stored in the database.We can create and use forms in PHP. To get form data, we need to use PHP superglobals. $_GET and $_POST.

The form request may be get or post. To retrieve data from get request we need to use $_GET, for post request $_POST

Browser Role - GET and POST Methods

Browser used one of the two HTTP methods - GET and POST to communicate with the server. Both methods are used to pass the information differently to the server.

GET method:

1. The GET method sends the encoded user information appended to the page request (to the url). The code and the encoded information are separated by the ? character.
2. http://www.test.com/index.htm?name1=value1&name2=value2
3. The GET method produces a long string that appears in our server logs, in the browser's location:box.
4. Never use the GET method if we have a password or other sensitive information to be sent to the server. Get cannot be used to send binary data, like images or word documents, to the server.
5. The data sent by the GET method can be accessed using the QUESRY_STRING environment variable.
6. PHP provides a $_GET associative array to access all the sent information using the GET method.

POST Method

1. The POST method transfers information via HTTP headers. The information is encoded as described in the case of the GET method and put into a header called QUERY_STRING.
2. The POST method does not have any restriction on data size to be sent.
3. The POST method can be used to send ASCII as well as binary data.
4. The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP you can make sure that your information is secure.

The PHP provides a $_POST associative array to access all the sent information using the POST method.

Difference between get and post

GET Method

1. Information sent from a form with the get method is visible to everyone(all variable names and values are displayed in the URL)
2. GET has limits on the amount of information to send. The limitation is about 2048 characters
3. $_GET is an array of variables passed to the current script via the URL parameters.
4. Can be bookmarked.
5. Can be cached.

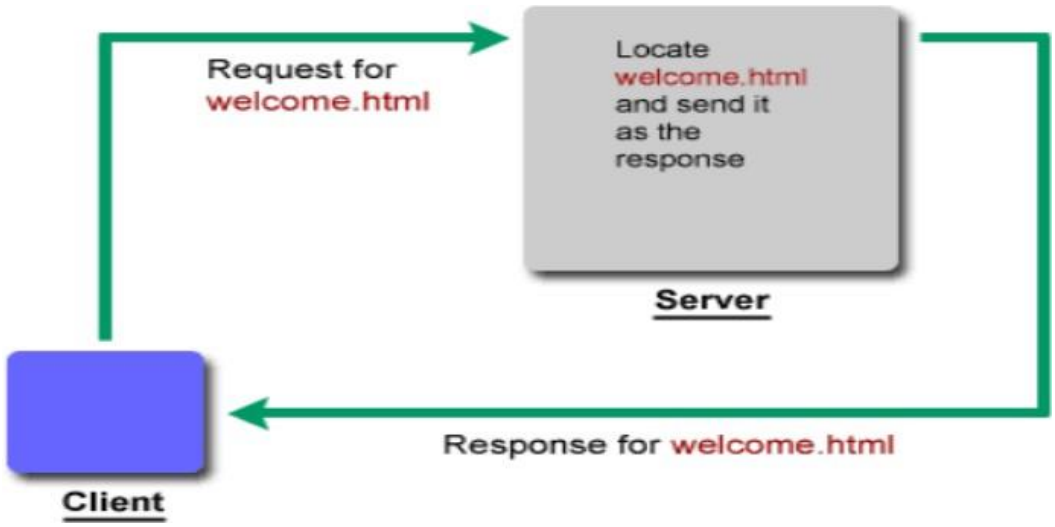
6. Parameters remain in browser history.

POST Method

1. Information sent from a form with the post method is invisible to others(all names/values are embedded within the body of the HTTP request)
2. Post has no limits on the amount of information to send.
3. $_POST is an array of variables passed to the current script via the HTTP POST method.
4. Cannot be bookmarked.
5. Not cached.
6. Parameters are not saved in browser history.

Server Role

1. Role of PHP in Web Applications.
2. PHP is a server side scripting language. That means its processing happens in the server by consuming server's resources and sends only the output to the client.
3. In a client side scripting language like JavaScript, processing happens in the client's compute Consuming its resources



4.2 Form controls: text box, text area, radio button, check box, list, button

1. Textbox

A text input field allows the user to enter a single line of text

Textbox field enable the user to input text information to be used by the program

The text field can be set using

<input type="text" size="30" name="user" value=" "

Step1:create a form in some HTML file as follows-

Input.html

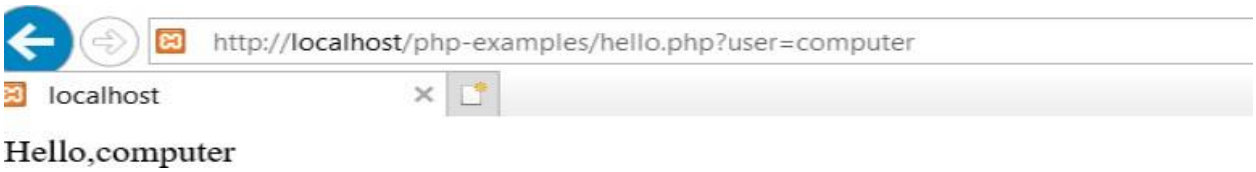
```
<form action="hello.php" method="get">
Name:<input type="text" name="user"/>
    <input type="submit" value="Submit"/>
</form>
```

Step 2:Now the hello.php program can be as follows-

```
<?php
$name=$_GET["user"];
echo"Hello,$name";
?>
```

step3

Step 4



2. Textarea

A text area field is similar to a text input field but it allows the user to enter multiple line of text

Syntax

<textarea name=name of component row="some number"cols="some number">< /textarea>

Eg<textarea name="address" rows="5" cols="40">< /textarea>

3. Checkbox

<form action="#" method="post">

```
<input type="checkbox" name="gender" value="Male">Male</input>
<input type="checkbox" name="gender" value="Female">Female</input>
<input type="submit" name="submit" value="Submit"/>
</form>
<?php
if (isset($_POST['gender'])){
echo $_POST['gender']; // Displays value of checked checkbox.
} ?>
```

To get value of multiple checked checkboxes,

```
<form action="#" method="post">
<input type="checkbox" name="check_list[]"value="C/C++"><label>C/C++</label><br/>
<input  type="checkbox"name="check_list[]"value="Java"><label>Java</label><br/>
<input type="checkbox" name="check_list[]"value="PHP"><label>PHP</label><br/>
<input type="submit" name="submit" value="Submit"/>
</form>

<?php
if(isset($_POST['submit']))
{//to run PHP script on submit
if(!empty($_POST['check_list']))
{
// Loop to store and display values of individual checked checkboxes.
foreach($_POST['check_list'] as $selected){
echo $selected."</br>";
}
}
}
?>
```

4. List Box

```
<html>
<body>
<div>
<form action="<?php echo $_SERVER['PHP_SELF'] ?>" method="post">
<label> Best Engineering College in Maharashtra</label>
<p><select name="products[]" multiple="multiple">
 <option>VESIT</option>
 <option>VJTI</option>
 <option>COEP</option>
 <option>PCIT</option>
```

```
 <option>IIT</option>
 <option>AVCOE</option>
 <option>DY Patil</option>
 <option>BATU</option>
</select></p>
<p><input type="submit" value="Test me!" /></p>
</form>
</div></body></html>
<?php
 if (is_array ( $_POST ['products'] )) {
   print "<p>Your product choices are:</p>";
   print "<ul>";
   foreach ( $_POST ['products'] as $value ) {
     print "<li>$value</li>\n";
   }
   print "</ul>";
 }?>
```

4.3 Working with multiple forms:

A web page having multiple forms It can be processed in two steps:

1. Posting each form to a different PHP script file for processing Multiple functionality can be provided in a single web page by providing multiple forms in a web page having different functionality.
2. Each form on this web page will be given a separate name that will uniquely identify the form in the web page with multiple forms.

Data from each form should be given to a separate PHP script file for processing by specifying PHP script filename in the action attribute of the form.

Each PHP script should be written in such a fashion that will handle all the data coming from the form.

Disadvantage - we have to write separate files for each form.

1.A web page having many forms

```
<form method="post">
<h3>personal information form</h3>
user name:<input type="text" name="username"/>
<br/><br/>
address:<input type="text" name="address"/>
<br/><br/>
<input type="submit" name="submit_personal_info" value="submit"/>
<br/>
```

```
</form>
<form method="post">
<h3>Feedback Form</h3>
<textarea name="feedback" rows="5" cols="50"></textarea>
<br/>
<input type="submit" name="submit_feedback"/>
</form>
<?php
if(!empty($_POST['submit_personal_info']))
{
        echo "<h3>welcome user:".$_POST['username']."</h3>";
 }
if(!empty($_POST['submit_feedback']))
{
echo "<h3>we value your feedback:</h3>";
echo "your feedback is<br/>".$_POST["feedback"];
}
?>
```

2. A form having multiple submit buttons.

```
<form method="post">
<h3>Simple Arithmetic Calculator</h3>
Number 1: <input type="text" size="5" name="num1"/>
<br/><br>
Number 2: <input type="text" size="5" name="num2"/>
<br/><br>
<input type="submit" name="add" value="ADDITION"/>
<input type="submit" name="sub" value="SUBTRACTION"/>
<input type="submit" name="mul" value="MULTIPLICATION"/>
<input type="submit" name="div" value="DIVISION"/>
<?PHP
if(!empty($_POST['add'])) {
        $result=$_POST['num1']+$_POST['num2'];
        echo "<h3> Addition: ".$result."</h3>";
}
if(!empty($_POST['sub']))
{
        $result=$_POST['num1']-$_POST['num2'];
        echo "<h3> Subtraction: ".$result."</h3>";
}
if(!empty($_POST['mul']))
```

```
{
        $result=$_POST['num1']*$_POST['num2'];
        echo "<h3> Multiplication: ".$result."</h3>";
}
if(!empty($_POST['div']))
{
        $result=$_POST['num1']/$_POST['num2'];
        echo "<h3> Division: ".$result."</h3>";
}
?>
```

4.4 Web Page Validation

What is Validation ?

Required field will check whether the field is filled or not in the proper way. Most of cases we will use the * symbol for required field.

Validation means checking the input submitted by the user. There are two types of validation available in PHP. They are as follows −

- Client-Side Validation − Validation is performed on the client machine web browsers.
- Server Side Validation − After submitted by data, The data is sent to a server and performs validation checks in the server machine.

Some of Validation rules for field

Field	Validation Rules
Name	Should required letters and white-spaces
Email	Should be required @ and .
Website	Should required a valid URL

Radio	Must be selectable at least once
Check Box	Must be checkable at least once
Drop Down menu	Must be selectable at least once

Example of validation

1.Valid URL

Below code shows validation of URL

```
$website = input($_POST["site"]);
if
(!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
  $websiteErr = "Invalid URL";
}
```

Above syntax will verify whether a given URL is valid or not. It should allow some keywords as https, ftp, www, a-z, 0-9,..etc..

2.Valid Email

Below code shows validation of Email address

```
$email = input($_POST["email"]);
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
  $emailErr = "Invalid format and please re-enter valid email";
}
```

Above syntax will verify whether a given Email address is well-formed or not.if it is not, it will show an error message.

Predefined function for validation in php

1. The filterName() function validates input value as a person's name. A valid name can only contain alphabetical characters (a-z, A-Z).

2. The filterEmail() function validates input value as email address.

3. The filterString() function only sanitizes the input value by stripping HTML tags and special characters. It doesn't validate the input valueagainstanything.

4.5 Cookie

1.Cookie is a small piece of information stored as a file in the user's browser by the web server.

2.Once created, cookie is sent to the web server as header information with every HTTP request.

3. You can use cookies to save any data but it should not exceed 1K(1024 bytes) in size.

4. Before we move on to how to create, update and delete cookies, let's learn a few real world uses of cookies.

Here,

1) A user requests for a page that stores cookies

2) The server sets the cookie on the user's computer

3) Other page requests from the user will return the cookie name and value

1. Why and when to use Cookies?

- Http is a stateless protocol; cookies allow us to track the state of the application using small files stored on the user's computer.
  The path where the cookies are stored depends on the browser.
  Internet Explorer usually stores them in the Temporary Internet Files folder.
- Personalizing the user experience – this is achieved by allowing users to select their preferences.
  The page requested that follows are personalized based on the set preferences in the cookies.
- Tracking the pages visited by a user
- To store user information like when he/she visited, what pages were visited on the website etc, so that next time the user visits your website you can provide a better user experience.
- To store basic website specific information to know this is not the first visit of a user.

- You can use cookies to store the number of visits or view counters.
- I hope this gives you an idea about how you can utilize cookies in your web application.

2. Types of Cookies

There are two types of cookies, they are:

1. **Session Cookie**: This type of cookies are temporary and expire as soon as the session ends or the browser is closed.
2. **Persistent Cookie:** To make a cookie persistent we must provide it with an expiration time. Then the cookie will only expire after the given expiration time, until then it will be a valid cookie.
3. You can use cookies to store number of visits or view counter

3.Creating a Cookie in PHP

In PHP we can create/set a cookie using the setcookie() function.

syntax for the function,

setcookie(name, value, expire, path, domain, secure)

The first argument which defines the name of the cookie is mandatory, rest all are optional arguments.

Argument	What is it for?
name	Used to specify the name of the cookie. It is a mandatory argument. Name of the cookie must be a string.
value	Used to store any value in the cookie. It is generally saved as a pair with a name. For example, name is *userid* and value is *7007*, the userid for any user.

expire	Used to set the expiration time for a cookie. if you do not provide any value, the cookie will be treated as a session cookie and will expire when the browser is closed.
path	Used to set a web URL in the cookie. If set, the cookie will be accessible only from that URL. To make a cookie accessible through a domain, set '/' as cookie path.
domain	The domain of your web application. It can be used to limit access to cookies for sub-domains.
secure	If you set this to 1, then the cookie will be available and sent only over HTTPS connection.

E.g So if we want to **create a cookie** to store the name of the user who visited your website, and set an expiration time of a week, then we can do it like this,

```
<?php
setcookie("username", "Shika", time()+60*60*24*7);
?>
```

To access a stored cookie we use the $_COOKIE global variable, and can use the isset() methods to check whether the cookie is set or not.

Let's have a complete example where we will set a cookie and then retrieve it to show its value in the HTML page.

```
<?php
// set the cookie
setcookie("username", "Ramkra", time()+60*60*24*7);
?>
<html>
  <body>
    <?php
  // check if the cookie exists
  if(isset($_COOKIE["username"]))
  {
     echo "Cookie set with value: ".$_COOKIE["username"];
  }
```

```
  else
  {
     echo "cookie not set!";
  }
  ?>

  </body>
</html>
```

So by providing the name of the cookie inside the square brackets with the global variable $_COOKIE[] we can access the cookie.

NOTE: setcookie() function should be placed before the starting HTML tag(<html>).

Updating Cookie in PHP

To update/modify a cookie, simply set it again. For example, if we want to update the username stored in the cookie created above, we can do it using setcookie() method again,

```
<?php
// updating the cookie
setcookie("username", "Ambarish", time()+60*60*24*7);
?>
<html>
  <body>
  <?php
  // check if the cookie exists
  if(isset($_COOKIE["username"]))
  {
     echo "Cookie set with value: ".$_COOKIE["username"];
  }
  else
  {
     echo "cookie not set!";
  }
  ?>
   </body>
</html>
```

We just update the value of the username cookie

Delete a Cookie in PHP

To delete/remove a cookie, we need to expire the cookie, which can be done by updating the cookie using the setcookie() function with expiration date in past.

```
<?php
```

```
// updating the cookie
setcookie("username", "Dhruv", time() - 3600);
?>
<html>
  <body>

  <?php

 echo "cookie username is deleted!";
  ?>
        </body>
</html>
```

And with this, we now know how to create a cookie, how to update it and how to delete it when we no longer need it.

4.6 What is a PHP Session?

1.A session is a way to store information (in variables) to be used across multiple pages.Unlike a cookie, the information is not stored on the user's computer.

2. When you work with an application, you open it, do some changes, and then you close it. This is much like a Session.

3. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.

4. Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.

5. So; Session variables hold information about one single user, and are available to all pages in one application.

Tip: If you need permanent storage, you may want to store the data in a database.

PHP mail

PHP mail is the built-in PHP function that is used to send emails from PHP scripts.

The mail function accepts the following parameters;

Email address

Subject

Message

CC or BCC email addresses

Use of Mail()

1. It’s a cost effective way of notifying users on important events.

2. Let users contact you via email by providing a contact us form on the website that emails theprovided content.
3. Developers can use it to receive system errors by email
4. You can use it to email your newsletter subscribers.
5. You can use it to send password reset links to users who forget their passwords
6. You can use it to email activation/confirmation links.
7. This is useful when registering users and verifying their email addresses

Sending mail using PHP

The PHP mail function has the following basic syntax

```
<?php
mail($to_email_address,$subject,$message,[$headers],[$parameters]);
?>
```

HERE,

"$to_email_address" is the email address of the mail recipient
"$subject" is the email subject
"$message" is the message to be sent.
"[$headers]" is optional, it can be used to include information such as CC, BCC
CC is the acronym for carbon copy. It's used when you want to send a copy to an interested person i.e. a complaint email sent to a company can also be sent as CC to the complaints board. BCC is the acronym for blind carbon copy. It is similar to CC. The email addresses included in the BCC section will not be shown to the other recipients.

Simple Mail Transmission Protocol (SMTP)

PHP mailer uses Simple Mail Transmission Protocol (SMTP) to send mail.On a hosted server, the SMTP settings would have already been set.The SMTP mail settings can be configured from **"php.ini"** & "**sendemail.ini "** file in the PHP installation folder.

1. File :-Php.ini

```
//email function
smtp_port=465
sendmail_from = testmailkale@gmail.com
sendmail_path = "\"C:\xampp\sendmail\sendmail.exe\" -t"
mail.add_x_header=on
```

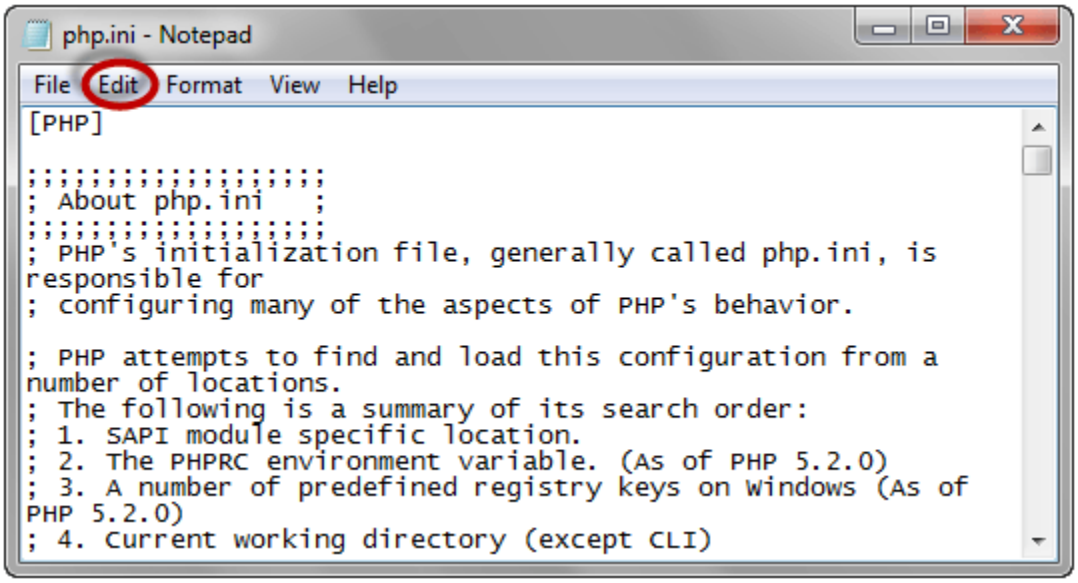
2. File:- sendemail.ini

```
smtp_server=smtp.gmail.com
smtp_port=465
smtp_ssl=auto
error_logfile=error.log
auth_username=testmailkale@gmail.com
```

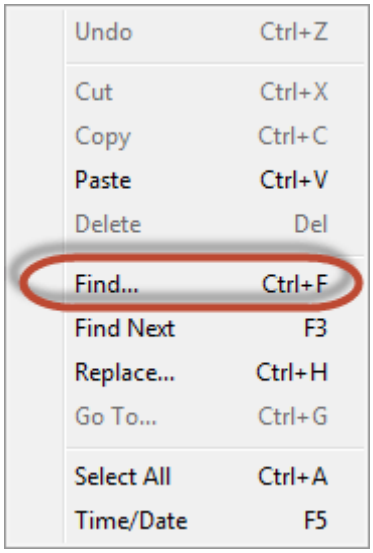
auth_password=

1. Configure SMTP settings on your localhost Assuming you are using xampp on windows, locate the “php.ini” in the directory “C:\xampp\php”.

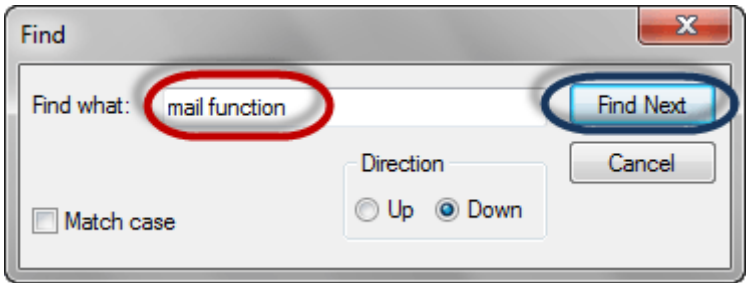
- Open it using a notepad or any text editor. We will use a notepad in this example. Click on the edit menu



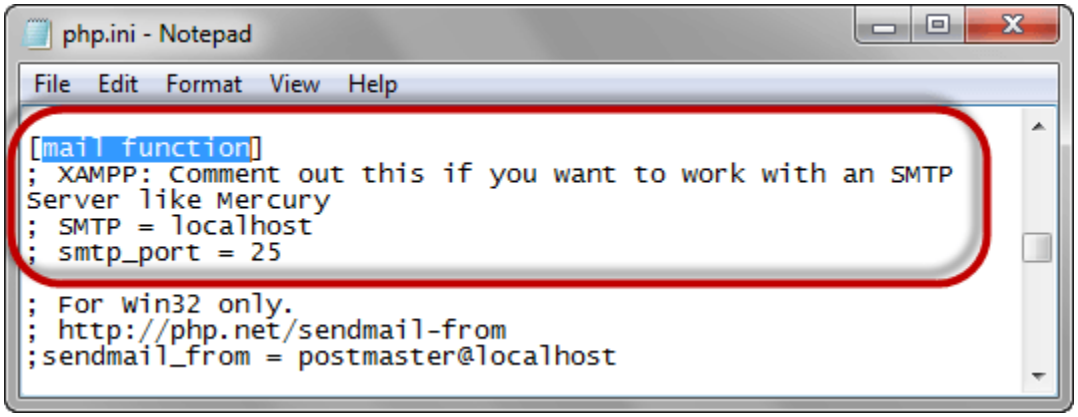
- Click on Find… menu



- The find dialog menu will appear



- Click on Find Next button



- Locate the entries
  - *[mail function]*
  - *; XAMPP:* Don't remove the semicolon if you want to work with an SMTP Server like Mercury
  - ; SMTP = localhost
  - ; smtp_port = 25
  - Remove the semi-colons before SMTP and smtp_port and set the SMTP to your smtp server and the port to your smtp port. Your settings should look as follows
    - SMTP = smtp.example.com
    - smtp_port = 25
    - *Note the SMTP settings can be gotten from your web hosting providers.*
    - If the server requires authentication, then add the following lines.
      - auth_username = example_username@example.com
      - auth_password = example_password
      - Save the new changes.
      - Restart Apache server.

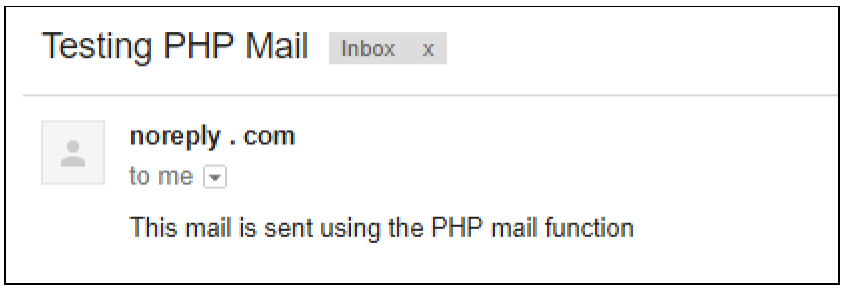
Php Mail Example

```
<?php
$to_email = 'name @ company . com';
$subject = 'Testing PHP Mail';
```

```
$message = 'This mail is sent using the PHP mail function';
$headers = 'From: noreply @ company . com';
mail($to_email,$subject,$message,$headers);
?>
```

t's now look at an example that sends a simple mail.

Output:



Note: the above example only takes the 4 mandatory parameters.