Homework: Team Workflow with Git & GitHub

Goal

The goal of this homework is to make you confident working with **Git from the command line** and **GitHub in a team environment**.
After completing this assignment, you should feel comfortable with:

- basic Git operations
- working with branches
- resolving merge conflicts
- different ways of merging into the main branch
- collaborating via GitHub using pull requests and reviews

You are **not allowed to use GUI-only Git tools**. All Git work must be done from the **command line and GitHub GUI**.

Project base

You must use the following repository as a base for the assignment:

https://github.com/taltech-coding/vanemarendaja-borsibaar

Team setup

- Work in teams of **4–5 students (same setup as for main project can be used)**
- One person acts as a **team lead / integrator**
- All other members act as **contributors**

Once completed, send the submission to your mentor

Deadline: 26.01.2026

Part 1: Repository setup

Step 1: Fork the repository

- Create a **fork** of the provided repository into your own GitHub account
- One team member creates a **team repository** from the fork (this will be the main repo for the team)
- Add all team members as collaborators to this repository

Step 2: Clone the repository

- Each team member clones the team repository to their local machine
- All further work must be done using the command line

Step 3: Initial commit

- Create a new file TEAM.md
- Describe:
    - team name
    - team members and GitHub usernames
    - short description of your team workflow (branches, reviews, merges)
- Commit and push this file to the main branch

Part 2: Branching and parallel development

Step 4: Define features

As a team, decide on **at least three independent changes** to the project.
Examples (you may choose others):

- change or extend text/content in the project
- add or modify a simple feature
- refactor existing files
- improve structure or naming

Each feature must:

- be worked on by **one person**
- affect **real project files**

- require more than one commit

Step 5: Create feature branches

- Each developer creates their **own feature branch** from main
- Branch names should clearly describe the feature

Step 6: Work on features

- Make several commits while implementing the feature
- Commit messages must clearly describe what was changed
- Push the branch regularly to GitHub

Part 3: Creating and resolving conflicts

Step 7: Intentionally create a conflict

- At least **two team members** must modify the **same file and same part of the file**
- Push both branches without coordinating the changes
- This should result in a **merge conflict**

Step 8: Resolve the conflict

- Resolve the conflict **locally using the command line**
- Make sure the final result makes sense and the project still works
- Commit the conflict resolution
- Push the updated branch

Part 4: Pull requests and code reviews (in GitHub GUI)

Step 9: Open pull requests

For **each feature branch**:

- Open a pull request into main
- Write a clear description of:
    - what was changed
    - why the change was made

Step 10: Review pull requests

- Assign **at least one reviewer** to each pull request
- Reviewers must:
    - leave comments
    - request changes or ask questions
- Authors must:
    - respond to comments
    - make additional commits if needed

Part 5: Merging strategies (in GitHub GUI)

Step 11: Use different merge types

Your team must demonstrate **at least two different merge strategies** when merging into main:

- a regular merge commit
- a squash merge
- a rebase-based merge

Each merge must happen **via GitHub pull requests**, not direct pushes.

Step 12: Explain your choices

- Update TEAM.md
- Add a section explaining:
    - which merge strategies you used
    - why you chose them
    - what problems you encountered

Part 6: Final cleanup

Step 13: Repository hygiene

- Delete merged feature branches
- Ensure main contains all final changes

- Make sure commit history is readable

Step 14: Documentation

- Create a file HANDIN.md
- Include:
    - links to all pull requests
    - explanation of the conflict you had and how it was resolved
    - what each team member worked on

Submission (REQUIRED)

You must submit **one of the following**:

Option 1: Screenshots

- Screenshot of:
    - commit history
    - pull request list
    - at least one pull request with review comments

Option 2: Links

- Link to your team repository
- Links to all relevant pull requests

Important rules

- No direct pushes to main after Part 1
- No GUI-only Git tools, except Part 4 and Part 5
- No copying solutions from other teams