**SOFTWARE ENGINEERING:  
­­­­­­­­­Q1: Intro to GItHub**

Git-Hub is a web-based platform for version control and collaborative software development.

Its primary functions and features are:

1. **Repositories:** Storage locations for project files and version histories.

2. **Branches:** Isolated workspaces to develop features or fix bugs.

3. **Pull Requests:** Proposals to merge changes from one branch into another.

It supports collaborative software development by tracking changes in source code during software development and managing access and collaboration among team members.

**Q2: Repositories on GitHub**

Creating a GitHub repo.

1. Log in to GitHub: Navigate to your GitHub dashboard.
2. Click on the "New" button: Found next to your repositories list.
3. Fill in the repository details:
   * Repository Name: Unique name for your repository.
   * Description: Optional brief description of your project.
   * Public/Private: Choose the visibility of your repository.
   * Initialize with a README: Optional but recommended to describe your project.
4. Click "Create repository": Your new repository is now ready.

Essential Elements in a Repository:

* README.MD: Provides an overview and documentation of the project.
* .gitignore: Specifies files and directories to be ignored by Git.

**Q3: Version Control with Git**

Git is a system that records changes to a file or set of files over time, allowing you to revert to specific versions later.

Git enhances Version Control for developers by use of **Centralized Collaboration:** Provides a central repository for code and **Pull requests** to simplify code reviews and collaboration.

**Q4: Branching and merging**

Branches are parallel versions of a repository, allowing developers to work on different features or fixes without affecting the main codebase.

Process of creating a branch, making changes, and merging it into the main branch.

Creating a Branch:

1. Navigate to your repository:
2. Click the "Branch" button: Found next to the branch selector dropdown.
3. Enter a branch name: Choose a descriptive name for your branch.
4. Click "Create branch":

Making Changes:

* Checkout the branch: git checkout branch\_name
* Make changes and commit: git add . and git commit -m "Your message"

Merging a Branch:

1. Create a pull request: Propose merging your branch into the main branch.
2. Review and discuss changes: Team members can review and comment.
3. Merge the pull request: Once approved, merge the branch.

**Q5: Pull Requests and Code Reviews**

A pull request is a way to propose changes to a GitHub repository. It allows developers to discuss, review, and approve changes before merging them into the main branch.

Steps to Create and Review a Pull Request:

1. Navigate to the repository:
2. Click the "Pull requests" tab:
3. Click "New pull request":
4. Select the branch to merge:
5. Provide a title and description:
6. Create the pull request:

Reviewing a Pull Request:

1. Navigate to the pull request:
2. Review the changes: Use inline comments to provide feedback.
3. Approve or request changes: Indicate your approval or request further modifications.
4. Merge the pull request: Once approved, merge it into the main branch.

**Q6: GitHub Actions**

GitHub Actions are workflows that automate tasks such as building, testing, and deploying code.

Example:  
name: CI/CD Pipeline

on: [push, pull\_request]

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

- run: npm run build

**Q7: Introduction to Visual Studio Code**

Visual Studio is an integrated development environment (IDE) from Microsoft.

**Key Features:**

* **Code editor:** Supports multiple languages.
* **Debugger:** Advanced debugging capabilities.
* **Extensions:** Enhance functionality with plugins

How does Visual Studio differ from Visual Studio Code?

Visual Studio is a full IDE for performing complex tasks, while Visual Studio Code is a lightweight, extensible code editor for performing simpler tasks

**Q8: Integrating GitHub with Visual Studio**

Steps to Integrate GitHub with Visual Studio:

1. Install GitHub extension: From the Visual Studio marketplace.
2. Sign in to GitHub: Use your GitHub credentials.
3. Clone a repository: Use the GitHub extension to clone a repository.
4. Make changes and commit: Use Visual Studio’s integrated Git tools.

Enhancement to Development Workflow:

* Seamless integration: Directly commit, push, and pull changes.
* Code reviews: Use GitHub PRs within Visual Studio.
* Issue tracking: Link commits to GitHub issues.

**Q9:** **Debugging in Visual Studio**

Debugging Tools:

* Breakpoints: Pause code execution at specific points.
* Watch windows: Monitor variables and expressions.
* Call stack: View the call stack at any point during execution.
* Immediate window: Execute code during debugging.
* Step into/over/out: Control execution flow.

Identifying and Fixing Issues:

* Set breakpoints: Pause execution to examine the state of the application.
* Inspect variables: Use watch windows to check values.
* Step through code: Follow the execution path to identify issues.

Q10: **Collaborative Development using GitHub and Visual Studio**

Using GitHub and Visual Studio Together:

* Version control: Use GitHub for version control and collaboration.
* Development environment: Use Visual Studio for coding and debugging.
* Code reviews: Conduct code reviews using GitHub PRs within Visual Studio.
* Issue tracking: Link issues and PRs to specific commits and branches.

Real-World Example:

* Project: Developing a web application.
* Setup: Use GitHub for version control, Visual Studio for development.
* Workflow: Clone the repository in Visual Studio, create branches for features, push changes, create PRs for code reviews, and merge changes into the main branch.
* Collaboration: Team members can review code, suggest changes, and work on different features simultaneously.