OPEN SOURCE TECHNOLOGIES (01CE0618) Lab Manual

Name: Dharmik Trada
Enrolment No: 92410103142
Class: EC5
Batch: B

INDEX

Lab	Program	Date	Marks	Signature
1.	Explore GitHub/GitLab for open-source projects with different licenses			
2.	Setup Git and explore commands related to Version Control System (VCS)			
3.	Create a GitHub/GitLab repository and upload sample code			
4.	Use npm / yarn / pip to install and manage packages			
5.	Deploy a simple application using Nginx / Apache			
6.	Setup Nginx to handle proxy requests and load balancing			
7.	Develop a Laravel / Django based web application			
8.	Use pytest to test a Python application			
9.	Use Selenium to create browser-based tests			
10.	Use Postman to test a sample API			
11.	Use OWASP ZAP to check security integrity			
12.	Modify any open-source desktop application			
	Contribute to any web-based open-source project			
	Use source code of an open-source web application and deploy it locally			

Experiment 1

AIM: Explore GitHub/GitLab for open-source projects with different licenses

1. **List of GitHub Licenses**
   - **No License**
   - **Apache License 2.0**
   - **GNU General Public License v3.0 (GPL-3.0)**
   - **MIT License**
   - **BSD 2-Clause "Simplified" License**
   - **BSD 3-Clause "New/Revised" License**
   - **Boost Software License 1.0**
   - **Creative Commons Zero v1.0 (CC0)**
   - **Eclipse Public License 2.0 (EPL-2.0)**
   - **GNU Affero General Public License v3.0 (AGPL-3.0)**
   - **GNU General Public License v2.0 (GPL-2.0)**
   - **GNU Lesser General Public License v2.1 (LGPL-2.1)**
   - **Mozilla Public License 2.0 (MPL-2.0)**
   - **The Unlicense**

2. Licenses and Short Description Table

License Name	Short Description
No License	Code is copyrighted; others cannot use, modify, or distribute it.
Apache License 2.0	Permissive license with patent protection; allows commercial use.
GNU General Public License v3.0 (GPL-3.0)	Strong copyleft; modified code must be open-sourced.
MIT License	Very permissive; allows almost any use with attribution.
BSD 2-Clause "Simplified" License	Permissive; minimal restrictions, similar to MIT.
BSD 3-Clause "New/Revised" License	BSD 2-Clause plus non-endorsement rule.
Boost Software License 1.0	Highly permissive; commonly used for C++ libraries.
Creative Commons Zero v1.0 (CC0)	Public domain; no restrictions or attribution needed.
Eclipse Public License 2.0 (EPL-2.0)	Weak copyleft; only modified files must be shared.
GNU Affero General Public License v3.0 (AGPL-3.0)	Strong copyleft; network/SaaS use requires source release.
GNU General Public License v2.0 (GPL-2.0)	Strong copyleft; derivatives must remain GPL-licensed.
GNU Lesser General Public License v2.1 (LGPL-2.1)	Weak copyleft; allows linking with proprietary software.
Mozilla Public License 2.0 (MPL-2.0)	File-level copyleft; balances open source and commercial use.
The Unlicense	Public domain; free use with no conditions.

3. Licenses Comparison Table

License	Type	Commercial Use	Source Code Must Be Shared?	Network (SaaS) Clause	Restriction Level
No License	Proprietary	❌ No	❌ Not allowed	❌ No	🔴 Very High
AGPL v3.0	Strong Copyleft	❌ Limited	☑ Yes (Always)	☑ Yes	🔴 Very High
GPL v3.0	Strong Copyleft	❌ Limited	☑ Yes	❌ No	🔴 High
GPL v2.0	Strong Copyleft	❌ Limited	☑ Yes	❌ No	🔴 High
LGPL v2.1	Weak Copyleft	☑ Yes	⚠ Library only	❌ No	🟠 Medium
EPL 2.0	Weak Copyleft	☑ Yes	⚠ Modified files only	❌ No	🟠 Medium
MPL 2.0	File-level Copyleft	☑ Yes	⚠ Modified files only	❌ No	🟠 Medium
Apache 2.0	Permissive	☑ Yes	❌ No	❌ No	🟢 Low
MIT	Permissive	☑ Yes	❌ No	❌ No	🟢 Low
BSD 2-Clause	Permissive	☑ Yes	❌ No	❌ No	🟢 Low
BSD 3-Clause	Permissive	☑ Yes	❌ No	❌ No	🟢 Low
Boost 1.0	Permissive	☑ Yes	❌ No	❌ No	🟢 Low
CC0	Public Domain	☑ Yes	❌ No	❌ No	⚪ None
Unlicense	Public Domain	☑ Yes	❌ No	❌ No	⚪ None

4. List of GitHub Alternatives

Platform	Type	Open Source	Best For	Key Points
GitLab	Cloud / Self-hosted	□ Yes	DevOps, CI/CD	Built-in CI/CD, issue tracking, very powerful
Bitbucket	Cloud / Self-hosted	□ No	Teams using Jira	Strong Atlassian integration
Gitea	Self-hosted	□ Yes	Lightweight Git server	Simple, fast, low resource usage
Forgejo	Self-hosted	□ Yes	Community-driven	Fork of Gitea, fully open-source
SourceForge	Cloud	□ No	Open-source hosting	Oldest platform, still used
Azure DevOps	Cloud	□ No	Enterprise projects	Git repos + pipelines + boards
Codeberg	Cloud	□ Yes	Open-source projects	Privacy-focused, EU-based
AWS CodeCommit	Cloud	□ No	AWS users	Secure Git repos inside AWS
Phabricator	Self-hosted	□ Yes	Code review	Advanced code review tools
Launchpad	Cloud	□ Yes	Ubuntu projects	Used mainly by Canonical
Pagure	Self-hosted	□ Yes	Fedora projects	Red Hat ecosystem
RhodeCode	Self-hosted	□ No	Enterprises	Git + Mercurial + SVN

5. GitHub vs GitLab Table

Feature	GitHub	GitLab	Notes / When to pick
Hosting options	Cloud (github.com) + GitHub Enterprise (self-hosted / GHES)	Cloud (gitlab.com) + robust self-hosted CE/EE	Both offer self-hosting; GitLab historically easier to run fully on-prem.
Primary audience	Massive public/open-source community, teams, enterprises	DevOps teams, CI/CD-centric orgs, enterprises	GitHub stronger for OSS visibility; GitLab for integrated dev lifecycle.
CI/CD	GitHub Actions (powerful, flexible)	Built-in GitLab CI/CD (mature, integrated)	GitHub Actions is newer but very popular; GitLab CI is feature-rich out of box.

Feature	GitHub	GitLab	Notes / When to pick
Issue tracking / Project management	Issues, Projects (Kanban), Project boards	Issues, Epics, Milestones, Roadmaps, Issue weights	GitLab offers more built-in PM features (epics, roadmaps) without plugins.
Code review workflow	Pull Requests, Reviews, Checks	Merge Requests, Approvals, Pipelines	Functionally similar; naming differs. Both support required reviewers.
Repository limits & storage	Generous on cloud plans; public repos free	Generous; self-host limits depend on infra	For large repos, consider plan and storage costs.
Container registry	GitHub Container Registry (GHCR)	Built-in Container Registry	Both provide registries; GitLab includes it by default in projects.
Package registries	GitHub Packages (npm, NuGet, Maven, etc.)	Package Registry (multiple formats)	Comparable capabilities.
Security & compliance	Code scanning, secret scanning, Dependabot, advanced for Enterprise	SAST/DAST, dependency scanning, license compliance (built-in EE)	GitLab bundles more security in self-hosted EE; GitHub has strong ecosystem tools.
Authentication / SSO / LDAP	SSO / OAuth / Enterprise SAML	SSO / LDAP / OAuth / SAML	Enterprise features comparable; self-hosted GitLab offers flexible auth options.
Integrations / Marketplace	Huge Marketplace	Integrations & built-in tools;	GitHub has broader third-

Feature	GitHub	GitLab	Notes / When to pick
	& third-party ecosystem	smaller marketplace	party ecosystem.
Interface & UX	Polished, familiar to many developers	Very capable, slightly more utilitarian	Preference-based; both are actively improved.
Import / migration	Good import tools (GitLab import, others)	Strong import/export tools, easy GitHub import	Migrating between them is straightforward with built-in importers.
Pricing model	Free for public/private repos; paid tiers and GH Enterprise	Free tier (very capable); paid tiers and self-hosted EE	Compare compute & runner costs for CI-heavy usage.
Community & discoverability	Largest OSS community — best for visibility & collaboration	Growing OSS community; used heavily in enterprises	Use GitHub to maximize discoverability of public projects.
Best fit	Public open-source, developer collaboration, broad ecosystem	Full DevOps lifecycle, on-premises DevOps, teams wanting built-in CI/CD & PM	Choose based on whether you prioritize community exposure (GitHub) or integrated DevOps features/self-hosting (GitLab).

6. Open-Source vs Proprietary vs Freeware

Aspect	Open-Source Software	Proprietary Software	Freeware
Source Code Access	☑ Available to users	❌ Not available	❌ Not available
Modification Allowed	☑ Yes	❌ No	❌ No
Redistribution	☑ Allowed (with license terms)	❌ Not allowed	⚠ Limited / Not allowed
Cost	Free (mostly)	Paid	Free
License Type	MIT, GPL, Apache, BSD, etc.	Commercial / Private license	Proprietary license
Customization	☑ High	❌ None	❌ None
Transparency	☑ Fully transparent	❌ Closed	❌ Closed
Security	Community-audited	Vendor-controlled	Vendor-controlled
Commercial Use	☑ Usually allowed	⚠ Restricted	⚠ Restricted
User Control	☑ Full	❌ Very limited	❌ Limited
Support	Community / Paid support	Official vendor support	Minimal or none
Examples	Linux, Firefox, GitLab	Windows, MS Office	Zoom (free), Skype

Experiment 2

AIM: Setup Git and explore commands related to Version Control System (VCS)

Step 1: Download from https://git-scm.com

Step 2: Configuration

git config --global user.name "dharmiktrada"

git config --global user.email "dharmik.pvt21@gmail.com"

Step 3:

C:\Users\DP203>git --version

git version 2.43.0.windows.1

C:\Users\DP203>git init

Reinitialized existing Git repository in C:/Users/DP203/.git/

C:\Users\DP203>mkdir ost

C:\Users\DP203>mkdir ost

A subdirectory or file ost already exists.

C:\Users\DP203>git init

Reinitialized existing Git repository in C:/Users/DP203/.git/

C:\Users\DP203>cd ost

C:\Users\DP203\ost>git init

Initialized empty Git repository in C:/Users/DP203/ost/.git/

```
C:\Users\DP203\ost>git status
On branch master
No commits yet
nothing to commit (create/copy files and use "git add" to track)

C:\Users\DP203\ost>echo sample.txt
sample.txt
C:\Users\DP203\ost>echo demo.py
demo.py

C:\Users\DP203\ost>git status
On branch master
No commits yet
nothing to commit (create/copy files and use "git add" to track)

C:\Users\DP203\ost>dir
Volume in drive C is OS
Volume Serial Number is E2F6-8AB0
Directory of C:\Users\DP203\ost
26-Jan-26 08:29 PM <DIR> .
26-Jan-26 08:22 PM <DIR> ..
0 File(s) 0 bytes
2 Dir(s) 17,472,352,256 bytes free

C:\Users\DP203\ost>echo sample.txt
sample.txt
```

```
C:\Users\DP203\ost>dir
Volume in drive C is OS
Volume Serial Number is E2F6-8AB0
Directory of C:\Users\DP203\ost
26-Jan-26 08:29 PM <DIR> .
26-Jan-26 08:22 PM <DIR> ..
0 File(s) 0 bytes
2 Dir(s) 17,472,425,984 bytes free

C:\Users\DP203\ost>echo Hello> sample.txt

C:\Users\DP203\ost>dir
Volume in drive C is OS
Volume Serial Number is E2F6-8AB0
Directory of C:\Users\DP203\ost
26-Jan-26 08:35 PM <DIR> .
26-Jan-26 08:22 PM <DIR> ..
26-Jan-26 08:35 PM 7 sample.txt

1 File(s) 7 bytes
2 Dir(s) 17,471,066,112 bytes free

C:\Users\DP203\ost>type sample.txt
Hello
```

```
C:\Users\DP203\ost>git commit -m "Initial Commit"
On branch master
Initial commit
Untracked files:
(use "git add <file>..." to include in what will be committed)
sample.txt
nothing added to commit but untracked files present (use "git add" to track)

C:\Users\DP203\ost>git config --global init.defaultBranch main

C:\Users\DP203\ost>git commit -m "Initial Commit"
On branch master
Initial commit
Untracked files:
(use "git add <file>..." to include in what will be committed)
sample.txt
nothing added to commit but untracked files present (use "git add" to track)

C:\Users\DP203\ost>git add sample.txt

C:\Users\DP203\ost>git commit -m "Initial Commit"
[master (root-commit) df439d9] Initial Commit
1 file changed, 1 insertion(+)
create mode 100644 sample.txt
C:\Users\DP203\ost>git branch
* master
```

```
C:\Users\DP203\ost>git branch -m master main

C:\Users\DP203\ost>git branch
* main

C:\Users\DP203\ost>git checkout main
Already on 'main'

C:\Users\DP203\ost>git checkout -b dev
Switched to a new branch 'dev'

C:\Users\DP203\ost>echo Hello from dev>> sample.txt
C:\Users\DP203\ost>git add .

C:\Users\DP203\ost>git commit -m "Update in dev"
[dev de8fe00] Update in dev
1 file changed, 1 insertion(+)

C:\Users\DP203\ost>git checkout main
Switched to branch 'main'

C:\Users\DP203\ost>git merge dev
Updating df439d9..de8fe00
Fast-forward
sample.txt | 1 +
1 file changed, 1 insertion(+)
```

```
C:\Users\DP203\ost>dir
Volume in drive C is OS
Volume Serial Number is E2F6-8AB0
Directory of C:\Users\DP203\ost
26-Jan-26 08:48 PM <DIR> .
26-Jan-26 08:44 PM <DIR> ..
26-Jan-26 08:48 PM 23 sample.txt
1 File(s) 23 bytes
2 Dir(s) 17,468,366,848 bytes free

C:\Users\DP203\ost>type sample.txt
Hello
Hello from dev

C:\Users\DP203\ost>git remote -v

C:\Users\DP203\ost>git remote add origin
https://github.com/dharmiktrada/ostdemo.git

C:\Users\DP203\ost>git remote -v
origin https://github.com/dharmiktrada/ost-dharmik (fetch)
origin https://github.com/dharmiktrada/ost-dharmikgit (push)

C:\Users\DP203\ost>git push -u origin main
remote: Invalid username or token. Password authentication is not supported for Git operations.
fatal: Authentication failed for https://github.com/dharmiktrada/ost-dharmik
```

C:\Users\DP203\ost>

C:\Users\DP203\ost>

C:\Users\DP203\ost>git push -u origin main

info: please complete authentication in your browser...

Enumerating objects: 6, done.

Counting objects: 100% (6/6), done.

Delta compression using up to 16 threads

Compressing objects: 100% (2/2), done.

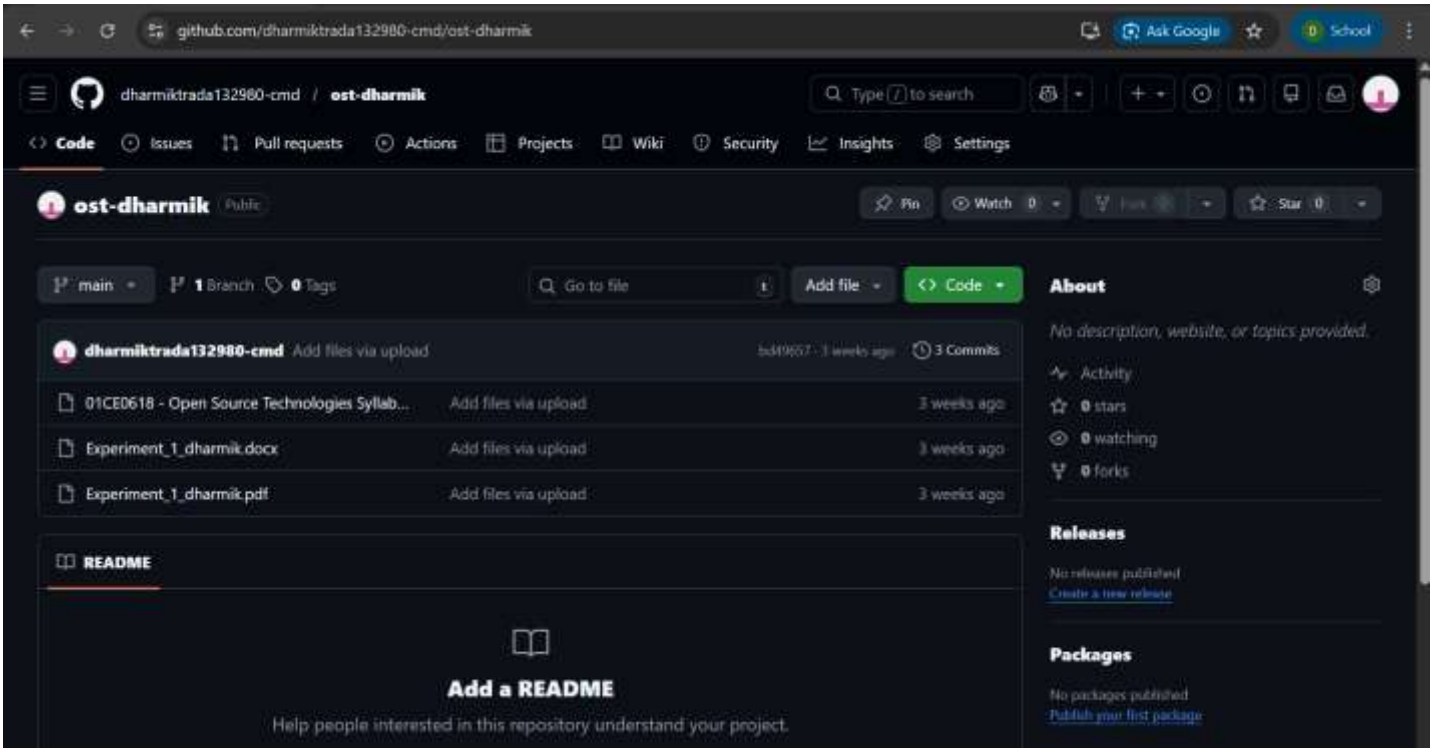
Writing objects: 100% (6/6), 452 bytes | 226.00 KiB/s, done.

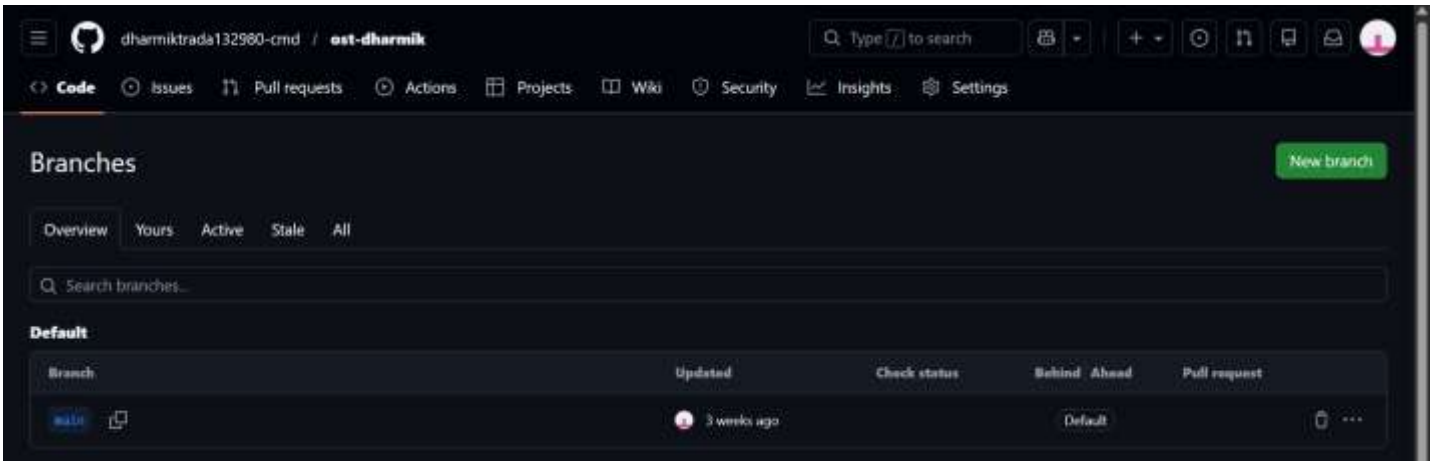
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0

To https://github.com/ dharmiktrada /ostdemo.git

* [new branch] main -> main

branch 'main' set up to track 'origin/main'.




dharmiktrada132980-cmd / ost-dharmik
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Branches
New branch
Overview
Yours
Active
Stale
All
Search branches...
Default
Branch
Updated
Check status
Behind Ahead
Pull request
3 weeks ago
Default