**Chapter 5**

**Change Management**

**5.1 Change Management (Version Control Using Git)**

**Change management is a critical process in software development to ensure that all changes made to the source code are tracked, coordinated, and managed efficiently. In our project, we used Git as the version control system to handle change management. Git allows multiple developers to work on the same project simultaneously without overwriting each other’s changes.**

**In the development of our Kindergarten Website, change management played a vital role in organizing and tracking the progress of our project. To efficiently manage updates, modifications, and collaboration among team members, we used Git, a distributed version control system.**

**Work collaboratively on different modules (e.g., registration form, attendance system, parent communication).**

**Create separate branches for features like student registration, admin login, and fee management without affecting the main version.**

**Maintain project history, so we could review or roll back to previous versions when necessary.**

**Avoid conflicts through effective branch and merge strategies.**

**This approach ensured that our development process was organized, safe, and collaborative, especially as we worked on both frontend and backend parts of the website**

**Basic Workflow in Git**

**The basic Git workflow involves a series of steps that help developers track and manage changes to their project in an organized way. Here’s how it works:**

**1. Start a Project**

**You begin by creating a new Git repository or copying (cloning) an existing one from a remote platform like GitHub.**

**2. Make Changes**

**You make updates to your project files, such as editing or adding HTML, CSS, PHP, or JavaScript files for your kindergarten website.**

**3. Check the Status**

**You check which files have been changed, added, or deleted since the last save (commit). This helps you keep track of your work.**

**4. Stage the Changes**

**You select the specific changes you want to include in your next save. This is called staging, and it prepares the changes for the next step.**

**5. Commit the Changes**

**You save the staged changes in Git with a message describing what was done (e.g., “Added contact form” or “Fixed layout issue”).**

**6. Push Changes to Remote Repository**

**After committing your changes locally, you upload them to a remote repository like GitHub. This makes your work accessible to others and creates an online backup.**

**7. Pull Latest Updates**

**Before starting new work, you download any new changes made by other team members to ensure your version is up to date.**

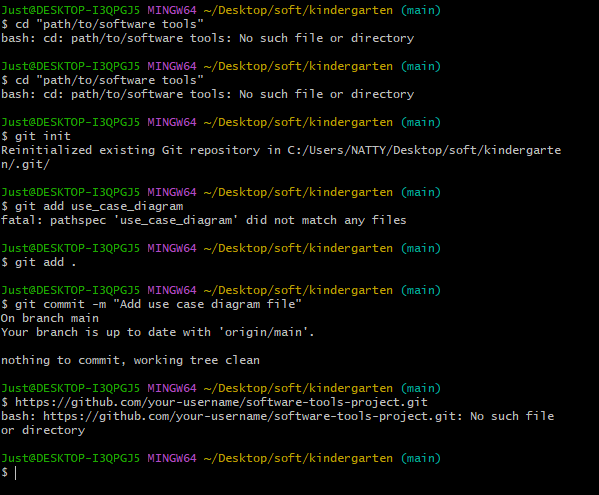
**5.2 Steps and Tools Used to Implement Git in Our Project**

**To effectively manage our project using Git, we followed these steps and used the following tools:**

**Step 1: Initialize Git Repository**

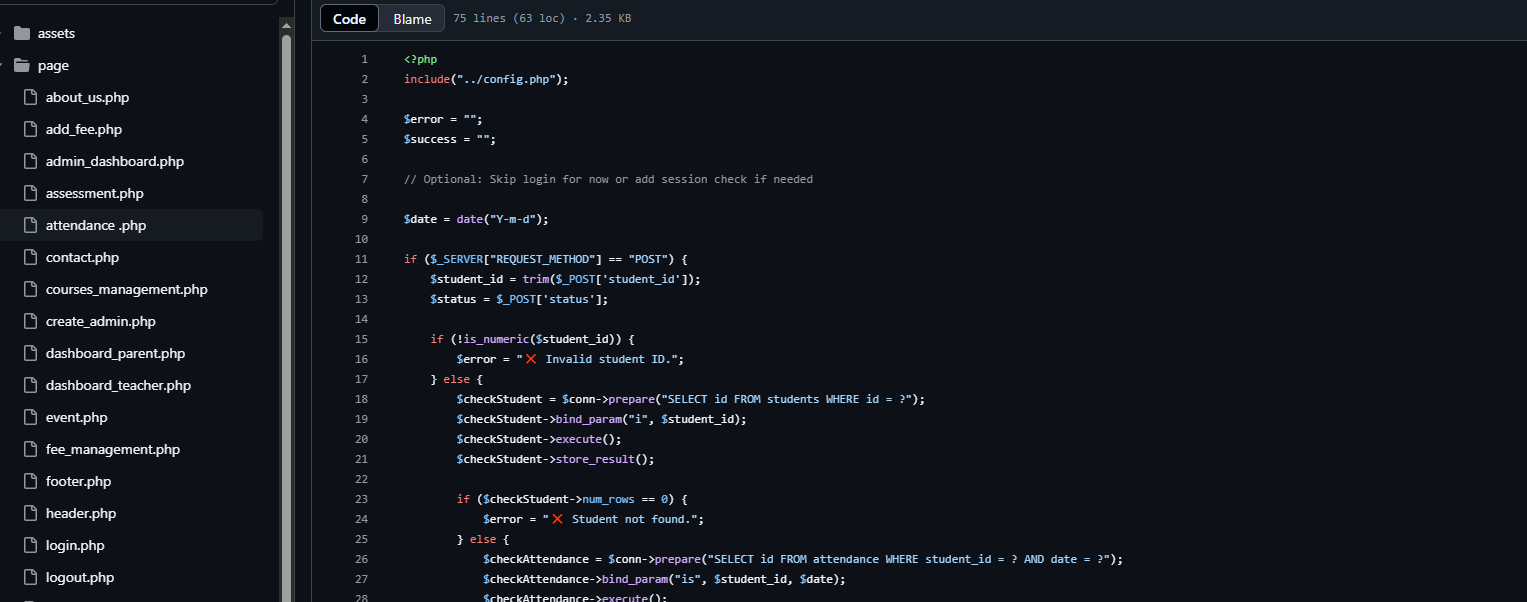
**We initialized a Git repository in our project folder using:**

**Git init**

****

**Step 2: Create GitHub Repository**

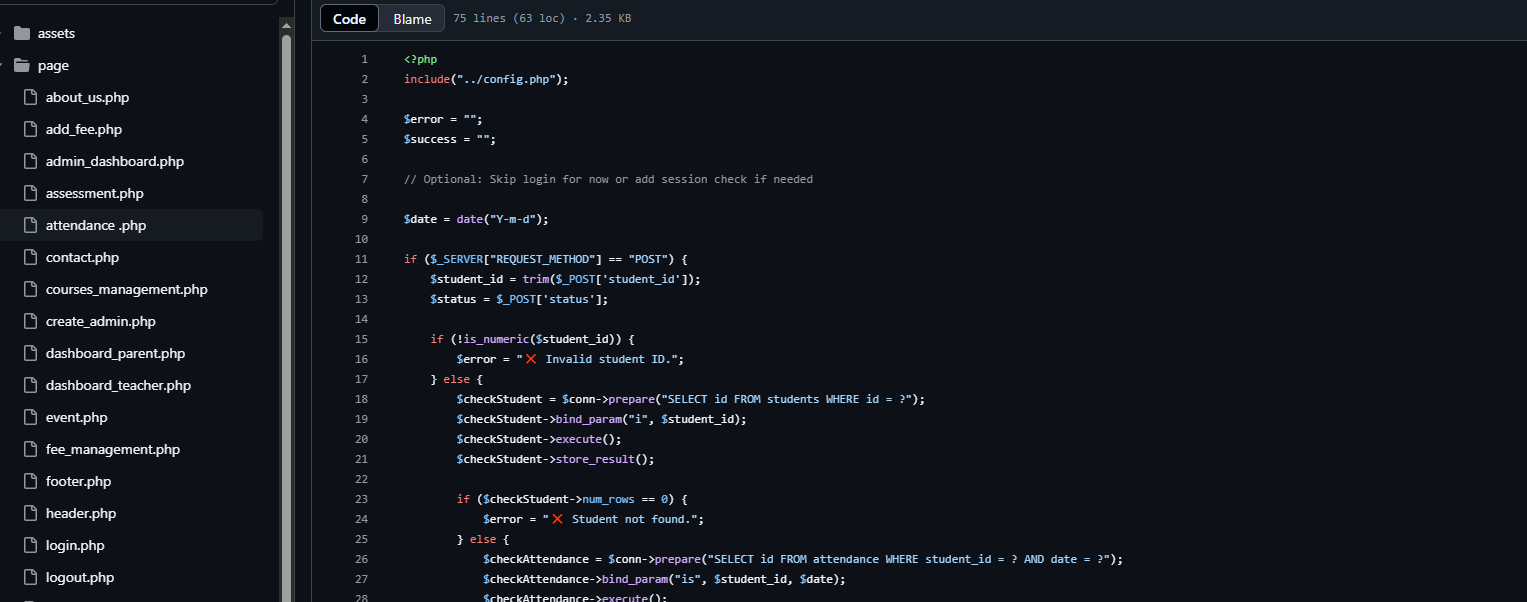
**We created a remote repository on GitHub to store our project online and enable team collaboration.**

****

**Step 3: Connect Local to Remote Repository**

**We connected our local Git repository to the GitHub repository using:**

**git remote add origin** [**https://github.com/chere-collab/kindergarten.git**](https://github.com/chere-collab/kindergarten.git)

****

**Step 4: Add and Commit Changes**

**We used the following commands to track and commit our changes:**

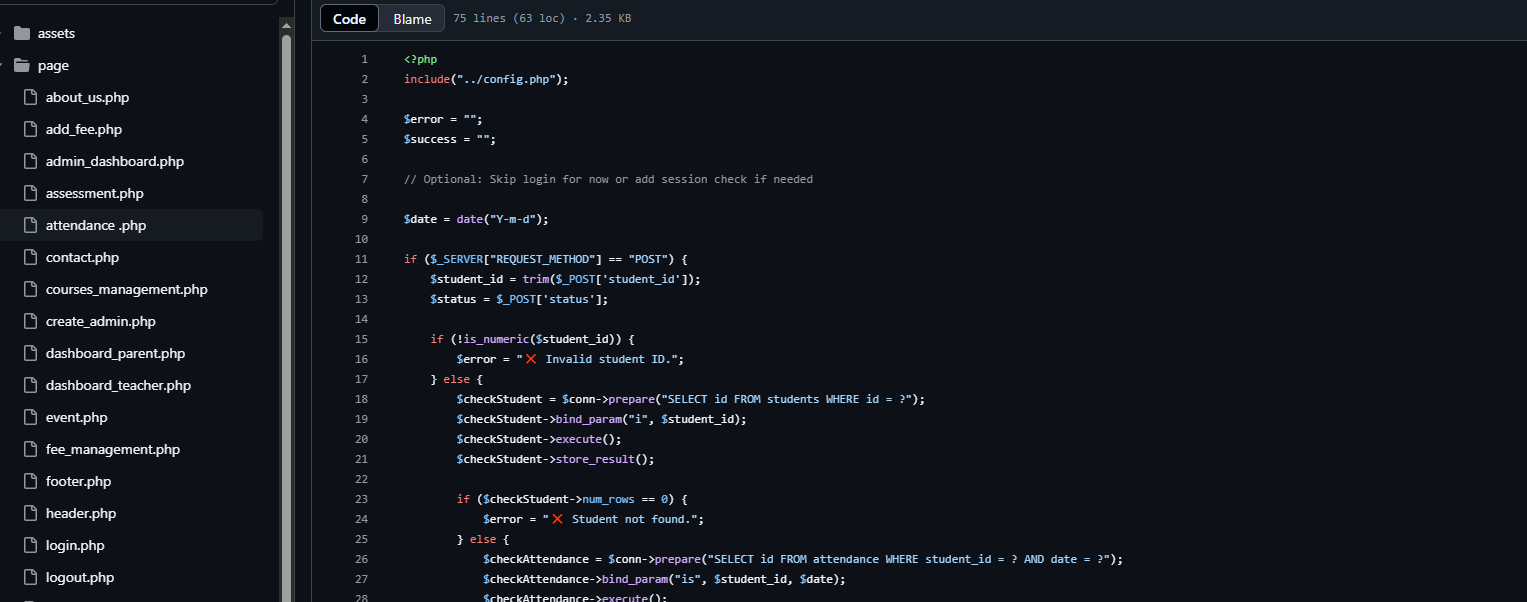
**git add .**

**git commit -m "Initial commit with project structure"**

**Step 5: Push Changes to GitHub**

**We pushed our local commits to the remote repository using:**

**git push -u origin main**

****

**Step 6: Branching and Merging**

**For new features or bug fixes, we created separate branches:**

**git checkout -b feature-branch**

**After completing the feature, we merged it into the main branch:**

**git checkout main**

**git merge feature-branch**

**Step 7: Pulling Updates**

**To stay updated with the latest changes, we used:**

**git pull**

**Tools Used:**

**Git (installed locally)**

**GitHub (as a remote repository)**

**VS Code Git Integration (for easier commits, pushes, and pull requests)**

**Git Bash (for command-line operations)**

**Benefits of Version Control (Using Git)**

**1. Tracks Every Change**

**Git records every modification to the code, including when it was made and by whom. This helps in understanding the evolution of the project.**

**2. Supports Team Collaboration**

**Multiple developers can work on different parts of the website (e.g., registration, attendance, fee management) without overwriting each other’s work.**

**3. Branching for New Features**

**Developers can create branches to add new features or fix bugs without affecting the main project. These branches can later be merged safely.**

**4. Easy Reversion**

**If a mistake is made or a bug is introduced, you can revert to a previous stable version of the project.**

**5. Better Code Organization**

**Git allows clean organization of development stages like “Initial Commit,” “Add Registration Form,” “Fix Attendance Error,” etc.**

**6. Improves Code Quality**

**With proper version control, changes can be reviewed before merging, allowing for better code through peer review.**