Karunya INSTITUTE OF TECHNOLOGY AND SCIENCES
(Declared as Deemed to be University under Sec.3 of the UGC Act, 1956)
A CHRISTIAN MINORITY RESIDENTIAL INSTITUTION
AICTE Approved & NAAC Accredited

Division of Computer Science and Engineering

School of Computer Science and Technology

Lab Manual

23CS2503 – GIT AND GITHUB FOR BEGINNERS LAB
(0:0:2:1)

Karunya INSTITUTE OF TECHNOLOGY AND SCIENCES
(Declared as Deemed to be University under Sec.3 of the UGC Act, 1956)
A CHRISTIAN MINORITY RESIDENTIAL INSTITUTION
AICTE Approved & NAAC Accredited

Division of Computer Science and Engineering

School of Computer Science and Technology

Lab Manual

23CS2503 – GIT AND GITHUB FOR BEGINNERS LAB
(0:0:2:1)

Prepared by	Verified by	Approved by
A. R. Darshika Kelin		Dr. Immanuel John Raja

INDEX

VISION

To raise world class Computer Science and Engineering professionals excelling in academics, research and providing solutions to human problems.

MISSION

- To develop professionals with strong fundamental concepts, programming and problem-solving skills with an exposure to emerging technologies.
- To promote research in the state of art technologies, providing solutions to human problems especially in the areas of Health, Water, Energy and Food.
- To develop leadership qualities with ethical values to serve the society.

PEO, PO & PSO - B.Tech. Computer Science and Engineering

Programme Educational Objectives

Graduates will

1. demonstrate the knowledge acquired to design and develop innovative software solutions.
2. Exhibit technical skills and excel as computer professionals, academicians, researchers and entrepreneurs.
3. Execute professional practice to serve the society with ethics.

Programme Outcomes

Graduates will have the ability to

1. **Engineering Knowledge:** Apply the knowledge of mathematics, natural sciences, engineering fundamentals specialization to the solution of complex engineering problems.
2. **Problem Analysis:** Identify, formulate, review research literature and analyze complex Engineering Problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.
3. **Design / Development of Solutions:** Design solutions for complex engineering problems and design system components or processes that meet specific needs with appropriate consideration for the public health and safety and the cultural, societal and environmental considerations.
4. **Conduct Investigations of Complex Problems:** Use research- based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of the information to provide valid conclusions.
5. **Modern Tool Usage:** Create, Select and Apply appropriate techniques, resources and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The Engineer and Society:** Apply Reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and Sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts and demonstrate the knowledge of and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and Team Work:** Function effectively as an individual and as a member or leader in diverse teams and in multi-disciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large such as being able to comprehend and write effective reports and

design documentation make effective presentations and give and receive clear instructions.

11. **Project Management and Finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply this to one's own work, as a member and leader in a team, to manage projects and in multi-disciplinary environments.
12. **Life – Long Learning:** Recognize the need for and have the preparation and ability to engage in independent and lifelong leaning in the broadest context of technological change.

Program Specific Outcomes (PSOs)

Graduates will have the ability to

1. Understand, analyse and develop software products and solutions using standard practices and strategies in the areas related to algorithms, system software, multimedia, web design, database, and networking for effective and efficient design of computer-based systems of varying complexity.
2. Employ modern computer languages, environments, and platforms in creating innovative career paths to be an entrepreneur, and a zest for higher studies.

PEO, PO & PSO - B.Tech. Artificial Intelligence and Data Science

Program Educational Objectives (PEOs)

Graduates will

1. demonstrate the knowledge acquired in artificial intelligence and data science, to analyse and identify the requirements, formulate and develop innovative intelligent solutions.
2. Apply artificial intelligence and data science proficiency as professionals, academicians and entrepreneurs to solve human problems.
3. practise ethical and moral values and serve the humanity with social concern.

Program Outcomes (POs)

1. **Engineering Knowledge:** Apply the knowledge of mathematics, natural sciences, engineering fundamentals specialization to the solution of complex engineering problems.
2. **Problem Analysis:** Identify, formulate, review research literature and analyze complex Engineering Problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.
3. **Design / Development of Solutions:** Design solutions for complex engineering problems and design system components or processes that meet specific needs with appropriate consideration for the public health and safety and the cultural, societal and environmental considerations.
4. **Conduct Investigations of Complex Problems:** Use research- based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of the information to provide valid conclusions.
5. **Modern Tool Usage:** Create, Select and Apply appropriate techniques, resources and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The Engineer and Society:** Apply Reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and Sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts and demonstrate the knowledge of and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and Team Work:** Function effectively as an individual and as a member or leader in diverse teams and in multi-disciplinary settings.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large such as being able to comprehend and write effective reports and design documentation make effective presentations and give and receive clear instructions.
11. **Project Management and Finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply this to one's own work, as a member and leader in a team, to manage projects and in multi-disciplinary environments.
12. **Life – Long Learning:** Recognize the need for and have the preparation and ability to engage in independent and lifelong leaning in the broadest context of technological change.

Program Specific Outcome (PSOs)

Graduates will have ability to

1. understand, analyse, design and develop intelligent systems of varying complexity using statistical and computational principles in the areas of algorithms, artificial intelligence, data science, robotics, multimedia and distributed systems
2. apply contemporary tools and techniques of artificial intelligence and data science in exhibiting skills for employability, entrepreneurship and research.

B.Tech. Computer Engineering

Program Educational Objectives (PEOs)

Graduates will

1. demonstrate the knowledge attained, to analyse the technical requirements, and develop secure innovative solutions.
2. provide solutions for real world problems through the assimilated expertise as technical professionals, academicians and entrepreneurs
3. contribute to the humanity by exercising the code of ethics and professional practice.

Program Outcomes (POs)

1. **Engineering Knowledge:** Apply the knowledge of mathematics, natural sciences, engineering fundamentals specialization to the solution of complex engineering problems.
2. **Problem Analysis:** Identify, formulate, review research literature and analyze complex Engineering Problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.
3. **Design / Development of Solutions:** Design solutions for complex engineering problems and design system components or processes that meet specific needs with appropriate consideration for the public health and safety and the cultural, societal and environmental considerations.
4. **Conduct Investigations of Complex Problems:** Use research- based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of the information to provide valid conclusions.
5. **Modern Tool Usage:** Create, Select and Apply appropriate techniques, resources and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The Engineer and Society:** Apply Reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and Sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts and demonstrate the knowledge of and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and Team Work:** Function effectively as an individual and as a member or leader in diverse teams and in multi-disciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large such as being able to comprehend and write effective reports and

design documentation makes effective presentations and give and receive clear instructions.

11. **Project Management and Finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply this to one's own work, as a member and leader in a team, to manage projects and in multi-disciplinary environments.
12. **Life – Long Learning:** Recognize the need for and have the preparation and ability to engage in independent and lifelong leaning in the broadest context of technological change.

Program Specific Outcome (PSOs)

Graduates will have ability to

1. understand, analyse and develop products and software solutions using standard practices and strategies in the areas of algorithms, communication networks, real time systems, system administration, robotics, virtual reality, artificial intelligence, data science and cyber security.
2. employ modern programming languages and platforms in the field of computing to develop skills for employability, entrepreneurship, and research.

B.Tech. Computer Science and Engineering (Artificial Intelligence)

Program Educational Objectives (PEOs)

Graduates will

1. demonstrate the technical knowledge attained, to investigate and identify the requirements, and develop intelligent solutions.
2. exhibit proficiency as artificial intelligence professionals, academicians and entrepreneurs to develop solutions for real world problems.
3. serve the society by adopting ethical and moral values.

Program Outcomes (POs)

1. **Engineering Knowledge:** Apply the knowledge of mathematics, natural sciences, engineering fundamentals specialization to the solution of complex engineering problems.
2. **Problem Analysis:** Identify, formulate, review research literature and analyze complex Engineering Problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.
3. **Design / Development of Solutions:** Design solutions for complex engineering problems and design system components or processes that meet specific needs with appropriate consideration for the public health and safety and the cultural, societal and environmental considerations.
4. **Conduct Investigations of Complex Problems:** Use research- based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of the information to provide valid conclusions.
5. **Modern Tool Usage:** Create, Select and Apply appropriate techniques, resources and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The Engineer and Society:** Apply Reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and Sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts and demonstrate the knowledge of and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and Team Work:** Function effectively as an individual and as a member or leader in diverse teams and in multi-disciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large such as being able to comprehend and write effective reports and design documentation make effective presentations and give and receive clear instructions.

11. **Project Management and Finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply this to one's own work, as a member and leader in a team, to manage projects and in multi-disciplinary environments.
12. **Life – Long Learning:** Recognize the need for and have the preparation and ability to engage in independent and lifelong leaning in the broadest context of technological change.

Program Specific Outcome (PSOs)

Graduates will have ability to

1. understand, analyse and develop innovative solutions using the principles of artificial intelligence, algorithms, web technology, data science, networking, and software engineering.
2. use artificial intelligence tools and languages to exhibit skills in employability, entrepreneurship and research.

B.Tech. Computer Science and Engineering (Artificial Intelligence and Machine Learning)

Program Educational Objectives (PEOs)

Graduates will

1. demonstrate the technical knowledge acquired, to analyse the requirements, identify the technical specifications, design and develop intelligent solutions.
2. exhibit the expertise as artificial intelligence and machine learning professionals, academicians and entrepreneurs.
3. serve the society with social concern by adopting code of ethics and professional practice.

Program Outcomes (POs)

1. **Engineering Knowledge:** Apply the knowledge of mathematics, natural sciences, engineering fundamentals specialization to the solution of complex engineering problems.
2. **Problem Analysis:** Identify, formulate, review research literature and analyze complex Engineering Problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.
3. **Design / Development of Solutions:** Design solutions for complex engineering problems and design system components or processes that meet specific needs with appropriate consideration for the public health and safety and the cultural, societal and environmental considerations.
4. **Conduct Investigations of Complex Problems:** Use research- based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of the information to provide valid conclusions.
5. **Modern Tool Usage:** Create, Select and Apply appropriate techniques, resources and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The Engineer and Society:** Apply Reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and Sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts and demonstrate the knowledge of and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and Team Work:** Function effectively as an individual and as a member or leader in diverse teams and in multi-disciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large

such as being able to comprehend and write effective reports and design documentation make effective presentations and give and receive clear instructions.

11. **Project Management and Finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply this to one's own work, as a member and leader in a team, to manage projects and in multi-disciplinary environments.
12. **Life – Long Learning:** Recognize the need for and have the preparation and ability to engage in independent and lifelong leaning in the broadest context of technological change.

Program Specific Outcome (PSOs)

Graduates will have ability to

1. understand, analyse and develop innovative solutions using the principles of artificial intelligence, machine learning, data science, algorithms, web technologies and communication networks.
2. apply artificial intelligence and machine learning tools and languages to exhibit skills in employability, entrepreneurship and research.

Course Code	GIT AND GITHUB FOR BEGINNERS	L	T	P	C
23CS2503		0	0	2	1

Course Objectives

Enable the students to:

1. Collaborate among the project community.
2. Manage notifications for project events.
3. Create branches to manage work concurrently.

Course Outcomes

The student will be able to:

1. Reason the significance of version control in coding and collaboration
2. Compare the different states in Git and between branches and commits
3. Illustrate Git installation and run on local machine
4. Use and interact with GitHub
5. Manage and update files inside and outside Git
6. Collaborate through remote repositories

List of Exercises 30 hours

1. Installation of Git and creating repositories.
2. Working with Remote Repositories on local machines.
3. Version controlling in software application development
4. GitHub issues
5. Working with collaborative repository management using Git.
6. Implementation of Core Review in Git

Exercises from the above list will be approved by the HoD during the start of the semester.

Recommended by Board of Studies

Approved by Academic Council

LIST OF EXERCISES

Ex.No: 1	INSTALLATION OF GIT AND CREATING REPOSITORIES.

Introduction of GIT

Git is one of the ways of implementing the idea of version control. It is Distributed Version Control System.

Installing GIT

Before you start using Git, you have to make it available on your computer.

it's already installed, it's probably a good idea to update to the latest version. You can either install it as a package or via another installer, or download the source code and compile it yourself.

Installing on Windows

There are also a few ways to install Git on Windows. The most official build is available for download on the Git website. Just go to https://git-scm.com/download/win and the download will start automatically

To get an automated installation you can use the Git Chocolatey package.

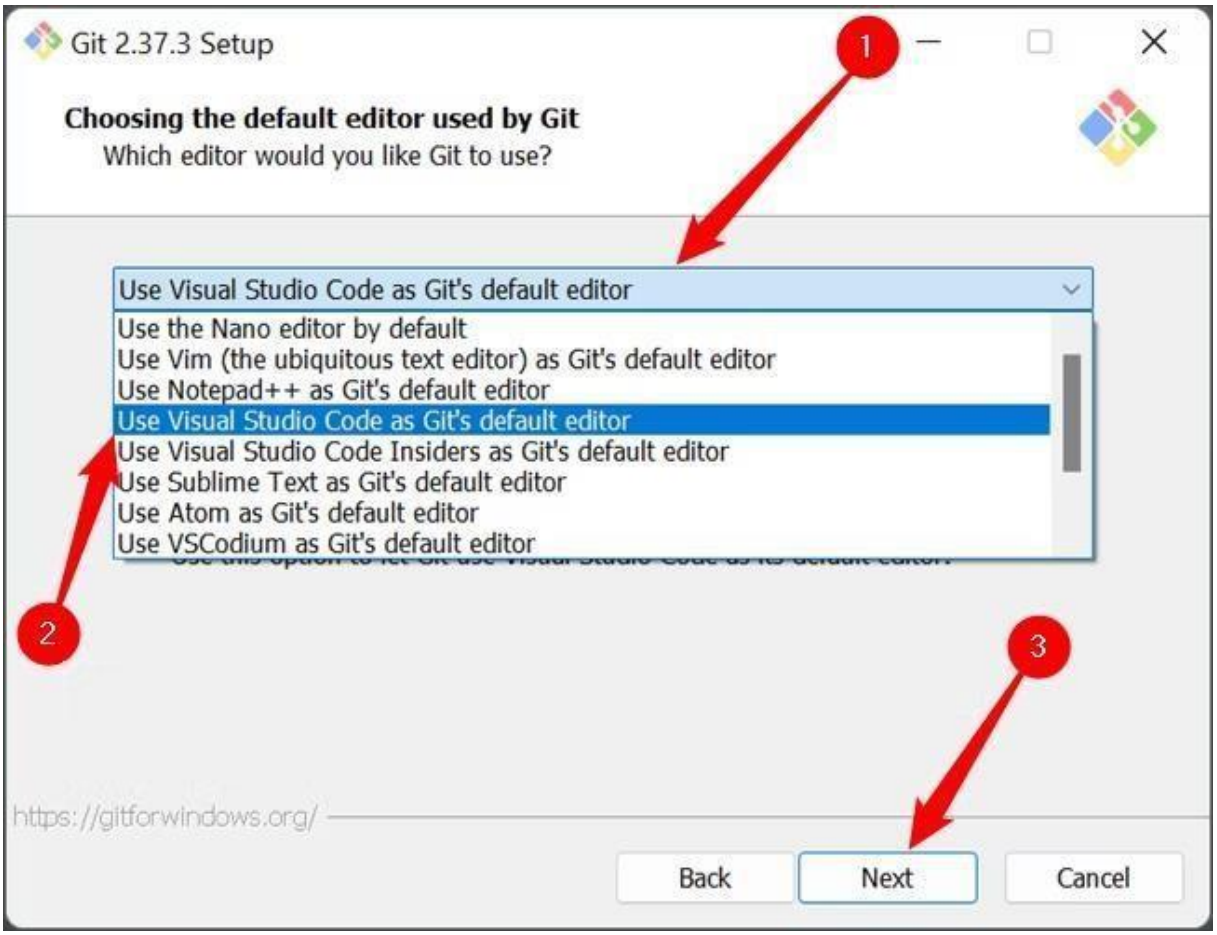
The easiest way to get Git is to download the executable from the Git website.

Click "64-bit Git for Windows Setup" to start the download, and then wait a moment — the download is only about 50 megabytes, so it shouldn't take very long.

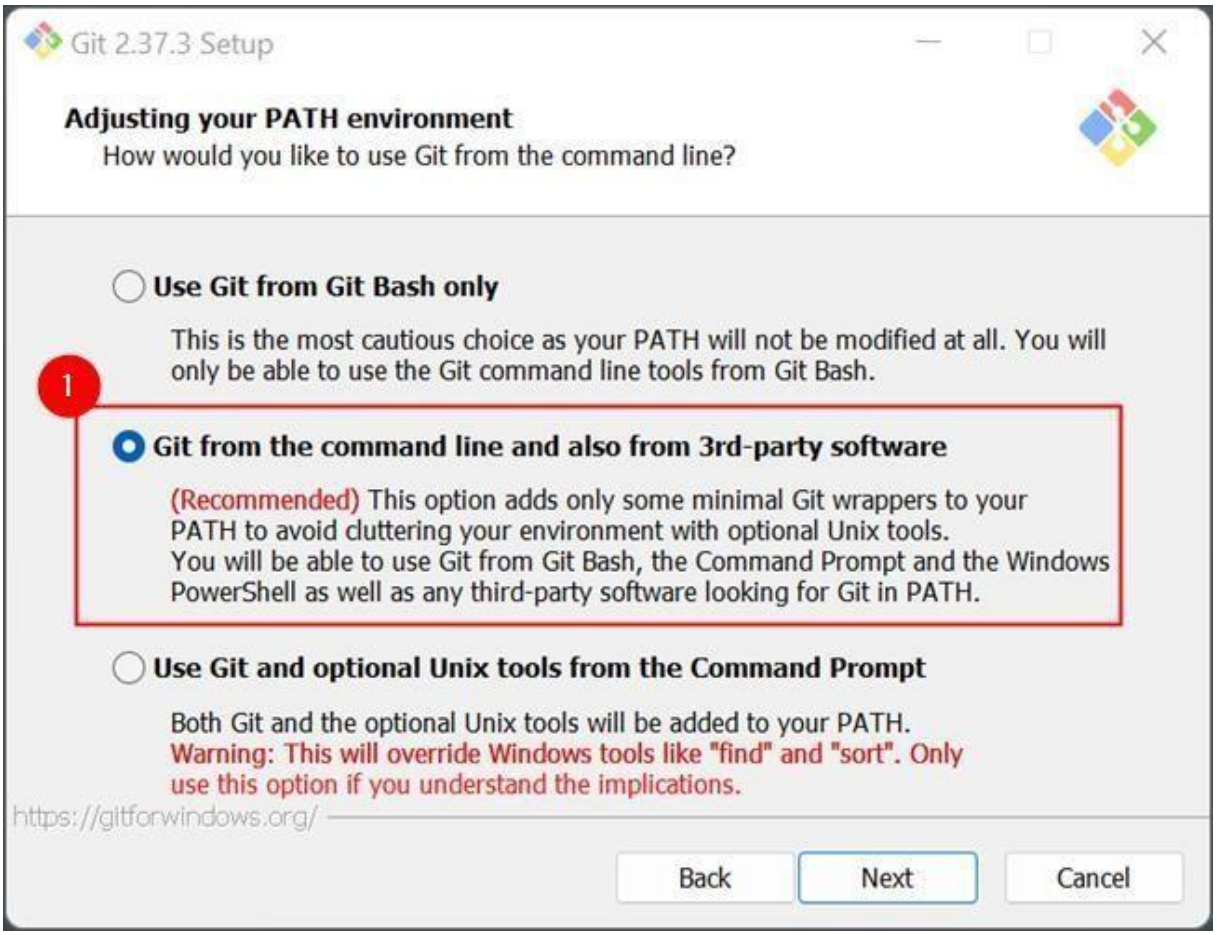
Double-click the executable you just downloaded, then click "Next" to move through the installation prompts.

The first is the text editor Git will use. The default selection is Vim. Vim is ubiquitous and a hallmark of command-line interfaces everywhere but learning to use its idiosyncratic

commands can be daunting. You should probably pick something else instead, like Visual Studio Code, Sublime, NotePad++, or any other plain text editor you like.



The second is the way Git integrates itself into your PC's PATH. Make sure that the "Git From The Command Line And Also From 3rd-Party Software" is selected.



Click through the remaining options, and wait for everything to finish downloading. The time requires to download everything will vary depending on what you chose to install. The default selection results in a download that is about 270 megabytes.

Managing Private and Public RepositoryMaking a

repository Private

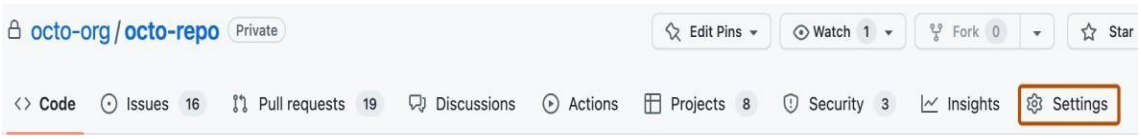
- GitHub will detach public forks of the public repository and put them into a new network. Public forks are not made private.
- If you're using GitHub Free for personal accounts or organizations, some features won't be available in the repository after you change the visibility to private. Any published GitHub Pages site will be automatically unpublished. If you added a customdomain to the GitHub Pages site, you should remove or update your DNS records before making the repository private, to avoid the risk of a domain takeover.
- GitHub will no longer include the repository in the GitHub Archive Program.
- GitHub Advanced Security features, such as code scanning, will stop working.

Making a repository Public

- GitHub will detach private forks and turn them into a standalone private repository.
- If you're converting your private repository to a public repository as part of a move toward creating an open source project.
- Once your repository is public, you can also view your repository's community profile to see whether your project meets best practices for supporting contributors.
- The repository will automatically gain access to GitHub Advanced Security features.

Changing a repository's Visibility

1. On GitHub.com, navigate to the main page of the repository.
2. Under your repository name, click **Settings**. If you cannot see the "Settings" tab, select the dropdown menu, then click **Settings**.



3. In the "Danger Zone" section, to the right of to "Change repository visibility", click **Change visibility**.
4. Select a visibility.
5. To verify that you're changing the correct repository's visibility, type the name of the repository you want to change the visibility of.
6. Click **I understand, change repository visibility**.

Pushing changes to the repository

1. On Github.com, ensure your repository is public.
2. Clone the repository to your local machine.
   i. Open a terminal on your local machine.
   ii. Use the following command to clone the repository to your local machine. Replace 'your-username' and 'your-repo-name' with your GitHub username and repository name
   git clone https://github.com/your-username/your-repo-name.git
   iii. Move into the newly cloned repository
   cd your-repo-name
3. Make a change to the README file (Create a README.md file if it does not exist).
4. Commit the change.
   i. Add the modified README file to the staging area using the following command:
   git add README.md
   ii. Commit the changes with a meaningful commit message:
   git commit -m "Update README file"
5. Push it to the GitHub repository using the following command. If your main branch is called 'master', then replace 'main' with 'master'.
   git push origin main

EX. NO: 2	WORKING WITH REMOTE REPOSITORIES ON LOCAL MACHINE.

To set up and configure Git on your local machine, you can follow these steps:

Step 1: Install Git

- Download the Git installer from the official Git website.

- Run the installer and follow the step-by-step instructions in the setup wizard.

- Make sure to select the appropriate options during the installation process.

Step 2: Configure Git

- Open a terminal or command prompt.

- Set your username and email address by running the following commands:

git config --global user.name "Your Name"

git config --global user.email "your.email@example.com"

Replace "Your Name" with your desired username and "your.email@example.com" with your email address.

Step 3: Set up the default branch name (optional)

- By default, Git creates a branch called "master" when you initialize a new repository with `git init`.

- If you prefer to use a different name for the initial branch, you can set it using the following command:

git config --global init.defaultBranch main

This command sets the default branch name to "main". Replace "main" with your preferred branch name if desired.

Step 4: Verify your Git configuration

- To check your Git configuration settings, you can use the following command:

git config --list

This command will display a list of your Git configuration settings, including your username and email address.

These steps should help you set up and configure Git on your local machine. Remember to replace "Your Name" and "your.email@example.com" with your own information.

Working with Remote Repository

When working with remote repositories in Git, there are several commands and actions you can perform.

Cloning a Remote Repository

- The **git clone** command is used to create a local copy of a remoterepository on your machine.

To clone a remote repository, use the following command

git clone <remote-url>

- Replace **<remote-url>** with the URL of the remote repository.
  - For example, to clone a repository hosted on GitHub, you can use

git clone https://github.com/username/repository.git

Adding a Remote Repository

- If you have an existing local repository and want to connect it to a remote repository, you can use the git remote add command.
- To add a remote repository, use the following command:

git remote add <remote-name> <remote-url>

- Replace <remote-name> with a name for the remote repository (e.g.,"origin") and <remote-url> with the URL of the remote repository.
  - For example

git remote add origin https://github.com/username/repository.git

Pushing Changes to a Remote Repository

After making changes to your local repository, you can push those changes tothe remote repository using the git push command.

To push changes to a remote repository, use the following command:

git push <remote-name> <branch-name>

Replace <remote-name> with the name of the remote repository (e.g., "origin") and <branch-name> with the name of the branch you want to push.

For example:

git push origin main

Pulling Changes from a Remote Repository

- To update your local repository with the latest changes from the remote repository, you can use the git pull command.
- To pull changes from a remote repository, use the following command:

git pull <remote-name> <branch-name>

- Replace <remote-name> with the name of the remote repository (e.g., "origin") and <branch-name> with the name of the branch you want to pull.
- For example

git pull origin main

These are some basic commands for working with remote repositories in Git.Remember to replace <remote-url>, <remote-name> and <branch-name> with the appropriate values for your specific repository.

EX. NO: 3	VERSION CONTROL IN SOFTWARE APPLICATION DEVELOPMENT

What is a "version control system"?

Version control systems are a category of software tools that helps in recording changes made to files by keeping a track of modifications done in the code.

Types of Version Control Systems:

- Local Version Control Systems
- Centralized Version Control Systems
- Distributed Version Control Systems

Three important steps of version control:

- **git add** changed files to version control tracking.
- **git commit** the changed files to create a unique snapshot of the local repository.
- **git push** those changed files from the local copy of a repository to the cloud

Check the Status of Changes Using GIT Status

Once you start working, you can use the **git status** command to check what changes are being identified by **git**.

To practice working with this command, use the **terminal** to navigate to your git practice repository:

$ cd practice-git-skillz

Next, run git status.

$ git status

On branch main
Your branch is up to date with

'origin/main'.nothing to commit,

working tree clean

Notice that when you run git status it returns: **working tree clean**. This means that there areno changes to any files in your repo - YET.

Next, open and make a small change to the README.md file in a text editor. Then, run the command git status to check that changes have been made to your file(s).

git status
On branch main
Your branch is up-to-date with
'origin/main'.Changes not staged for
commit:

```
        modified:  README.md

no changes added to commit (use "git add" and/or "git commit -a")
```

The output from the git status command above indicates that you have modified a file (e.g. README.md) that can be added to version control.

Important Git Commands

These two commands make up the bulk of many workflows that use **git** for version control:

- git add: takes a modified file in your working directory and places the modified version in a staging area for review.
- git commit: takes everything from the staging area and makes a permanent snapshot of the current state of your repository that has a unique identifier.

Add Changed Files Using git add

After making changes, you can add either an individual file or groups of files to version control tracking. To add a single file, run the command:

```
git add file-name.extension
```

For example, to add the README.md file, you would use:

```
git add README.md
```

You can also add all of the files that you have edited at the same time using:

```
git add .
```

Commit Changed Files Using git commit

Once you are ready to make a snapshot of the current state of your repository (i.e. move changes from staging area), you can run git commit. The git commit command requires a commit message that describes the snapshot (i.e. changes) that you made in that commit. A commit message should outline what changed and why. These messages:

1. help collaborators and your future self understand what was changed and why.
2. allow you and your collaborators to find (and undo if necessary) changes that were previously made.

When you are not committing a lot of changes, you can create a short one line commit message using the -m flag as follows:

```
git commit -m "Update title and author name in homework for week 3"
```

Creating branches

To keep track of changes to this file using **git**, you need to:

1. Clone the repository.
2. Move into the cloned repository
3. Create a new branch using the command (replace *feature-branch* with your desired branch name).

   git checkout -b feature-branch
4. Make modifications to files in your project.
5. Use git add to add the changes to the staging area
6. Commit the changes with a meaningful message

Merging branches

To keep track of changes to this file using **git**, you need to:

1. Switch back to the main branch.

   git checkout main
2. Merge the branch into the main branch

   git merge feature-branch
3. Resolve conflicts (if necessary)
   i. Open the conflicting files and resolve the conflicts manually.
   ii. After resolving conflicts, add the changes to the staging area and commit:

   git add .
   git commit -m "Merge feature-branch into main"
4. Push changes to github using the following command.

   git push origin main

Ex. No: 4	GITHUB ISSUES

Github Issues:

GitHub Issues is a robust project management feature integrated into the GitHub platform, designed to streamline collaboration, communication, and issue tracking within software development projects.
It serves as a centralized hub for tracking tasks, enhancements, bug reports, and other project-related activities within a GitHub repository. It provides a structured and organized way for development teams to manage their workflow, ensuring that issues are identified, discussed, and resolved efficiently.

Steps to Create and Address GitHub Issues

1. Create a new issue in your repository:
    i. Go to your GitHub repository.
    ii. Click on the "Issues" tab.
    iii. Click the "New issue" button.
    iv. Fill in the issue title and description.
    v. Optionally, assign the issue to a collaborator, apply labels, and set milestones.
2. Assign the issue to a collaborator:
    i. Within the issue, use the "Assignees" section to assign the issue to a collaborator.
    ii. The assigned collaborator will receive notifications about the issue.
3. Label the issue with appropriate tags:
    i. Apply labels to categorize and prioritize issues.
    ii. Create and use labels like "bug," "feature," or any relevant labels for your project.
    iii. Labels can be added when creating the issue or edited later.
4. Close the issue using a commit message:
    i. Make changes to the codebase to address the issue.
    ii. In your local repository, commit the changes with a commit message that references the issue number:
    git commit -m "Fix #1: Resolve issue with feature X"
    Replace "1" with the actual issue number.
    iii. Push the changes to GitHub:
    git push origin main
    The issue will be automatically closed when the commit is pushed.

5. Optional: Comment and Discuss:
    i. Encourage collaboration by adding comments to the issue.
    ii. Mention collaborators using @username to notify them.
    iii. Discuss the issue, provide additional information, or ask for feedback.

Ex.No: 5	WORKING WITH COLLABORATIVE REPOSITORY MANAGEMENT USING GIT

Working with collaborative repository management using Git involves collaborating with other developers and effectively managing changes to ashared codebase. Here are some key practices and concepts to consider:

Forking a Repository

- When you want to contribute to a project hosted on a remote repository, it's common to start by forking the repository.

- Forking creates a personal copy of the repository under your GitHub account or another hosting platform, where you can freely make changes without affecting the original repository.

- To fork a repository on GitHub, navigate to the repository's page and click the "Fork" button.

Cloning a Forked Repository

- After forking a repository, you need to clone the forked repository to your local machine to start making changes.

- Use the `git clone` command, as explained in a previous answer, to clone theforked repository.

Adding an Upstream Remote

- The original repository that you forked is known as the "upstream" repository.It represents the source of truth for the project.

- To synchronize your forked repository with the latest changes from theupstream repository, you can add an "upstream" remote.

- Use the `git remote add` command to add the upstream remote URL. Forexample:

git remote add upstream <upstream-url>

Keeping Your Forked Repository Up to Date

- To incorporate the latest changes from the upstream repository into your forked repository, follow these steps:

- Fetch the latest changes from the upstream remote: **`git fetch upstream`**

- Checkout your local main branch: **`git checkout main`**

- Merge the changes from the upstream/main branch into your local mainbranch: **`git merge upstream/main`**

- Push the updated main branch to your forked repository: **`git push originmain`**

Creating Branches for Collaborative Work

- When working on a collaborative project, it's common to create branches for new features, bug fixes, or other changes.

- Use the `git branch` command to create a new branch: **`git branch <branch-name>`**

- Switch to the newly created branch using `git checkout`: **`git checkout** <branch-name>`

- Alternatively, you can create and switch to a new branch in one command**: `gitcheckout -b <branch-name>`**

Pushing Changes and Creating Pull Requests

- Once you've made changes on a branch, you can push the branch to yourforked repository using **`git push origin <branch-name>`.**

- After pushing the branch, you can create a pull request on the upstream repository to propose your changes for merging into the main codebase.

- On the upstream repository's page, find the "Pull requests" section and clickthe "New pull request" button.

- Select the appropriate branches for the base (usually main) and compare (your branch) branches.

- Provide a title and description for your pull request, then click "Create pullrequest" to submit it.

Reviewing and Merging Pull Requests

- Collaborators or maintainers of the upstream repository can review and comment on your pull request.

- They may request changes or provide feedback before merging the changes.

- Once the pull request is approved, it can be merged into the main branch of theupstream repository.

EX. NO: 6	IMPLEMENTATION OF CORE REVIEW IN GIT

Introduction

Git is a powerful tool for version control, but it's also a crucial tool for collaborating with remote teams. When working remotely, effective communication and collaboration are key to ensure that your team is working together towards a common goal.

Use a shared repository

Using a shared repository is the foundation of effective collaboration withGit. By hosting your repository on a remote server, you can give your team members access to the latest version of your code, regardless of their location.

Create a new repository: **git init**

Clone a repository: **git clone [respository URL]**

Add a remote repository: **git remote add [name] [repository URL]**

Fetch changes from a remote repository: **git fetch**Pull

changes from a remote repository**: git pull** Push

changes to a remote repository: **git push Use**

branching and merging

Branching and merging are powerful features of Git that allow you to work on multiple versions of your code simultaneously, without interfering with each other's work. By using branching and merging effectively, your team can work on different features or bug fixes in parallel, without causing conflicts.

Create a new branch: **git branch [name]**Switch

to a branch: **git checkout [name]**Merge a

branch: **git merge [name]** Resolve merge

conflicts: **git mergetool** Delete a branch: **git**

branch -d [name]

Use Pull requests

Pull requests are a great way to review and merge changes from differentteam members before they are merged into the main codebase. By using pull requests, you can ensure that your code is reviewed and tested before it is merged into the main branch, which can help prevent bugs and other issues.

Create a new pull request: **git request-pull [branch] [repository URL]**

Review and merge a pull request**: git pull-request**

Close a pull request: **git request-pull -C [branch] [repository URL]**

Use Issue Tracking

Issue tracking is a great way to keep track of bugs, feature requests, and other issues that need to be addressed in your code. By using an issue tracking system like GitHub Issues, you can assign tasks to team members, track progress, and ensure that all issues are addressed in a timely manner.

Create a new issue: **git commit -m "[issue number] [commit message]"**

Assign an issue to a team member: **git assign [username]**

Clone an issue: **git clone [issue number]** Reopen an

issue: **git reopen [issue number]Communicate**

Effectively

Effective communication is key to collaboration, especially in a remote team setting. Use tools like Slack or Microsoft Teams to stay in touch with yourteam members, and use video calls or screen sharing to discuss code changes or work on problems together.

Start a video call: **git video-call [username]**

Share your screen**: git screen-share**

Send a message: **git message [username] [message]**

Use Git hooks

Git hooks are scripts that Git runs automatically at certain points in the Gitworkflow. You can use Git hooks to automate repetitive tasks, enforce coding standards, or perform other tasks that are important to your team's workflow.

Install a git hook: **git init [hook name]**

Write a git hook script: **nano.git/hooks/[hook name]**

Make the Git hook script executable: **chmod +x .git/hooks/[hook name]**

Use Git submodules

Git submodules are repositories that are embedded inside other repositories. You can use Git submodules to manage dependencies, or to includeshared code in multiple projects.

Add a Git submodule: **git submodule add [repository URL]**

Update a Git submodule: **git submodule update**

Remove a Git submodule: **git submodule deinit [submodule path]**

Use Git aliases

Git aliases are shortcuts for commonly used Git commands. You can useGit aliases to save time and improve your productivity when working with Git.

Set up a Git alias: **git config –global alias.[alias name] '[Git command]'**

Use a Git alias: **git [alias name]**