DEVOPS LAB

Class: CSM 3-2

Experiment 1:

Write code for a simple user registration form for an event.

Execution steps:

- Write all the three codes (pgm1.html, pgm1.js and pgm1.css) in three different notepads and save the files with same name.
- All the files should be kept in same directory.
- For output; go to folder and open the pgm1.html file.
- It opens in a browser.

Pgm1.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Sign up to {this}</title>
    <link rel="stylesheet" type="text/css" href="pgm1.css" />
    <script type="text/javascript" src="pgm1.js"></script>
  </head>
  <body>
    <form method="post" target="#" onsubmit="return validate()">
    <h1>Sign up to {this}</h1>
    <p>To be able to use all features of {this} you will require an account. The form below will allow you to create a new account if you don't have one yet.</p>
        Name<br>
        <input type="text" name="name"><br>
    Email<br>
```

```
        <input type="text" name="email"><br>
    Retype email<br>
        <input type="text" name="retype_email"><br>
    Create a password<br>
        <input type="password" name="password"><br>
    Retype password<br>
        <input type="password" name="retype_password"><br>
        <label>
      <input type="checkbox" name="terms">By signing up to {this} you agree to your
       <a href="#service">Terms of Service</a> and <a href="#policy">Privacy Policy</a>
    </label>
    <div class="button">
      <input type="submit" class="button" value="Set Up Account">
    </div>
        </form>
</body>
</html>
```

Pgm1.js

```
function validate() {
        var result = "";
        result += validateName();
        result += validateEmail();
        result += validatePassword();
        result += validateTerms();
        if(result != "")
        {
        alert("Validation Result:\n\n" + result);
        }
        else
        {
        alert("Congrats!!! Your account Created.");
        }
    }

function validateName() {
        var name = document.getElementsByName("name")[0].value;
        if (name.length <= 2)
                return "Name should be at least three characters.\n";
        return "";
}

function validatePassword() {
        var password = valueOf("password");
        var retype = valueOf("retype_password");
```

```
	if (password.length <= 5)
		return "Password should be at least 6 characters.\n";
	if (password != retype)
		return "Passwords do not match.\n";
	return "";
}

function validateEmail() {
	var email = valueOf("email");
	var retype = valueOf("retype_email");
	if (email.indexOf('@') == -1)
		return "Email should be a valid address.\n";
	if (email != retype)
		return "Email addresses do not match.\n";
	return "";
}

function validateTerms() {
	var terms = document.getElementsByName("terms")[0];
	if (!terms.checked)
		return "Please accept the Terms of Service and Privacy Policy";
	return "";
}
function valueOf(name) {
	return document.getElementsByName(name)[0].value;
}
```

Pgm1.css

```
body {
  font-family: Verdana, sans-serif;
}

input {
        border-radius: 5px;
        padding: 5px;
}

div.button {
        text-align: right;
}

a {
        text-decoration: none;
        color: #e51;
}

form {
        width: 480px;
        margin: auto;
        padding: 5px;
        border: solid black 1px;
        border-radius: 5px;
        box-shadow: 5px 5px 2px #888;
}
```

DEVOPS LAB

Class: CSM 3-2

Experiment 2 : Explore Git and GitHub commands.

Git: Git is a version-control system for tracking changes in source code during software development.

WHat is Bash? : Bash(also know as "Bourne Again SHell) is an implmentation of Shell and allows you to efficiently perform many tasks.

Features:

- easily navigate your computer to manage files and folders
- run programs that provide more functionality at the command line (e.g. git).
- launch programs from specific directories on your computer
- use repeatable commands for these tasks across many different operating systems (WIndows, Mac, Linux).

There are many Version Control Sysytem Tools amongst all Git is popular.

other tools like:SVN, CVS, mercurial.

Install Git using below link

https://git-scm.com/downloads

Choose Windows

Download and Install

Goto Start button and select Git Bash

Console window will be opened

Try the below Commands:

$ Git --version

$ mkdir <directory name>

$ CD <directory name>

$ dir etc

GitHub

SIgnup with Github using below link

https://github.com/

EXPERIMENT NO: 3. Practice Source code management on GitHub. Experiment with the source code written in exercise 1

Aim: Practice Source code management on GitHub. Experiment with the source code written in exercise 1

Description:

To practice source code management on GitHub, you can follow these steps:

- Create a GitHub account if you don't already have one.
- Create a new repository on GitHub.
- Clone the repository to your local machine: $ git clone <repository-url>
- Move to the repository directory: $ cd <repository-name>
- Create a new file in the repository and add the source code written in exercise 1.
- Stage the changes: $ git add <file-name>
- Commit the changes: $ git commit -m "Added source code for a simple user registration form"
- Push the changes to the remote repository: $ git push origin master
- Verify that the changes are reflected in the repository on GitHub.

These steps demonstrate how to use GitHub for source code management. You can use the same steps to manage any source code projects on GitHub. Additionally, you can also explore GitHub features such as pull requests, code review, and branch management to enhance your source code management workflow.

STEPS TO BE FOLLOWED TO INSTALL 'JENKINS' ON WINDOWS

STEP-1:Install Java Development Kit (JDK):

Make sure that there is a 'java jdk 17' software version has already installed in the system,
if not click the link to install
"https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.exe"

STEP-2: Set the Path for the Environmental Variable for JDK:

STEP-3: Install jenkins using the link
"https://www.jenkins.io/download/thank-you-downloading-windows-installer-stable"

STEP 4: In the Jenkin Setup screen, click Next.

STEP-5: Choose the location where you want to have the Jenkins instance installed (default location is C:\Program Files (x86)\Jenkins),
then click on Next button.

STEP-6: For Service logon credentials, choose Logon type as "Run Service as LocalSystem".

STEP-7: Choose the port available on the system and test it (Recommended is 8080).

STEP-8: Choose the JDK path and click Next.

STEP-9: Click on the Install button.

STEP-10: Once install is complete, click Finish.

STEP-11: After completing the Jenkins installation process ,Open the web Browser(Chrome) and type 'http://localhost:8080'

STEP-12: To Unlock Jenkins ,administration password has to be entered and you can get this from the link present just above in the form like 'C:\Program Files(x86)\Jenkins\secrets\initialAdminPassword'(Not recommended to copy this).

NOTE:Recommended to copy the link that is present on the screen.

STEP-13: Open the highlighted file using Notepad and copy the content of the initialAdminPassword file.

STEP-14: Paste the password it into browser’s pop-up tab (http://localhost:8080/login?form=%2F) and click on Continue button.

STEP-15: Click on the “Install suggested plugins button” so Jenkins will retrieve and install the essential plugins

STEP-16: After all suggested plugins were installed, the “Create First Admin User” panel will show up.
Fill all the fields with desired account details and hit the “Save and Finish” button.

NOTE:Remember the Username and Password to login further.

We have successfully installed a new Jenkins Server. Hit the “Start using Jenkins” button.

Login with the username and password to get started with jenkins.

EXPERIMENT NO: 5. Demonstrate continuous integration and development using Jenkins.

Aim: Demonstrate continuous integration and development using Jenkins.

DESCRIPTION

Continuous Integration (CI) and Continuous Development (CD) are important practices in software development that can be achieved using Jenkins. Here's an example of how you can demonstrate CI/CD using Jenkins:

Create a simple Java application:

- Create a simple Java application that you want to integrate with Jenkins.
- The application should have some basic functionality, such as printing "Hello World" or performing simple calculations.

Commit the code to a Git repository:

- Create a Git repository for the application and commit the code to the repository.
- Make sure that the Git repository is accessible from the Jenkins server.

Create a Jenkins job:

- Log in to the Jenkins web interface and create a new job.
- Configure the job to build the Java application from the Git repository.
- Specify the build triggers, such as building after every commit to the repository.

Build the application:

- Trigger a build of the application using the Jenkins job.
- The build should compile the code, run any tests, and produce an executable jar file.

Monitor the build:

- Monitor the build progress in the Jenkins web interface.
- The build should show the build log, test results, and the status of the build.

Deploy the application:

- If the build is successful, configure the Jenkins job to deploy the application to a production environment.
- The deployment could be as simple as copying the jar file to a production server or using a more sophisticated deployment process, such as using a containerization technology like Docker.

Repeat the process:

- Repeat the process for subsequent changes to the application.

- Jenkins should automatically build and deploy the changes to the production environment.

This is a basic example of how you can use Jenkins to demonstrate CI/CD in software development. In a real-world scenario, you would likely have more complex requirements, such as multiple environments, different types of tests, and a more sophisticated deployment process. However, this example should give you a good starting point for using Jenkins for CI/CD in your software development projects.