YANORIA, TRISTAN VON CEAZAR A.	GIT SEATWORK
CPE 408 - CPE43S1	ENGR. JAYSON CARL FERRER

Instructions: Answer the following questions and cite your sources

Essay:

Explain the Role of Git in Modern Software Development:

- Discuss how Git supports collaboration, version control, and continuous integration in modern software engineering workflows. Include examples of scenarios where Git significantly improves productivity and code quality.
- Git is a distributed version control system essential for modern software development. It enables collaboration by allowing developers to work on separate branches and merge their work efficiently. Platforms like GitHub support this with pull requests and code reviews. Git provides version control, letting teams track changes, revert mistakes, and identify who made specific edits. For example, if a bug appears, Git helps quickly trace and fix the issue. Git also supports continuous integration (CI). When code is pushed, automated tests and builds can run via tools like GitHub Actions, improving code quality and reducing manual errors.

Describe the Git Workflow Models and Their Use Cases:

- Explore different Git workflows, such as Git Flow, GitHub Flow, and trunk-based development. Discuss when and why each might be appropriate in a team or organization.
- Different Git workflows fit different team needs. Git Flow uses multiple branches to develop, release, then hotfix and is ideal for large projects with scheduled releases. GitHub Flow is simpler, using short-lived feature branches merged into the main branch, suitable for teams that deploy frequently, such as web or SaaS developers. Trunk-based development encourages developers to commit directly or through very short-lived branches to the main branch, enabling rapid integration and frequent releases, making it ideal for fast-paced DevOps or startup teams.

Explain the Internals of Git: How Does Git Work Under the Hood?

- Dive into Git's internal architecture, including the concepts of snapshots, commits, trees, blobs, and the staging area. Explain how these components interact to manage and track changes in a repository
- Git manages data using snapshots rather than differences. Each commit stores a full snapshot of the project, with unchanged files referencing earlier versions to save space.

Git uses blobs to store file contents, trees to represent directories, and commits to capture snapshots along with metadata like author and message. All objects are identified by SHA-1 hashes to ensure data integrity. The staging area which is the index that allows developers to select which changes to include in a commit, offering precise control over the project's history.

References:

- Chacon, S., & Straub, B. (2014). *Pro Git*. Apress.
- Loeliger, J., & McCullough, M. (2012). *Version Control with Git*. O'Reilly Media.
- Driessen, V. (2010). *A successful Git branching model*. https://nvie.com/posts/a-successful-git-branching-model/
- Fowler, M. (2006). *Continuous Integration*. martinfowler.com.
- Ford, N., Parsons, R., & Kua, P. (2017). *Building Evolutionary Architectures: Support Constant Change*. O'Reilly Media.