**Name:Vishal Chauhan**

**PRN:2020BTECS00090**

**Sub:SET-Lab**

**Ass:5**

Q 1. What is version control system and why it is important?

A version control system (VCS) is a software tool used to manage changes made to software code or other collections of files. It keeps track of the changes made over time and allows users to easily compare and revert to previous versions of the files. VCS is important because it helps manage code changes made by multiple people working on the same project, prevents loss of data, enables collaboration and teamwork, and provides an audit trail of all changes made to the code.

Q 2. Illustrate different types of version control system with example.

Some of the different types of version control systems are:

* Centralized VCS: SVN, CVS
* Distributed VCS: Git, Mercurial
* Visual VCS: TFS, Perforce
* Cloud-based VCS: Bitbucket, Github

Q 3. Perform below operations using CVS

a. cvs checkout: This command is used to download a copy of a repository from the CVS server to your local machine.

b. cvs update: This command is used to update your local copy of a repository with changes made to the version on the CVS server.

c. cvs add: This command is used to add a new file or directory to a repository.

d. cvs remove: This command is used to remove a file or directory from a repository.

e. cvs commit: This command is used to save changes made to your local copy of a repository back to the CVS server.

Q 4. Differentiate Between The Git & SVN Repository?

Git is a distributed VCS, while SVN is a centralized VCS. Git allows multiple people to work on the same codebase at the same time without a central repository. In contrast, SVN has a central repository that all changes are checked into. Git is generally faster than SVN, as it can perform most operations locally, while SVN requires communication with the central server. Git is better suited for large projects with many contributors, while SVN is better suited for smaller projects with a few contributors.

Q 5. What is “branch”, “tag” And “trunk” In SVN?

In SVN, a "branch" is a copy of the codebase that can be modified independently of the main codebase, often used for new features or experimental changes. A "tag" is a snapshot of a specific version of the codebase that is marked with a name, often used to mark a release version. The "trunk" is the main development line of the codebase in SVN.

Q 6. How CVS is different from SVN?

CVS is a centralized VCS, while SVN is an improved version of CVS that also has support for branching and merging. SVN has better handling of binary files, and allows atomic commits, while CVS does not. SVN also has better support for renaming and moving files than CVS.

Q.7 Demonstrate a display the app version in angular.

In Angular, you can display the app version using the following code in the app.component.ts file:

import { Component, VERSION } from '@angular/core';

@Component({

selector: 'app-root',

template: <h1>My App Version is {{ version }}</h1>,

})

export class AppComponent {

version = VERSION.full;

}

Q.8 Build a simple web app with Express and Angular.

Please refer to the provided link for instructions and code:<https://www.geeksforgeeks.org/build-a-simple-web-app-with-express-angular/?ref=rp>

⇒To build a simple web app with Express and Angular, you can follow these steps:

1>Create a new project directory and navigate into it.

Initialize a new Node.js project using npm init. This will create a package.json file.

Install the required dependencies by running the following command in the terminal:

npm install express body-parser cors mongoose --save

npm install @angular/cli --save-dev

2>Create a new Express application by creating a new file named server.js and adding the following code:

const express = require('express');

const bodyParser = require('body-parser');

const cors = require('cors');

const app = express();

*// Middlewares*

app.use(bodyParser.json());

app.use(cors());

*// Routes*

app.get('/', (req, res) => {

res.send('Hello from Express!');

});

*// Start server*

const PORT = process.env.PORT || 3000;

app.listen(PORT, () => console.log(`Server running on port ${PORT}`));

-->This will create a simple Express server that listens on port 3000 and responds with a "Hello from Express!" message when you visit http:*//localhost:3000.*

3>Create a new Angular app using the Angular CLI by running the following command in the terminal:

ng new client

This will create a new Angular app in a folder named client.

4>Navigate into the client folder and start the Angular app by running the following command in the terminal:

ng serve

This will start the app on port 4200 and you can access it by visiting http:*//localhost:4200.*

5>In the client folder, open src/app/app.component.html and add the following code:

html

<div>

<h1>Hello from Angular!</h1>

<p>{{ message }}</p>

</div>

This will create a simple Angular component that displays a message.

6>In the client folder, open src/app/app.component.ts and add the following code:

javascript

import { Component } from '@angular/core';

import { HttpClient } from '@angular/common/http';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

})

export class AppComponent {

message: string;

constructor(private http: HttpClient) {}

ngOnInit() {

*this*.http.get('/api').subscribe((data: any) => {

*this*.message = data.message;

});

}

}

This will create an Angular component that retrieves data from the Express server and displays it in the HTML template.

7>Modify the Express server by adding the following code to the server.js file:

javascript

const mongoose = require('mongoose');

mongoose.connect('mongodb://localhost/myapp');

const db = mongoose.connection;

db.on('error', console.error.bind(console, 'connection error:'));

db.once('open', function() {

console.log('Connected to MongoDB');

});

*// Routes*

app.get('/api', (req, res) => {

res.send({ message: 'Hello from Express!' });

});

This will connect to a MongoDB database and add a new route that responds with a JSON object.

8>Restart the server by running node server.js in the terminal.

9>Restart the Angular app by stopping the ng serve command in the terminal and running it again.

Now you should be able to access the web app at http:*//localhost:4200 and*

see the message "Hello from Angular!" along with the message "Hello from Express!" retrieved from the server.

Q.9 What is git version control?

Git is a distributed version control system used to manage changes made to software code or other collections of files. It allows multiple people to work on the same codebase at the same time without a central repository. It keeps track of changes made over time and allows users to easily compare and revert to previous versions of the files. Git also provides features for collaboration and teamwork, including branch and merge support, and can be used for both small and large projects.

Q.10 Demonstrate creation of repository in git.

To create a repository in Git, follow these steps:

1. Open your terminal or Git Bash.
2. Navigate to the directory where you want to create the repository.
3. Type git init to initialize a new Git repository.
4. Add the files and folders you want to track to the repository using git add.
5. Commit the changes using git commit -m "Initial commit".

Your repository is now created and the initial commit has been made. You can continue making changes and committing them to the repository using Git commands.