ADDIS ABABA SCIENCE AND TECHNOLOGY UNIVERSITY
UNIVERSITY FOR INDUSTRY

COLLEGE OF ENGINEERING

DEPARTMENT OF SOFTWARE ENGINEERING

SOFTWARE COMPONENT DESIGN

GIT/GITHUB

SECTION B

Name	ID
EDEN NEKER	ETS0395/13
ELSABETH ZELEKE	ETS0417/13
ETSUB TILAHUN	ETS0446/13
EYERUSLAEM BEZU	ETS0454/13
FIKREYOHANNES GIZACHEW	ETS0509/13

Submitted To: Mr. Gizatie Desalegn

Submission Date: Dec 18, 2024

Git

Git is a distributed version control system (VCS) that allows developers to track changes in files and collaborate on software projects. Unlike centralized VCS, Git provides each user with a full copy of the repository, including its entire history, making it highly reliable and fast. The key features of Git include:

- Version Control tracks changes to files over time, allowing developers to revert to earlier versions if needed by maintaining a detailed history of changes.
- Branching and Merging: Developers being able to create separate branches to work on features or fixes without affecting the main codebase and combining changes from one branch into another, ensuring seamless integration.
- Distributed System: Each developer has a full copy of the repository, ensuring no single point of failure.
- Lightweight and Fast: Operations like branching and committing are efficient, even for large projects.
- Staging Area: Changes are first added to a staging area before being committed, allowing better control over what changes are included.

Common Git Commands and Usage

Git Commands	Description
git config --global user.name "Your Name"	Sets the name that will be associated with your commits.
git config --global user.email "Your email"	Sets the email address that will be associated with your commits.
git clone <url>	Creates a copy of an existing remote repository at the specified URL.
git add <file>	Stages changes in the specified file for the next commit.

git commit -m "message"	Records the staged changes in the repository with a descriptive message.
git status	Displays the current state of the working directory and staging area, including untracked files and changes.
git log	Shows a chronological list of all commits in the current branch.
git branch <name>	Creates a new branch with the specified name.
git checkout <name>	Switches to the specified branch, updating the working directory to match.
git merge <branch>	Merges the specified branch into the current branch.
git remote add origin <url>	Adds a remote repository at the specified URL and names it "origin."
git push -u origin <branch>	Pushes the specified branch to the remote repository named "origin" and sets it to track the remote branch.
git pull	Fetches and merges changes from the remote repository into the current branch.

GitHub

GitHub is a web-based platform for hosting Git repositories. It extends Git's functionality by adding tools for collaboration, project management, and automation. The key features of GitHub include:

- Repository Hosting: Host public or private Git repositories and manage access controls for collaborators.
- Collaboration Tools
    - Pull Requests: propose changes, discuss them with the team, and merge them after approval.
    - Code Review: Inline comments and discussions on specific lines of code.
    - Issues: Track bugs, tasks, and feature requests with labels and milestones.
- Continuous Integration/Continuous Deployment (CI/CD)
    - Use GitHub Actions to automate tasks like testing, building, and deploying code.
- Version History and Comparison:
    - Compare changes between commits, branches, or forks visually.
- Security Tools:
    - Automatic code scanning for vulnerabilities.
- Project Management:
    - GitHub projects: organize tasks visually.
    - Milestones: Group issues and pull requests for better tracking.
    - Wikis: Document projects and provide resources for users and contributors.
- GitHub Pages:
    - Host websites directly from repositories with custom domains.

How Git and GitHub Work Together

- Local Development: Use Git to manage code on your local machine.
- Push Code to GitHub: Create a remote repository on GitHub and link it to your local repository, and push changes to share them with your team.
- Collaborate on GitHub: Create pull requests for code reviews and merge approved changes, and use issues and project boards to track progress.

- Automation with GitHub Actions: automate testing, building, and deploying using workflows.

Advantages of using Git and GitHub

- Collaboration: Teams can work on the same project without conflicts and use pull requests and reviews to ensure high-quality code.
- Version History: Track changes and revert to earlier versions when needed.
- Automation: Used to set CI/CD pipelines to improve development efficiency.
- Portfolio Showcase: GitHub repositories serve as a portfolio for developers.
- Networking: Engage with open-source communities and contribute to projects.

Project Description

The project we have worked on is the PC Registration System, which was designed to prevent the unauthorized removal of student PCs from campus. It ensures that each PC is registered under the corresponding student's details, complete with a unique serial number. The system features two primary user roles: Super Admin and Admin (security personnel). The Super Admin manages the overall system and admin accounts through a web application, while the Admin uses a separate web app to register new students and verify PCs during exit checks. This structured approach enhances campus security by providing tailored tools for each role to efficiently perform their tasks. Currently, we have worked mainly on the basic frontend and backend features, such as the admin login page, student PC registration page, and admin dashboard for the frontend, as well as corresponding backend functionalities for those pages.

To facilitate the development of this project, we utilized Git and GitHub, which allowed for effective version control and collaboration among team members. The project began with the initialization of a Git repository, enabling us to track changes and manage versions. We adopted a branching strategy, where each new feature or bug fix was developed in a separate branch. After committing the changes, it was submitted for review through pull requests. This process facilitated collaboration and ensured code quality, as team members could review each other's work before merging changes into the main branch.

In addition to version control, we implemented GitHub Actions to automate our landing page deployment process. A workflow file was created in the repository to define the steps for

Continuous Integration (CI) and Continuous Deployment (CD). This included checking out the code, installing dependencies, running tests, and deploying the application whenever changes were merged from other feature branches to the main branch. The automation facilitated a smoother deployment process, allowing us to focus on development while ensuring that the latest version of the application was always available. In addition to the deployment workflow, we have also added a testing workflow so that the sample tests added for testing some React components are run on merge to the main branch by utilizing GitHub Action.

In this project, we have effectively utilized both Git and GitHub, starting with basic features such as version control and branching and advancing to more complex functionalities. Through this process, we have gained a deeper understanding of the features Git and GitHub offer and their importance in software development. We have explored how these tools facilitate efficient management of code changes, streamline collaboration, and support the overall workflow of projects, ranging from simple tasks to more advanced implementations. This experience has reinforced our knowledge of their significance in improving the development process.