Assignment 2

GIT & GITHUB

Name: Sasanka Sekhar Kundu

Roll No: 150096725118

Date of Submission: 11th November 2025

Git and GitHub Assignment:

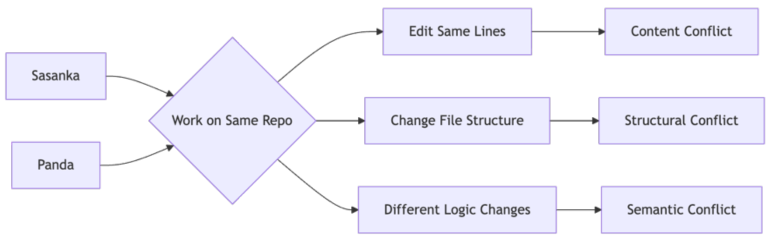
1. Define what is meant by a “conflict” in Git. Why do such conflicts occur in collaborative environments?

Answer: A conflict in Git happens when multiple contributors change the same section of a file or make opposing changes, and Git cannot decide which version should be kept. These conflicts often occur in team settings where several developers work at the same time. They usually happen when merging branches, rebasing, or pulling updates. In simple terms, conflicts arise when people need to decide how to combine changes. This situation shows both the benefits of teamwork and the care needed to keep versions consistent.

2. Differentiate between the following types of conflicts:
   a) Content Conflict
   b) Structural Conflict
   c) Semantic Conflict
   (Give a short explanation)

Answer:

Type of Conflict	Explanation	Example
Content Conflict	Occurs when two contributors modify the same part of a file, leading Git to demand manual resolution. Competing edits collide, like two voices trying to rewrite the same verse.	Both developers change the same function lines in login.js, causing Git to flag the overlapping edits.
Structural Conflict	Happens when file-level operations clash — one rename, another deletion, or parallel structural changes. Architecture shifts without harmony.	Dev A renames app.css to main.css, while Dev B deletes app.css in another branch.
Semantic Conflict	Code merges cleanly, but logic breaks — meaning, not syntax, is in conflict. The build succeeds, but behavior crumbles.	One branch changes discount logic to 10%, another updates price calculation. After merge, discounts stack incorrectly.



3. Describe the role of a developer when a conflict arises in a shared repository. What are the key steps they should follow?

Answer: A developer takes care of code quality by reviewing conflicting sections and choosing the best solution. They need to compare versions, understand the context, work with teammates if necessary, and merge the right logic. After making the changes, they conduct thorough testing and documentation. Finally, the developer commits the changes, ensuring both technical quality and teamwork.

4. Discuss why effective communication is essential in resolving conflicts within a team. Give two examples of how poor communication can worsen conflicts.

Answer: Clear communication builds mutual understanding, prevents assumptions, and ensures that the right decisions are made with shared clarity. Without it, confusion spreads and trust weakens.

Examples:

- If two developers modify the same module without informing each other, overlapping work may lead to extensive conflicts and wasted effort.
- Miscommunication around deadlines or task ownership can result in rushed merges, sloppy resolutions, and repeated breakages in the codebase.

5. Conflict resolution is not just a technical activity but also a collaborative process." Justify this statement with suitable reasoning.

Answer: Resolving conflicts requires not just simple fixes but also group thinking, empathy, and a sense of shared responsibility for the codebase. Teams do well when they clarify their goals, discuss openly, and maintain consistency throughout the project. In this flow of teamwork, technology acts as a tool, while people make up the real foundation. Harmony in code reflects harmony in collaboration; this is a lasting lesson in craftsmanship.

6. List and explain any four best practices to minimize the chances of conflicts during software development.

Answer: The Four best practices to minimize the chances of conflicts during software development are :

- **Frequent Pulls and Syncing:** Keep the local repository updated to avoid lengthy divergence.
- **Feature Branch Discipline:** Use separate branches for features and merge incrementally to reduce overlap.
- **Modular Code Structure:** Design code in self-contained modules so multiple contributors can work independently.
- **Clear Task Ownership:** Define roles and allocate responsibilities to avoid dual edits on the same component.

7. Explain why testing and validation are necessary after conflict resolution.

Answer : After merging changes, hidden logical issues may appear even after fixing syntax errors. Testing ensures that functionality works and performance remains steady while Validation protects project quality, supporting the important field of engineering where carefulness maintains integrity. In simple terms, we trust but verify, keeping our standard of delivery high.

8. How would you plan your workflow to minimize merge conflicts? Explain your approach.

Answer: My Strategic Workflow to Minimize Merge Conflicts

- Short-Lived Feature Branches: Keep branches focused and brief. The longer a branch diverges, the greater the risk of a collision. Principle: Ship in increments; honour continuous improvement.
- Frequent Pulls From Main Branch: Regularly fetch and merge updates from the main branch. This keeps work aligned with the shared truth. Old wisdom: Stay close to the source; drifting invites chaos.
- Clear Task Ownership: Assign features or modules to individuals or sub-teams to avoid simultaneous edits in the same files. Clarity means fewer surprises.
- Use Pull Requests With Code Reviews: Every merge goes through peer review. This is where refinement happens, creating a culture of quality and learning. Collaboration is action in progress.
- Modular, Decoupled Code Structure: Design systems so changes affect isolated components. A clean design protects against future issues. Well-structured systems respect boundaries.
- Write Self-Documenting, Predictable Code :Consistent naming, formatting, and structure reduce merge noise. Consistency shows respect for your craft and classmates.
- Continuous Integration Pipelines: Automate tests and merges whenever possible; fail fast, fix early. CI tools act as vigilant guardians.

- Communicate Early & Often: Team communication prevents silent, overlapping efforts. No tool can rival human coordination.