**Software Engineering Tools Lab Assignment No-6**

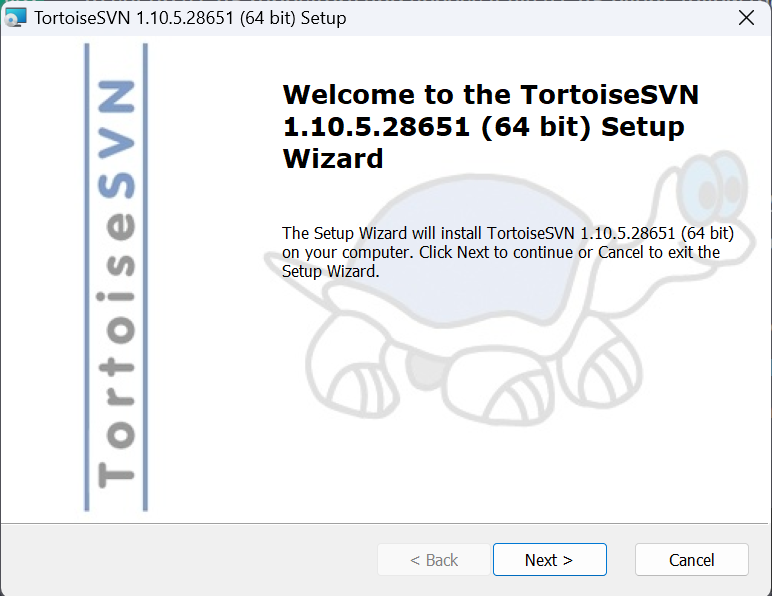
(Module 4- Version control)

**Q 1. What is Microsoft’s VSS? Provide the information of VSS tool with respect to below points.**

1. **Owner/ developer**
2. **Brief information/introduction**
3. **Basic operations involved**
4. **Advantages**
5. **Disadvantages**

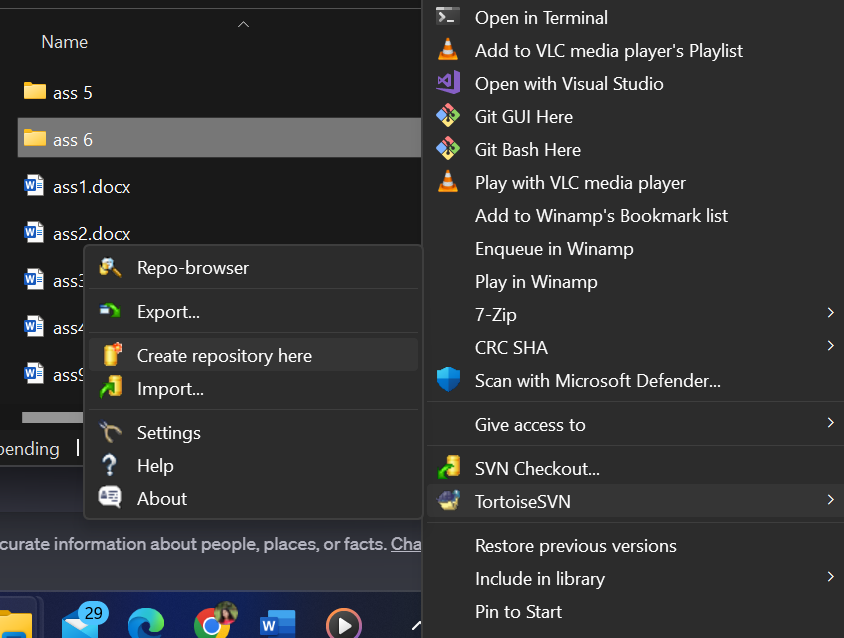
**Q 2. Create a SVN repository and perform below operations on that repository using SVN. Also explain below operations.**

1. Install TortoiseSVN Client

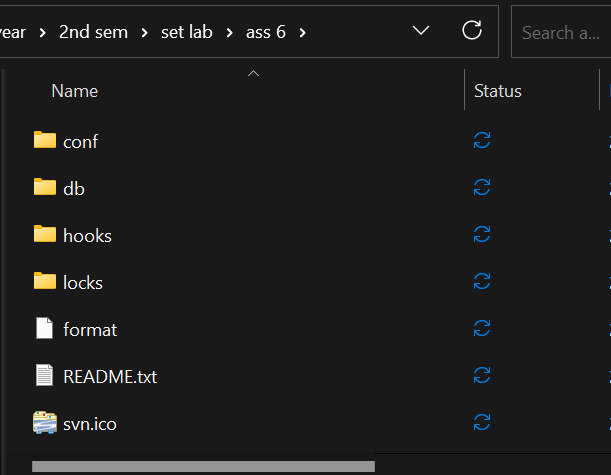
**

2.create repository for desired folder

(folder should be empty)

****

Default project structure having trunk/branch/tag

****

1. **Revert**
2. **Import**
3. **Checkout**
4. **Commit**
5. **Update**
6. **Copy**

**Q 3. Perform below operations using CVS**

**a. cvs checkout**

To checkout a module from the repository, you can use the cvs checkout command followed by the name of the module you want to checkout. For example:

cvs checkout mymodule

This will create a local copy of the mymodule module in your current directory.

**b . cvs update**

To update your local copy of the module with the latest changes from the repository, use the cvs update command followed by the name of the module. For example:

cvs update mymodule

This will retrieve any changes made to the mymodule module in the repository and apply them to your local copy.

**c. cvs add**

To add a new file to the repository, you can use the cvs add command followed by the name of the file. For example:

cvs add myfile.txt

This will add the myfile.txt file to the repository.

**d. cvs remove**

To remove a file from the repository, you can use the cvs remove command followed by the name of the file. For example:

cvs remove myfile.txt

This will remove the myfile.txt file from the repository.

**e. cvs commit**

To commit changes made to a file or files in your local copy of the module back to the repository, use the cvs commit command followed by the name of the file or files. For example:

cvs commit myfile.txt

This will commit any changes made to the myfile.txt file in your local copy of the module back to the repository. You will be prompted to enter a commit message to describe the changes made.

**Q 4. Differentiate Between The Git & SVN Repository?**

Git and SVN are both version control systems that are widely used for software development. While both systems share some similarities, they also have some fundamental differences. Here are some key differences between Git and SVN repositories:

Distributed vs. Centralized: Git is a distributed version control system, while SVN is centralized. This means that with Git, every developer has a complete copy of the repository on their local machine, and can work independently and offline. In contrast, with SVN, there is a central repository that all developers connect to, and changes must be made to the central repository.

Branching and Merging: Git is generally considered to be superior to SVN when it comes to branching and merging. With Git, branching is cheap and easy, and merging is generally straightforward. This makes it easy to experiment with different versions of a codebase, and to collaborate on multiple features simultaneously. SVN, on the other hand, can be more difficult to work with when it comes to branching and merging, and conflicts can be more common.

Performance: Git is generally faster than SVN, especially when it comes to large repositories or repositories with many branches. This is because Git is designed to work with local copies of the repository, while SVN requires network communication with the central repository for many operations.

Learning Curve: Git has a steeper learning curve than SVN, especially for developers who are new to version control. This is partly because Git is more flexible and powerful than SVN, but also because Git has a more complex command-line interface.

Integration with Other Tools: SVN has been around for longer than Git, and as a result, it has better integration with some development tools, such as IDEs and issue trackers. However, Git has become much more popular in recent years, and most development tools now support both Git and SVN.

Overall, both Git and SVN have their own strengths and weaknesses, and the choice of which system to use will depend on the specific needs and preferences of the development team.

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

In SVN, the terms "branch," "tag," and "trunk" refer to specific directories within a repository that are used to organize the codebase and track changes over time. Here's what each term means:

Branch: A branch is a copy of the codebase that diverges from the main development line (also known as the trunk). Branches are typically used to experiment with new features or to work on bug fixes without affecting the main codebase. Once changes have been made in a branch, they can be merged back into the trunk or into another branch.

Tag: A tag is a specific snapshot of the codebase that is marked with a descriptive name or version number. Tags are typically used to indicate significant releases or milestones in the development process, and are often used for archival or auditing purposes. Unlike branches, tags are not intended to be modified or merged back into the main codebase.

Trunk: The trunk is the main development line of the codebase, and is typically where the latest version of the code resides. Changes made in branches are eventually merged back into the trunk once they have been tested and reviewed.

In SVN, these directories are typically organized as follows:

/trunk: This directory contains the latest version of the codebase.

/branches: This directory contains one or more subdirectories, each of which represents a separate branch of development.

/tags: This directory contains one or more subdirectories, each of which represents a specific snapshot of the codebase (i.e. a tag).

By using branches, tags, and the trunk, developers can easily manage changes to the codebase and track the history of those changes over time.

**Q 6. How CVS is different from SVN?**

|  |  |
| --- | --- |
| **CVS** | **SVN** |
| In software development, CVS is a client-oriented, free version control system. | SVN is a version control system with the most advanced, hi-tech, and latest technology for developing software. |
| Allows a user to store no other information other than files. | A file can be attached to any number of named attributes with SVN. |
| A CVS-versioned file can't be replaced with a new item of the same name without inheriting the old item's history. | Subversion can add, delete, copy, and rename files and directories. Each new file has its own history. |
| Atomic commits are not supported by concurrent versions system, but SSH (Secure Shell) is. | Apache Subversion also supports both HTTP and HTTPS. |
| The CVS process is slower than the SVN process. | SVN is far faster than CVS. Your computer runs faster since all work files are backed up. |
| When it comes to CVS, it helps to roll back any changes made to a repository. Sometimes each file should be treated independently. | With SVN, you cannot roll back changes. |

CVS and SVN are both version control systems, but there are some differences between them:

Architecture: CVS uses a client-server architecture, where the central repository resides on a server and clients interact with it to access and modify files. SVN, on the other hand, uses a client-server architecture with a centralized repository like CVS, but also supports a distributed model.

File locking: CVS uses a pessimistic locking approach, which means that when a user checks out a file for editing, it is locked so that no other user can modify it until it is checked back in. SVN, on the other hand, uses an optimistic locking approach, where users can modify files simultaneously, but conflicts are detected and resolved when changes are committed.

Branching and merging: SVN provides better support for branching and merging than CVS, especially for complex projects with multiple branches and teams working in parallel.

Security: SVN provides better security features than CVS, with support for encrypted communication and authentication.

Performance: SVN is generally faster than CVS, especially for large projects with many files and users.

Overall, SVN is considered a more modern and feature-rich version control system than CVS, with better support for branching and merging, security, and performance. However, CVS still has a user base and is often used for legacy projects that have been using it for a long time.