**Name: Ann Kairuthi**

**Email:** [**kairuthiann2@gmail.com**](mailto:kairuthiann2@gmail.com)

**Task:** Answer the following questions based on your understanding of GitHub and Visual Studio. Provide detailed explanations and examples where appropriate.

Questions:

**Introduction to GitHub**

What is GitHub, and what are its primary functions and features? Explain how it supports collaborative software development.

GitHub is an online software development platform used for storing, tracking, and collaborating on software projects. It is built on top of Git, a distributed version control system.

***Primary functions and features***

Version control

GitHub leverages Git to track changes to source code over time, allowing developers to revert to previous versions, track history of modifications and manage multiple versions of their projects. In this case commit history, branches and tags are the key features

Repositories

A repository is a central place where all files, code and documentation for a project are stored. These can be private or public based on an individual's preferences when it comes to hosting options.

Pull Requests

These are mechanisms for proposing changes to the codebase. They allow team members to review and discuss the changes before integrating them into the main branch. Some of the features that apply in this case include, approval workflows and code review.

Code Hosting and Distribution

GitHub hosts code repositories, making them accessible to collaborators and wider community through GitHub pages for hosting static websites as well as release management for distributing software versions.

Continuous Integration/Continuous Deployment (CI/CD)

GitHub integrates with CI/CD tools to automate the testing, building and deployment of software.

***How GitHub Supports Collaborative Software Development***

GitHub wikis and README files enable teams to document their projects, ensuring that information about the codebase, usage and development processes is easily accessible.

Pull requests facilitate code reviews allowing team members to comment on and suggest improvement to code changes. Discussion threads enable asynchronous communication which is essential for distributed teams.

Developers can work on separate branches for different features or bug fixes. Through pull requests they are able to review and discuss changes before merging them into the main branch, ensuring code quality and team consensus.

GitHub Actions enables teams to set up workflows to automate building, testing and deployment processes thus ensuring that code changes are automatically validated and deployed, reducing manual efforts and error.

**Repositories on GitHub**

What is a GitHub repository? Describe how to create a new repository and the essential elements that should be included in it.

A repository is a central place where all files, code and documentation for a project are stored.

***How to create a new repository***

Open your web browser and navigate to GitHub

Log in to your GitHub account.

Click on the green button “New” on the left side of the GitHub interface

Configure the repository

* Repository Name: enter a unique name for your repository
* Description: (Optional) Provide a brief description of the repository
* Public or Private: Choose the visibility of the repository. Public repositories can be seen by anyone, while private repositories are only accessible to you and people you explicitly share it with.
* Initialize with a README: Check this box to include a README file, which is essential for providing information about your project
* Add. gitignore: This file specifies which files or directory to ignore in your repository.
* Add a license: You might choose an appropriate license for your project. It defines the terms under which others can use, modify and distribute your code

Click on the “Create repository “button to finalize the creation of your new repository

**Version Control with Git**

Explain the concept of version control in the context of Git. How does GitHub enhance version control for developers?

Version control is the practice of tracking and managing changes in software code at different points in time.

***GitHub enhances version control for developers by:***

Providing hosting for git repositories, making code easily accessible to all team members and collaborators from anywhere with internet access.

Enabling developers to propose changes, discuss them, review code and merge it into the main branch through pull requests.

Allowing developers to create and manage branches easily without affecting the main codebase.

Maintaining a detailed history of all changes made to the codebase, including who made the change, when and why (through commit messages).

GitHub Actions enables developers to automate workflows, such as building applications, running tests and deploying code whenever changes are pushed to the repository

**Branching and Merging in GitHub**

What are branches in GitHub, and why are they important? Describe the process of creating a branch, making changes, and merging it back into the main branch.

In GitHub a branch is a parallel version of the repository that diverges from the main codebase.

***Branches are important because***

* The allow multiple team members to work on different parts of the project simultaneously
* They isolate experimental changes, reducing the risk of affecting the main codebase
* They support structured workflow like GitFlow, where branches are used for features, hotfixes and releases.

***Creating a branch***

On GitHub, access your repository and click the "master" branch button.

There you can create a new Branch. Type in a descriptive name, and click Create branch:

The branch should now be created and active. You can confirm which branch you are working on by looking at the branch button.

When using the Git command run `git checkout -b new-branch-name`.

***Making changes***

Start working on an existing file in this branch. Click the "index.html" file and start editing:

After you have finished editing the file, you can click the "Preview changes" tab to see the changes you made highlighted:

If you are happy with the change, add a comment that explains what you did, and click Commit changes.

You now have a new branch on GitHub, updated with some changes

***Merging branches***

Open a Pull Request (PR) from the feature branch to the main branch and merge it once approved.

**Pull Requests and Code Reviews**

What is a pull request in GitHub, and how does it facilitate code reviews and collaboration? Outline the steps to create and review a pull request.

Pull request in GitHub is a feature that allows developers to notify team members and other collaborators about changes that have been made to a branch in a repository.

***How pull request facilitates code reviews and collaborations***

Team members are able to review the proposed changes, providing feedback, suggestions and approval before the changes are merged. Reviewers make comments on specific lines of code making it easier to discuss changes in context.

Pull requests provide a documented history of what changes were made, why they were made and who reviewed and approved them.

They support threaded discussions where team members can discuss the changes, share insights and address concerns.

***steps to create pull request***

1. Navigate to the repository on GitHub.
2. Click on the "Pull requests" tab.
3. Click the "New pull request" button.
4. Select the base branch (usually main) and the compare branch (your feature branch).
5. Add a title and description for the pull request, explaining the changes made.
6. Click "Create pull request."

***Steps to review a pull request***

1. Navigate to the repository on GitHub.
2. Click on the "Pull requests" tab.
3. Select the pull request you want to review.
4. Click on the "Files changed" tab to see the diff of the changes.
5. Click on specific lines to add comments or suggestions.
6. Provide overall feedback in the main comment section.
7. If the changes are satisfactory, click on "Review changes," select "Approve," and submit the review.
8. If changes are needed, select "Request changes," provide details on what needs to be addressed, and submit the review.

**GitHub Actions**

Explain what GitHub Actions are and how they can be used to automate workflows. Provide an example of a simple CI/CD pipeline using GitHub Actions.

GitHub Actions is a powerful feature within GitHub that allows developers to automate workflows directly within their repositories to automate directly within their repositories. With GitHub Actions, you can set up continuous integration (CI) and continuous deployment (CD) pipelines, automate testing and manage releases.

***Automation workflows with GitHub Actions***

Event-Driven Automation

Workflows can be triggered by various events like code pushes, pull request submissions, releases or even on a schedule e.g. a workflow that runs tests every time code is pushed to the repository.

Jobs and steps

A workflow consists of one or more jobs which run in parallel by default. Each job is composed of multiple steps that execute commands or actions. Steps within a job run sequentially.

Reusable actions

Actions are reusable units of code that can perform specific tasks such as checking out code, setting up a programming environment or deploying an application.

***Example of a simple CI/CD pipeline using GitHub Actions.***

Create a Workflow File:

In your GitHub repository, navigate to the. GitHub/workflows directory. If it doesn't exist, create it.

Create a new file named ci-cd-pipeline.yml.

Define the Workflow:

Open ci-cd-pipeline.yml and define the workflow as follows:

yaml

Copy code

name: CI/CD Pipeline

on:

push:

branches:

- main

jobs:

build:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v2

- name: Set up Node.js

uses: actions/setup-node@v2

with:

node-version: '14'

- name: Install dependencies

run: npm install

- name: Run tests

run: npm test

deploy:

needs: build

runs-on: ubuntu-latest

if: success()

steps:

- name: Checkout code

uses: actions/checkout@v2

- name: Deploy to production

env:

HEROKU\_API\_KEY: ${{ secrets.HEROKU\_API\_KEY }}

run: |

git remote add heroku https://git.heroku.com/<your-app-name>.git

git push heroku main

I**ntroduction to Visual Studio**

What is Visual Studio, and what are its key features? How does it differ from Visual Studio Code?

Visual Studio is an integrated development environment (IDE) developed by Microsoft. It is a comprehensive suite of tools designed primarily for software developers working on large-scale enterprise applications.

***Key features of visual studio***

Intergrated Development Environment (IDE)

Provides a comprehensive environment for development, including a rich editor, solution, e explorer and integrated tools for project management

Multi-language Support

Supports multiple programming languages like C#, VB.NET, C++, Python, JavaScript, and more. This allows developers to work on different types of projects without switching tools.

Advanced Debugging and Diagnostics

Offers advanced debugging tools, including breakpoints, watch windows, call stacks, and IntelliTrace (historical debugging). This makes it easier to identify and fix issues in the code.

Integrated Testing Tools

Includes unit testing, load testing, and UI testing frameworks ensuring code quality and performance through comprehensive testing.

Azure Integration

Provides tools for developing, deploying, and managing applications on Microsoft Azure.

***Differences between Visual studio and Visual studio code***

|  |  |
| --- | --- |
| Visual studio | Visual studio code |
| It is an integrated development environment | lightweight code editor |
| Extensive, with advanced features for .NET and C++ | Broad, extensible through extensions |
| Testing through integrated testing frameworks | Testing is via extensions |
| It involves advanced debugging tools, including IntelliTrace | Built-in debugging with extensions for advanced support |
| Collaboration through Azure DevOps, Git integration | Collaboration Git integration |

**Integrating GitHub with Visual Studio**

Describe the steps to integrate a GitHub repository with Visual Studio. How does this integration enhance the development workflow?

***steps to integrate a GitHub repository with Visual Studio***

Set Up GitHub Account in Visual Studio:

1. Open Visual Studio.
2. Go to File > Account Settings > All Accounts.
3. Click on Add an account, then select GitHub.
4. Sign in to your GitHub account using your credentials.

Clone a GitHub Repository:

1. Open Visual Studio.
2. Go to File > Open > Open from Source Control > Clone Repository.
3. In the Clone a Repository window, enter the URL of your GitHub repository and choose a local folder where you want to clone the repository.
4. Click Clone.

Create a New Repository on GitHub from Visual Studio:

1. Open your project in Visual Studio.
2. Go to File > New > Repository > Add to Source Control > Git.
3. In the Create a Git repository window, select GitHub and provide a name and description for your repository.
4. Choose the option to create a. gitignore file and a README.md file if desired.
5. Click Create and Push.

Push Changes to GitHub:

1. Make changes to your code in Visual Studio.
2. Open the Team Explorer window by going to View > Team Explorer.
3. In Team Explorer, click on Changes to view the changes you have made.
4. Enter a commit message and click Commit All.
5. Click Sync to push the changes to GitHub.

Pull Changes from GitHub:

1. In the Team Explorer window, click on Sync.
2. Click Pull to fetch and integrate changes from the remote GitHub repository to your local repository.

***How integration enhances the development workflow?***

Managing Git operations like cloning, committing, pushing and branching directly within visual studio without needing to switch to a command-line interface or other tool

Creating and managing pull requests directly from Visual Studio, facilitating code reviews and collaborating with team members.

Integrating GitHub Actions for CI/CD workflows to automate testing, building, and deploying your application whenever code is pushed to the repository.

Visualizing the commit history and browsing through changes, providing clear traceability of code modifications over time.

**Debugging in Visual Studio**

Explain the debugging tools available in Visual Studio. How can developers use these tools to identify and fix issues in their code?

***Key Debugging tools in Visual Studio***

Breakpoints

These allow developers to pause the execution of their program at specific lines of code. Set a breakpoint by clicking on the left margin next to the line number or pressing `F9`

Local and Autos Windows

Displays all local variables in the current scope providing a quick view of variable values relevant to the current execution context, aiding in debugging local issues.

Call Stack Windows

Displays the order of function calls that led to the current point of execution

Immediate Window

Executes code or evaluates expressions during a debugging session. Open the immediate window from Debug > Windows > Immediate and type commands or expressions.

IntelliTrace

Records the history of your debugging session, capturing data such as function calls, variable values and exceptions.

***Using Debugging tools to Identify and Fix Issues***

Identify the section of code where you suspect issues then set breakpoints at strategic points to pause execution and examine the program's state.

Use step-in (F11), step-over (F10), and step-out (Shift+F11) functions to navigate through code line by line. This helps in understanding the flow and identifying where things go wrong.

Execute commands or evaluate expressions in the Immediate window to test fixes or check values. This allows for quick validation of potential solutions.

Use IntelliTrace to review the history of your debugging session. Navigate back to previous states to understand how the application reached the current state and identify where issues originated.

Use the Call Stack window to trace the sequence of function calls.

Identify if any function calls are out of order or if there are unexpected calls that could be causing issues.

**Collaborative Development using GitHub and Visual Studio**

Discuss how GitHub and Visual Studio can be used together to support collaborative development. Provide a real-world example of a project that benefits from this integration.

GitHub provides robust version control, allowing multiple developers to work on the same codebase simultaneously. Visual Studio integrates seamlessly with GitHub, making it easy to commit, push, pull, and manage branches directly from the IDE. This ensures that changes are tracked, and conflicts are managed efficiently.

Developers can link commits and pull requests to specific GitHub issues, providing a clear traceability of tasks and progress. This helps in managing development tasks, tracking bugs, and prioritizing features.

Visual Studio Live Share allows developers to collaboratively edit and debug in real-time. This can be complemented by GitHub's collaborative features. Facilitates real-time problem-solving and knowledge sharing among team members.

Automate workflows such as building, testing, and deploying applications. Visual Studio provides integration with GitHub Actions, allowing developers to trigger CI/CD pipelines directly from the IDE. Ensures that the application is tested and deployed consistently, reducing the chances of errors.

***Real-world example of a project that benefits from* GitHub and Visual Studio *integration***

A developer finds a bug in VS Code's syntax highlighting. They clone the repository using Visual Studio, fix the bug, and submit a pull request on GitHub. The pull request undergoes review and is eventually merged, improving the editor for all users.

A user requests a new feature for Python debugging. The issue is tracked on GitHub, discussed by the community, and picked up by a contributor who implements the feature and submits a pull request.

A new feature is added via a pull request. GitHub Actions automatically run tests to verify the feature's integration. Once tests pass, the feature is merged and included in the next release.

**References**

Git Documentation. (n.d.). Git Documentation. Retrieved from <https://git-scm.com/doc>

Microsoft. (n.d.). Visual Studio Code Documentation. Retrieved from <https://code.visualstudio.com/docs>

Microsoft. (n.d.). Visual Studio Documentation. Retrieved from <https://docs.microsoft.com/en-us/visualstudio/>

Microsoft. (n.d.). Debugging in Visual Studio. Retrieved from <https://docs.microsoft.com/en-us/visualstudio/debugger/>

**Submission Guidelines:** Your answers should be well-structured, concise, and to the point. Provide real-world examples or case studies wherever possible. Cite any references or sources you use in your answers. Submit your completed assignment by [due date].