**\*\*SEPTA TEAM TITAN\*\***

**-- MICHAEL AKANDE --**

\*\*Question 1: What is Git, and what problem does it solve in the context of software development?\*\*

Git serves as a decentralized version control system that facilitates collaborative development among multiple developers working on the same project.

It tackles the challenge of managing and coordinating code changes contributed by various developers. Git empowers developers to work autonomously on their local copies of the codebase, make alterations, and subsequently merge these modifications into a central repository.

\*\*Question 2: Explain the difference between Git and GitHub.\*\*

Git functions as the foundational version control system, while GitHub acts as a web-based platform providing hosting for repositories managed by Git.

GitHub incorporates features that foster collaboration such as tools for working together, tracking issues, initiating pull requests, and more. Essentially, Git provides the underlying technology for version control, while GitHub is a service built upon the Git framework.

\*\*Question 3: Describe the basic workflow for using Git for version control.\*\*

- Clone: Create a duplicate of a remote repository on your local machine using `git clone`.

- Work: Implement changes to files within your local repository.

- Stage: Utilize `git add` to designate changes for the upcoming commit.

- Commit: Capture staged changes as a commit via `git commit`. This produces a snapshot of alterations accompanied by a commit message.

- Push: Dispatch your commits to the remote repository through `git push`, sharing your changes with others.

- Pull: If other contributors have made changes, employ `git pull` to retrieve and merge those changes into your local repository.

\*\*Question 4: What is a Git repository? How does it store and manage versions of files?\*\*

A Git repository constitutes a directory housing the comprehensive history of project versions. It maintains this history as a series of commits. Commits embody snapshots of the entire project at specific time points. Git employs a directed acyclic graph structure to depict connections between commits. Each commit documents modifications as a set of differences relative to the previous commit.

\*\*Question 5: What is a commit in Git? What information does a commit include?\*\*

In Git, a commit symbolizes a snapshot of the project at a particular juncture. It encompasses:

* - A distinct hash (SHA-1) for identification.
* - Information regarding the author and committer.
* - Timestamp indicating the commit's creation time.
* - A commit message elucidating the implemented changes.
* - References to parent commit(s), constituting the commit lineage.

\*\*Question 6: Explain the concept of branching in Git. How can branching be useful in collaborative development?\*\*

Branching in Git enables developers to establish distinct development pathways within a repository. Each branch represents a distinct rendition of the code. This feature proves beneficial for segregating features, rectifying bugs, or conducting experiments without influencing the core codebase. Developers can work autonomously on their branches, subsequently merging their changes into the main branch when they deem it appropriate. Consequently, this approach facilitates multiple developers to collaborate concurrently on diverse features without impeding each other's progress.