GitHub Student Handbook

A complete practical guide for beginners and future developers

19. Pulling Updates from GitHub

Module 4 — Branching and Merging

20. Understanding Branches
21. Creating and Switching Branches
22. Working on Feature Branches
23. Merging Branches
24. Resolving Merge Conflicts

Module 5 — Team Collaboration

25. Forking Repositories
26. Creating Pull Requests
27. Code Review Basics
28. Handling Review Feedback
29. Using Issues for Bug Tracking
30. Introduction to Project Boards

Module 6 — Professional GitHub Usage

31. Writing a Strong README File
32. Documenting Your Projects
33. Maintaining Clean Commit History
34. Optimizing Your GitHub Profile
35. Using GitHub as Your Developer Portfolio

Module 7 — Advanced but Practical

Stuff companies love but beginners rarely learn properly

Module 1 — Foundations of Version Control

1. What is Version Control?

> *“Version control is the memory of your project.”*

Concept Explanation

When you write a program, you rarely finish it in one attempt.
You keep changing it — fixing bugs, adding features, removing mistakes, improving structure.

After a few days, questions appear:

- Which version was working?
- What exactly did I change yesterday?
- Can I go back if today’s code breaks?

Version Control is a system that:

- records every change made to your files
- stores the complete history of the project
- allows you to return to any previous version
- helps multiple people work safely on the same code

Think of it as the **save system of a video game**.
If today’s save is broken, you simply load yesterday’s.

Why This Matters in Real Work

In professional development:

- dozens of developers work on the same project
- mistakes are unavoidable
- features are developed simultaneously

Without version control:

- people overwrite each other’s work
- bugs destroy stable versions
- teamwork becomes impossible

Version control creates **order, safety, and accountability**.

Key Terms

Term	Meaning
Repository	Main project folder with full history
Commit	Saved checkpoint
History	All previous versions
Revert	Go back to an older version
Branch	Independent line of development

Example — Without Version Control

```
project_v1
project_v2
project_v3
final_project
final_project_fixed
final_project_really_fixed
```

With Version Control

```
Project
 └── History
      ├── Version 1
      ├── Version 2
      ├── Version 3
      └── Version 4
```

One folder. Full safety.

Common Mistakes

- Keeping many backup folders
- Editing final versions directly
- Losing track of working code

Practice

1. Write two reasons why version control is important.
2. Describe a situation where version control could have saved your work.

2. What is Git?

> *“Git is the engine of version control.”*

Concept Explanation

Git is a software that runs on your computer and performs version control.

Git allows you to:

- track file changes
- save checkpoints
- create separate work branches
- merge work safely
- recover lost versions
- work offline

Git is the most widely used version control system in the world.

Why This Matters

Every serious software company uses Git.
Knowing Git is not optional — it is essential.

Core Concepts

Concept	Meaning
Working Directory	Your project folder
Staging Area	Files prepared for commit
Repository	Where history is stored
Commit	Saved snapshot

Basic Git Flow

```
Edit → Stage → Commit → Repeat
```

Common Mistakes

- Making huge commits
- Forgetting to commit regularly
- Not checking status before committing

Practice

1. Research why Git was created.
2. Explain the difference between saving a file and committing.

3. What is GitHub?

> *“GitHub is the home of your projects on the internet.”*

Concept Explanation

GitHub is a website that stores Git repositories online and enables collaboration.

It provides:

- online backup
- team collaboration
- pull requests
- issue tracking
- project management
- portfolio building

Your GitHub profile becomes your **technical identity**.

Why This Matters

Companies host products on GitHub.
Recruiters judge your skills using GitHub.
Developers collaborate using GitHub.

Git vs GitHub

Git	GitHub
Software	Online platform
Local	Cloud
Tracks changes	Hosts projects
Manages history	Enables teamwork

Common Mistakes

- Confusing Git with GitHub
- Using GitHub without understanding Git

Practice

1. Create a GitHub account.
2. Explore trending repositories.

4. Git vs GitHub

Git is the engine.
GitHub is the collaboration platform.

Both are required.

5. Why Companies Use Git and GitHub

Companies require:

- stability
- collaboration
- accountability
- safety
- history tracking

Git and GitHub provide all of this.

6. How Teams Collaborate Using GitHub

Teams collaborate using:

- branches
- pull requests
- code reviews
- issues

Direct edits to main branch are avoided.

Module 2 — Setup and First Contact

7. Creating a GitHub Account

Visit **https://github.com**
Create an account
Verify your email
Choose a professional username

Your username becomes part of your career.

8. Installing Git

Download Git from:
https://git-scm.com

Install with default settings.

Verify installation:

```
git --version
```

Expected output:

```
git version 2.x.x
```

9. Configuring Git

Set your identity:

```
git config --global user.name "Your Name"
git config --global user.email "you@email.com"
```

Check configuration:

```
git config --list
```

10. Creating Your First Repository

Create repository on GitHub.
Add README file.

Choose public.

11. Cloning a Repository

```
git clone https://github.com/username/project.git
```

Expected output:

```
Cloning into 'project'...
Receiving objects...
Resolving deltas...
```

12. Connecting VS Code with GitHub

Open project in VS Code.
Open Source Control panel.
Sign in to GitHub.

13. Understanding Project Structure

```
project/
 ├── README.md
 ├── src/
 ├── docs/
 └── .gitignore
```

14. Checking File Status

```
git status
```

15. Staging Files

```
git add .
```

16. Making Commits

```
git commit -m "Initial commit"
```

17. Writing Professional Commit Messages

Good commits are:

- short
- descriptive
- written in present tense

18. Pushing Changes

```
git push origin main
```

19. Pulling Updates

```
git pull origin main
```