ADDIS ABABA SCIENCE AND TECHNOLOGY UNIVERSITY
UNIVERSITY FOR INDUSTRY

COLLEGE OF ENGINEERING

DEPARTMENT OF SOFTWARE ENGINEERING

SOFTWARE COMPONENT DESIGN ASSIGNMENT

GitHub - Comprehensive Document

No.	Name	ID (ETS)
1	Abenezer Merdekios	ETS 0056/13
2	Besufikad zenebe	ETS 0230/13
3	Aser Hailu	ETS 0161/13
4	Ayenew Tarekegn	ETS 0170/13
5	Betelhem Kirub	ETS 0234/13

Table of Content

Introduction to GitHub

What is GitHub?

GitHub is a web-based platform that leverages the Git version control system to facilitate software development. It provides a centralized location for developers to store, manage, and collaborate on code projects. GitHub is not just a repository hosting service; it is a comprehensive development platform that includes features for project management, code review, and continuous integration.

History and Evolution

GitHub was launched in 2008 by Tom Preston-Werner, Chris Wanstrath, and PJ Hyett. It quickly became the go-to platform for open-source projects and has since expanded to support private repositories, enterprise solutions, and a wide array of integrations. In 2018, Microsoft acquired GitHub, further enhancing its capabilities and integrating it with other Microsoft tools and services.

Key Features

- Repositories: Centralized storage for code.
- Branching and Merging: Facilitates parallel development.
- Pull Requests: Enables code review and collaboration.
- Issues and Projects: Tools for task management and project tracking.
- GitHub Actions: Automates workflows and CI/CD pipelines.
- GitHub Pages: Hosts static websites directly from a repository.

How GitHub is Applied in Software Development

Version Control with Git

Git is a distributed version control system that allows developers to track changes in their codebase. GitHub extends Git's capabilities by providing a web interface and additional features like pull requests and code review tools. This ensures that every change is documented, and the codebase remains stable and secure.

Collaboration and Teamwork

GitHub fosters collaboration by allowing multiple developers to work on the same project simultaneously. Features like pull requests, code reviews, and issues enable teams to communicate effectively, share knowledge, and maintain code quality.

Continuous Integration and Continuous Deployment (CI/CD)

GitHub Actions allows developers to automate their CI/CD pipelines. By integrating with various tools and services, GitHub Actions can build, test, and deploy code automatically, ensuring that the software is always in a deployable state.

How GitHub is Used

Setting Up a GitHub Account

To start using GitHub, you need to create an account at github.com. Once registered, you can create repositories, join organizations, and start collaborating on projects.

Creating and Managing Repositories

A repository (or "repo") is a project's folder on GitHub. You can create a new repository by clicking the "New" button on your GitHub dashboard. Repositories can be public (visible to everyone) or private (visible only to collaborators).

Using Branches and Pull Requests

Branches allow developers to work on new features or bug fixes without affecting the main codebase. Once the work is complete, a pull request can be created to merge the changes back into the main branch. This process involves code review, where team members can comment on and approve the changes.

GitHub Actions and Workflows

GitHub Actions is a powerful tool for automating workflows. You can create custom workflows using YAML files stored in the `.github/workflows` directory of your repository. These workflows can be triggered by events like push, pull request, or issue creation, and can perform tasks like running tests, building applications, and deploying code.

The Importance of GitHub

Code Management and Security

GitHub provides a secure platform for managing code with features like branch protection, required reviews, and two-factor authentication. This ensures that only authorized changes are made, and the project remains stable and secure.

Community and Open Source

GitHub supports millions of open-source projects, fostering a global development community. Developers can collaborate, contribute, and learn from projects worldwide, helping them grow professionally and build robust portfolios.

Documentation and Knowledge Sharing

GitHub encourages thorough project documentation through README files, wikis, and integrated project boards. This improves knowledge sharing and makes onboarding new team members easier.

Working Example: Our Group Portfolio Web - PlyCollab

Project Overview

Our group portfolio web showcases our projects and skills. The website is hosted using GitHub Pages and managed through a GitHub repository.

GitHub Repository Structure

- `/docs`: Contains HTML, CSS, and JavaScript files.
- `/assets`: Stores images and media files.
- `/README.md`: Provides an overview and setup instructions.
- `.github/workflows/deploy.yml`: Automates website deployment.

Collaboration Workflow

1. Branching: Each team member works on a separate branch.
2. Pull Requests: Created upon task completion.
3. Code Review: Team reviews and approves changes.
4. Merging: Approved changes merge into the main branch.

Deployment and Maintenance

The website is automatically deployed to GitHub Pages when changes are merged, ensuring the latest updates are always live.

Practical Guide: Essential Git Commands and Configurations

Git Installation and Configuration

1. Install Git: Follow instructions from git-scm.com.
2. Configure Git:

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

Basic Git Commands

- Initialize a Repository:
  ```
  git init
  ```
- Clone a Repository:
  ```
  git clone <repository-url>
  ```
- Check Repository Status:
  ```
  git status
  ```
- Stage Changes:
  ```
  git add <file-name>
  ```
- Commit Changes:
  ```
  git commit -m "Commit message"
  ```
- Push Changes:
  ```
  git push origin <branch-name>
  ```

Managing Branches and Pull Requests

- Create a New Branch:
  ```
  git checkout -b <branch-name>
  ```

- Switch Branches:
  ```
  git checkout <branch-name>
  ```
- Merge Branches:
  ```
  git merge <branch-name>
  ```
- Create a Pull Request: Use the GitHub web interface to propose changes.

Handling Merges and Conflicts

- Resolve Conflicts: Open conflicting files, make corrections, and mark resolutions.
- Commit Resolved Changes:
  ```
  git add <file-name>

  git commit -m "Resolved merge conflicts"
  ```

Conclusion

GitHub is an indispensable tool in modern software development, offering a comprehensive suite of features for version control, collaboration, and automation. Its importance extends beyond code management to fostering a global community of developers and promoting open-source collaboration. By leveraging GitHub effectively, teams can streamline their workflows, enhance code quality, and deliver software more efficiently.

Future Trends and Innovations

GitHub continues to evolve with new features such as AI-powered tools like GitHub Copilot, enhanced CI/CD workflows, and deeper integration with cloud services. Developers can expect further innovations focused on automation, security, and collaboration.

References

- [ ] GitHub Official Website
- [ ] Git Documentation
- [ ] GitHub Actions Documentation
- [ ] GitHub Pages Documentation
- [ ] https://github.com/ebenezermerd/PlyCollab.git