Study Point Assignment (20 SP): Version Control and Team Collaboration

Objective

You must demonstrate how to collaborate effectively on a software project using GitHub. The focus is on modern practices that improve developer experience (DX), developer efficiency, and team collaboration. You will end up with a template repository on GitHub that can be reused for future projects and a video that illustrates and discusses your work.

The use of Pull Requests (pre-integration code reviews) is mentioned in several of the tasks described below as part of a branching policy. If your team decides NOT to use mandatory Pull Requests, leave out the tasks that involve PR's. Instead, you must argue about your choice.[1]

Hand-in

Tuesday 23/9 Mail to tm@ek.dk: Team names, GitHub and video links

Tasks

1. Branching Strategy

- As a team, decide on a branching strategy (e.g., GitHub Flow, GitFlow, Trunk-Based).
- Document your choice in the repository's README and explain why you chose this strategy.

2. GitHub Projects & Issues

- Create a **GitHub Project (Kanban)** for your team.
- Use **Issues** to represent tasks, bugs, or features.
- Automate movement of Issues on the Project board using workflow rules:
    - When a PR is opened → move Issue to *In Progress*.
    - When PR is merged/closed → move Issue to *Done*.

3. Issue & PR Templates

- Add **Issue templates** for:
    - bug_report.md
    - feature_request.md
- Add a **Pull Request template** that includes:
    - Summary of changes
    - Linked Issue(s)
    - Checklist for tests/docs/labels

[1] You can read more about code reviews and pull requests here: Continuous Integration

4. Commit & PR Discipline

- Follow **good commit message practices** (imperative style, meaningful scope).
- Each PR should be small, focused, and linked to an Issue.
- Use **draft PRs** for work in progress.

5. CODEOWNERS & Reviews

- Add a .github/CODEOWNERS file to enforce automatic reviewer assignment.
- Require at least one approved review before merging (add a branch protection rule).

6. Labels & Labeler Workflow

- Define a consistent set of **labels** (bug, feature, documentation, priority:high, etc.).
- Configure a **GitHub Actions labeler workflow** so files/paths automatically get labels (e.g., changes in /docs/** → documentation).

7. Dev Container

- Create a .devcontainer/devcontainer.json so all developers have the same environment.
- Document how to open the project in a dev container and run the project.

8. Make the Repository a Template

- Configure your repository as a **template repository** (Settings → General → Template repository).
- Document in your README how someone could use your repo as a starting point for their own project.

9. Metrics & Reflection

- Track and reflect on **collaboration metrics**, such as:
    - Time from Issue creation → PR merge
    - Number of PRs merged per week

10. Contributing Guidelines

- Create a CONTRIBUTING.md file in the root of the repository.
- The file must include:
    1. **Getting started:** How to clone and set up the repo (including dev container instructions).
    2. **Branching rules** (e.g., always branch off main, naming conventions like feature/xyz or bugfix/abc).
    3. **Commit message guidelines** (imperative mood, reference Issues: Fixes #12).
    4. **How to create a Pull Request** (template, linking Issues, review process).
    5. **Code style rules** (if any, or link to guidelines/linter/formatter).
       Expected workflow (e.g. open Issue → branch → PR → review → merge → Issue closes).
- This file should serve as the **entry point** for any new developer who wants to contribute. It is supposed to remove friction for new contributors and reinforce discipline in commits, PRs, and collaboration.

11. README.md

- Create a README.md that explain what the repo is, how to get started, how the team works together, and how to reuse it.
  Table of Contents:
  1. About the Project
  2. Branching Strategy decision
  3. Getting Started (how to clone it)
  4. Automation & Workflows
  5. Metrics
  6. License (to tell anyone who looks at the repo what they are allowed to do with the code.)

Deliverables

- **GitHub repository** (set as a template) including:
  - README.md & CONTRIBUTING.md
  - Project board + Issues with labels
  - Issue & PR templates
  - CODEOWNERS file
  - Labeler and metrics workflow
  - Dev container config

- **A video** (max 10 minutes) to demo and discuss how your practices are affected:
  - Is the workflow smooth and supportive?
  - Have you made extra enhancements?
  - Does automation (labels, Actions, board automation) save time?