JUDITH AKINYI ATELA

SOFTWARE ENG. ASSIGN 4

**GitHub Overview**

GitHub is a web-based platform that provides version control and collaboration tools for software developers. It uses Git, a distributed version control system, to track changes in source code during software development. GitHub offers a web-based graphical interface and desktop as well as mobile integration.

**Primary Functions and Features:**

1. Repositories: Storage locations for projects.

2. Branches: Different versions of a repository.

3. Pull Requests: Proposed changes to a repository.

4. Issues: Bug tracking and project management.

5. Actions: Automation and CI/CD workflows.

6. Projects: Kanban-style project boards.

7. Wiki: Documentation for repositories.

8. Gists: Shareable code snippets.

**Collaborative Software Development with GitHub**

GitHub supports collaborative development by allowing multiple developers to work on the same project simultaneously, merge their changes, review code, and track progress. Its version control system ensures that every change is recorded and reversible, promoting a structured and organized workflow.

**Repositories on GitHub**

GitHub Repository: A repository is a central place where all the files for a project are stored. Each repository contains all the project files, including the revision history.

**Creating a New Repository:**

1. Go to your GitHub account.

2. Click the "+" icon in the upper-right corner and select "New repository".

3. Fill in the repository name, description (optional), and choose visibility (public or private).

4. Initialize the repository with a README, .gitignore, and license if desired.

5. Click "Create repository".

**Essential Elements of a Repository:**

- README.md: Provides an overview and documentation.

- .gitignore: Specifies files to ignore in version control.

- LICENSE: Specifies the legal usage of the project.

- src/: Directory for source code.

- docs/: Directory for documentation.

**Version Control with Git**

Version Control: A system that records changes to a file or set of files over time so that specific versions can be recalled later.

Git: A distributed version control system that tracks changes and allows multiple developers to collaborate.

GitHub Enhances Version Control:

- Central Repository: GitHub acts as a central repository for all code.

- Collaboration Tools: Pull requests, code reviews, and comments.

- Backup and Recovery: Stores code in the cloud, preventing data loss.

- Integration: Integrates with various tools and services for enhanced functionality.

**Branching and Merging in GitHub**

Branches: Branches are used to develop features, fix bugs, or experiment without affecting the main codebase. They are important because they allow multiple development tracks to exist concurrently.

Creating a Branch:

1. Navigate to your repository on GitHub.

2. Click the "Branch: main" dropdown.

3. Enter a new branch name and press "Create branch".

Making Changes and Merging:

1. Switch to your new branch.

2. Make and commit changes.

3. Push changes to GitHub.

4. Open a pull request to merge changes into the main branch.

5. Review and merge the pull request.

Pull Requests and Code Reviews

Pull Request (PR): A pull request is a method of submitting contributions to a project. It facilitates code reviews and collaboration.

**Creating a Pull Request:**

1. Navigate to the repository and switch to the branch with changes.

2. Click "New pull request".

3. Compare changes and create a pull request.

4. Add reviewers and comments.

5. Reviewers review, comment, and approve changes.

Reviewing a Pull Request:

1. Open the pull request from the repository.

2. Review the changes.

3. Add comments or request changes.

4. Approve and merge the pull request if satisfied.

GitHub Actions

GitHub Actions: GitHub Actions allows you to automate workflows for your repository. It can be used to build, test, and deploy code automatically.

Example of a Simple CI/CD Pipeline:

1. Create a `.github/workflows` directory in your repository.

2. Add a YAML file (e.g., `ci.yml`) with the following content:

```yaml

name: CI

on: [push]

jobs:

build:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v2

- name: Set up Node.js

uses: actions/setup-node@v2

with:

node-version: '14'

- run: npm install

- run: npm test

```

3. Commit and push the changes to trigger the workflow.

**Introduction to Visual Studio**

Visual Studio: An integrated development environment (IDE) from Microsoft. It supports multiple programming languages and has tools for debugging, code completion, and project management.

Key Features:

- Advanced debugging and diagnostics tools.

- Code refactoring and IntelliSense.

- Integrated Git and GitHub support.

- Azure development integration.

\*\*Visual Studio vs. Visual Studio Code\*\*:

- \*\*Visual Studio\*\*: Full-featured IDE for complex, enterprise-level applications.

- \*\*Visual Studio Code\*\*: Lightweight, open-source code editor with support for extensions.

Integrating GitHub with Visual Studio

Steps to Integrate GitHub:

1. Open Visual Studio.

2. Go to `File > Add to Source Control`.

3. Select `Git` and connect to your GitHub account.

4. Clone an existing repository or create a new one.

5. Use the built-in Git tools to commit, push, and pull changes.

Enhancing Development Workflow:

- Seamless integration with GitHub for version control.

- Easy collaboration with team members through pull requests and code reviews.

- Access to GitHub Actions for CI/CD within the IDE.

Debugging in Visual Studio

Debugging Tools:

- Breakpoints: Pause execution at specific lines.

- Watch: Monitor variables and expressions.

- Call Stack: View the order of function calls.

- Immediate Window: Execute expressions during debugging.

- Locals: Inspect variables within the current scope.

Using Debugging Tools:

1. Set breakpoints by clicking the gutter next to the code line.

2. Start debugging (F5).

3. Use the Debug toolbar to step into, over, and out of functions.

4. Inspect variables and modify their values during execution.

Collaborative Development using GitHub and Visual Studio

**Combining GitHub and Visual Studio:**

- Version Control: Git integration in Visual Studio for version control.

- Code Reviews: Pull requests for code reviews directly from the IDE.

- Project Management: Use GitHub Issues and Projects for task management.

- CI/CD: GitHub Actions for automated testing and deployment.

Real-World Example:

A team developing a web application uses Visual Studio for coding and debugging. They manage their codebase with GitHub, where they create branches for new features, submit pull requests for review, and use GitHub Actions to automatically deploy changes to a staging environment. This integration ensures efficient collaboration, continuous integration, and seamless deployment.