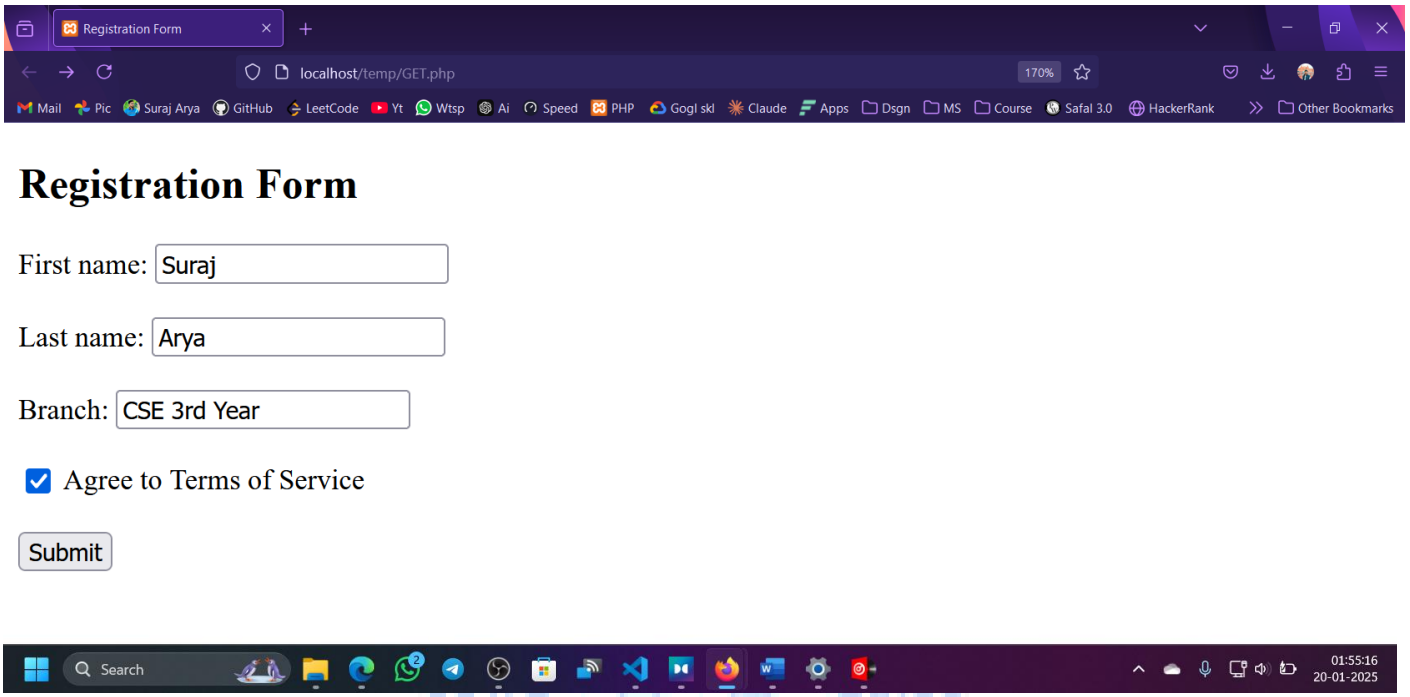
Practical No: 1

Object: Create Web forms and pages that properly use HTTP GET and POST protocol as appropriate.

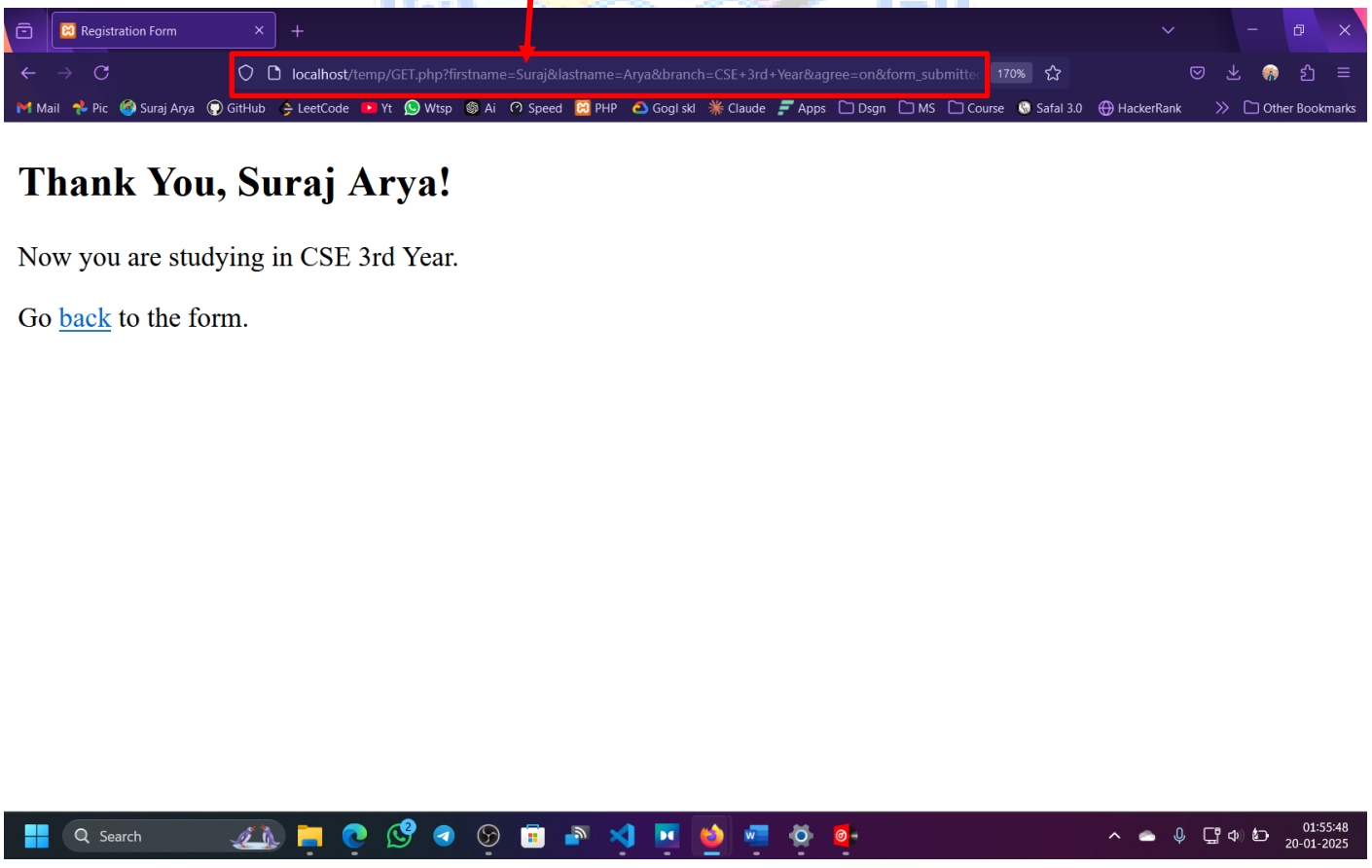
Creating a web form Using GET Method:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Registration Form</title>
</head>
<body>
    <?php if (isset($_GET['form_submitted'])): ?>
        <?php if (!isset($_GET['agree'])): ?>
            <p>You have not accepted our terms of service.</p>
            <p>Go <a href="javascript:history.back()">back</a> to the
form.</p>
        <?php else: ?>
            <h2>Thank You, <?php echo htmlspecialchars($_GET['firstname']) .
' ' . htmlspecialchars($_GET['lastname']); ?>!</h2>
            <p>Now you are studying in <?php echo
htmlspecialchars($_GET['branch']); ?>.</p>
            <p>Go <a href="javascript:history.back()">back</a> to the
form.</p>
        <?php endif; ?>
    <?php else: ?>
        <form action="" method="GET">
            <h2>Registration Form</h2>
            <label for="firstname">First name:</label>
            <input type="text" name="firstname" id="firstname"
required><br><br>
            <label for="lastname">Last name:</label>
            <input type="text" name="lastname" id="lastname"
required><br><br>
            <label for="branch">Branch:</label>
            <input type="text" name="branch" id="branch" required><br><br>
            <label>
                <input type="checkbox" name="agree">
                Agree to Terms of Service
            </label><br><br>
            <input type="hidden" name="form_submitted" value="1">
            <input type="submit" value="Submit">
        </form>
    <?php endif; ?>
</body>
</html>
```

Result: Using GET method.



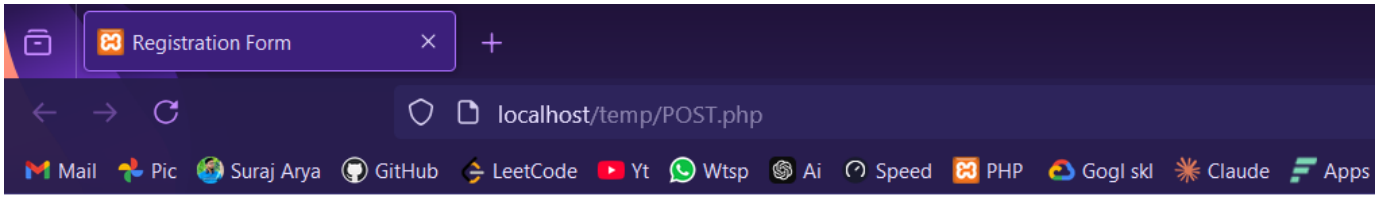
Appear in the address bar after submission



Creating a web form Using POST Method :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Registration Form</title>
</head>
<body>
    <?php if (isset($_POST['form_submitted'])): ?>
        <?php if (!isset($_POST['agree'])): ?>
            <p>You have not accepted our terms of service.</p>
            <p>Go <a href="javascript:history.back()">back</a> to the
form.</p>
        <?php else: ?>
            <h2>Thank You, <?php echo htmlspecialchars($_POST['firstname'])
. ' ' . htmlspecialchars($_POST['lastname']); ?>!</h2>
            <p>Now you are studying in <?php echo
htmlspecialchars($_POST['branch']); ?>.</p>
            <p>Go <a href="javascript:history.back()">back</a> to the
form.</p>
        <?php endif; ?>
    <?php else: ?>
        <form action="" method="POST">
            <h2>Registration Form</h2>
            <label for="firstname">First name:</label>
            <input type="text" name="firstname" id="firstname"
required><br><br>
            <label for="lastname">Last name:</label>
            <input type="text" name="lastname" id="lastname"
required><br><br>
            <label for="branch">Branch:</label>
            <input type="text" name="branch" id="branch" required><br><br>
            <label>
                <input type="checkbox" name="agree">
                Agree to Terms of Service
            </label><br><br>
            <input type="hidden" name="form_submitted" value="1">
            <input type="submit" value="Submit">
        </form>
    <?php endif; ?>
</body>
</html>
```

Result: Using GET method.



Registration Form

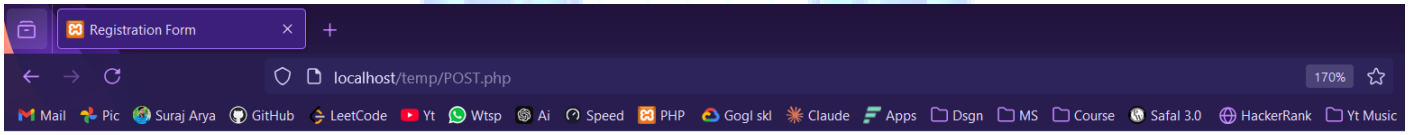
First name: Arya

Last name: Suraj

Branch: CSE

Agree to Terms of Service

Submit



Thank You, Arya Suraj!

Now you are studying in CSE.

Go back to the form.

Practical No: 2

Object:

Create PHP code that utilizes the commonly used API library functions built in to PHP

Theory: The full name of the API is Application Prepondering interferon Pinte Programming code that helps to communicate between two differ function is programs. Many functions are used in the API, of which the most common library is:

Requests: Requests are a library function that helps in issuing HTTP Requests. Using the Requests library tells us what action we are going to perform by calling the API. A total of four types of actions t performed in the Requests library is:

GET: This is the most common request method. Using this, we can get the data that we want from the APL

POST: You can add new data to the server using this request method.

PUT: Using this request method, you can change the information present in the server.

DELETE: Using this request method, we can delete the information present in the server which does not work in the server.

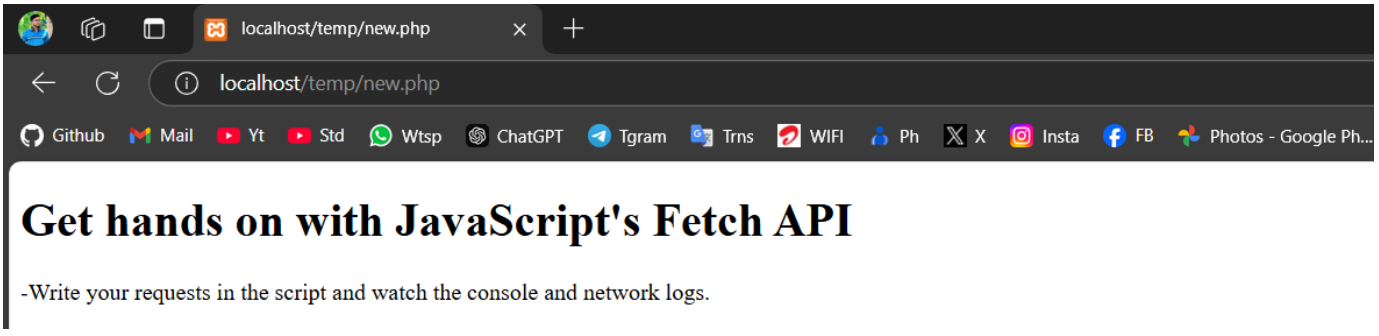
Programming:

```
<?php

// Include the Requests library (assuming you have it installed via
Composer)
require_once 'vendor/autoload.php';

// Define headers and authentication options
$headers = array('Accept' => 'application/json');
$options = array(
    'auth' => array('user', 'pass') // Replace with actual credentials
);

// Make a GET request to the GitHub API
$request = Requests::get('https://api.github.com/gists', $headers,
$options);

// Check the content type from the headers
var_dump($request->headers["content-type"]); // Expecting
"application/json; charset=utf-8"

// Check the body of the response (the content returned from the API)
var_dump($request->body);
?>
```

Creating a PHP code:

```
<html>
<body>
    <h1>Get hands on with JavaScript's Fetch API</h1>
    <p>-Write your requests in the script and watch the console logs.</p>
    <script>

        fetch('https://jsonplaceholder.typicode.com/todos')
            .then(response => response.json())
            .then(json => console.log(json));

        fetch('https://jsonplaceholder.typicode.com/todos/5')
            .then(response => response.json())
            .then(json => console.log(json));

        fetch("https://jsonplaceholder.typicode.com/todos", {
            method: 'POST',
            body: JSON.stringify({
                userId: 1,
                title: "clean room",
                completed: false
            }),
            headers: {
                "Content-type": "application/json; charset=UTF-8"
            }
        })
        .then(response => response.json())
        .then(json => console.log(json));

        // Delete task with id-1
        fetch('https://jsonplaceholder.typicode.com/todos/1', {
            method: 'DELETE'
        })
        .then(response => response.json())
        .then(json => console.log(json));
    </script>
</body>
</html>
```

Result: Thus created a first code that used the common library function requests.



Practical No: 3

Object: Creation of a Basic Blogging Website on WordPress.

Procedure (Detailed):

1. **Opening the Website:**
   - Launch a web browser (e.g., Chrome, Firefox, etc.).
   - In the address bar, type **www.wordpress.com** and press Enter. This will open the WordPress homepage.
2. **Selecting the Blog Option:**
   - On the WordPress homepage, you will see two main options:
     - **Create Website**
     - **Create Blog**
   - For this practical, choose the **Create Blog** option. This option is specifically designed for users who wish to create a blog-based website.
3. **Creating a WordPress Account:**
   - After selecting the **Create Blog** option, a new webpage will appear prompting you to sign up.
   - You will be required to fill out the following details:
     - **Email Address:** Enter your email address. This will be used for account verification and notifications.
     - **Username:** Choose a unique username for your WordPress account. This will be associated with your blog.
     - **Password:** Set a secure password to protect your account.
   - Once all the fields are filled, click on the **Create Your Account** button to proceed.
4. **Finalizing Blog Setup:**
   - After clicking on the **Create Your Account** button, WordPress will take you through a few more setup steps, such as selecting a blog theme, and you may need to verify your email address to complete the registration process.
   - Once these steps are completed, your blog will be created, and you will be able to access the WordPress dashboard. This dashboard is where you can manage posts, themes, and settings for your blog.
5. **Personalizing Your Blog:**
   - You can further personalize your blog by selecting a theme (design), adding posts, and configuring settings such as titles, taglines, and widgets.
   - You may also choose to upgrade your account to a paid version for additional features like custom domain names, more themes, and increased storage.

Result:
Thus, a basic blog website is created on WordPress, and you can now start adding content, customize the appearance, and share your blog with others.

Practical No: 4

Object: Design Website using WordPress

Theory:

To design a website on WordPress, the following steps are followed:

1. **Opening WordPress:**
    - Open a web browser and type **www.wordpress.com** in the address bar. Press Enter to open the website.
2. **Choosing Website Option:**
    - On the homepage of WordPress, two options are presented:
        - **Create Website**
        - **Create Blog**
    - Select the **Create Website** option.
3. **Creating Account:**
    - A new webpage will appear asking you to enter your details. You need to provide the following:
        - **Email ID:** Enter a valid email address.
        - **Username:** Choose a unique username.
        - **Password:** Create a secure password.
    - After filling in the details, click on the **Create Your Account** button.
    - Once the account is created, you will have access to the WordPress dashboard.
4. **Designing the Website:**

After creating the website, follow the steps below to design it:

i. **Make the Website Viewable by the Public:**

- Navigate to **Settings → Reading**.
- Ensure that the option **"Discourage search engines from indexing this site"** is unchecked to make your website visible to the public.

ii. **Set Title and Tagline:**

- Go to **Settings → General**.
- Set the **Title** and **Tagline** for your website to define its identity.

iii. **Allow Comments on Your Website:**

- Navigate to **Settings → Discussion**.
- Enable or disable comment settings to allow visitors to leave comments on your posts and pages.

iv. **Change the Theme:**

- Go to **Appearance → Themes** in the WordPress dashboard.
- Browse and select a theme that suits the design you want for your website.

v. **Customize Typography:**

- Click on **Settings → Typography**.
- You can change the font style, font color, background color, and other typography settings to personalize the website's text appearance.

vi. **Insert Plugins:**

- Plugins are used to extend the functionality of your website. To install plugins, navigate to **Plugins → Add New** and search for the desired plugins to install them.

vii. **Create Basic Pages:**

- To create a new page, go to **Pages → Add New** in the dashboard.
- You will be directed to a screen where you can add the page title and content.
  - **Title Section:** Enter the title of the page.
  - **Body Section:** Write the content for the page.
  - You can also insert images, audio, and video using the options in the editor.
- After creating the page, click **Publish** to make it live.

viii. **Navigate the Website:**

- Navigate through your website using the **Menu** option.
- Depending on the selected theme, you can add and arrange menu items that will appear on your website's navigation bar.

Result:

Thus, the website is designed on WordPress, with customizations for title, tagline, theme, typography, comments, plugins, and pages, along with navigation settings.

Practical No: 5

Object: Designing Web Pages using PHP, CSS, XHTML, Syntax, and Structure

Theory:

In this practical, we will design a webpage using PHP, CSS, and XHTML by structuring it with a header, body, and footer page. PHP is used to dynamically fetch and display the content. Below are the steps for creating the header, body, and footer pages:

1. Creating the Header Page (header.html):

The header page contains the opening structure of the website, including meta tags, title, and an optional logo or image.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Suraj Arya</title>
</head>
<body>
<header>
    <h1>Hey, I'm Suraj</h1>
    <p>This is Random Page</p>
    <p>Date: October 23, 2024</p>
</header>
</body>
</html>
```

Save this file as **header.html**.

2. Creating the Body Page (body.html):

The body page contains the main content of the website. Here is a placeholder text used for layout purposes.

```
<p>It helps the designer plan where the content will sit. It helps in
creating drafts of the content on the pages of the website. It originates
from the Latin text but is seen as gibberish. Sometimes, the reader gets
distracted while creating or working on the website. That’s why this
language is important. This tool makes the work easier for the webmaster.
</p>
```

Save this file as **body.html**.

3. Creating the Footer Page (footer.html):

The footer page typically contains copyright and other footer information.

```
<footer>
    <p>All rights reserved.</p>
</footer>
```

Save this file as **footer.html**.

4. Creating the Index Page (index.php):

The index page will fetch the content from the header, body, and footer pages using PHP's file_get_contents function and display them on the webpage.

```
<?php echo file_get_contents("html/header.html"); ?>
<?php echo file_get_contents("html/body.html"); ?>
<?php echo file_get_contents("html/footer.html"); ?>
<!-- Adding copyright and dynamic year -->
<p>Copyright © Suraj Arya <?php echo date("Y"); ?></p>
```

- The file_get_contents() function fetches the content of the HTML files and embeds it into the index page.
- The date("Y") function in PHP dynamically displays the current year.

Result:

When running the **index.php** file in a browser, it will display the content from the header, body, and footer files, along with a dynamically updated copyright notice showing the current year.