CI Report

a. Summarise your continuous integration **method**(s) and approach(es), explaining why these are appropriate for the project. (5 marks, ≤ 1 page) Procedure (not exact tool)
b. Give a brief report on the actual continuous integration infrastructure you have set up for your project. (5 marks, ≤ 1 page) supporting infrastructure

The main aim of continuous integration is to integrate any modification to the codebase to the instead of periodically.

(in general: part a => general method (how CI supports the devs), part b => the actual tools and utilisation of those tools)

--- Apart from tests
We have also configured code style checking (checkstyle) and static analysis tool (error prone) to catch programming mistakes (at compile time, through the build tool (gradle), and the CI automaton)

--- Using CI
We could also deliver the product progressively,
(Minimum viable product)

Part A - Methods & Approaches

- To increase the development pace, team members are encouraged to push finished features into the central repository as soon as they are completed.
    - Off-loading tasks from developers (and their machines) to the CI automaton.
- To maintain the code quality, code linting is done
    - (formatting of the code is checked through the `checkstyle`)
- To make

- Our team uses a kanban for all the features and tasks that need to be done.
    - The kanban is integrated into the source code hosting facility for issue tracking

CI: What? - VCS, Automation, Testing, Integration Machine
Integration Machine: Event - Dev makes code integration

- We use a centralised repository for source code hosting.

Why
- Badges
- Code quality
    - Coding standards

        - Error checking on the automaton
            - `Error prone`
            - `SpotBugs`
        - Code linting
            - `checkstyle`
    - Code coverage
    - Deployment

(Web)Hooks provided by the online central repository hosting service, as well as automation service, also provided by the repo hosting service. (traditionally this would be a separate software e.g. Travis CI, Circle CI

Sometimes devs may work on a feature that may interact/interfere with another feature another team member is working on.

Team members are encouraged to push finished features into the central repository as soon as they are completed.
This ensures that everyone is able to work on the latest

Developers are encouraged to push the finished features into
And preferably not run all the tests locally, and offload the tests to the CI
This ensures agility and increases the speed of development
As well as taking the differences of development environments (such as machines) into account

(as CI build environment is consistent)

Note: GitHub Actions setup

- Build test (make sure the code still builds)
- Unit test (make sure the test passes)
- Make s
- Linting and formatting (make sure the code is up to standard)

Part B - Infrastructure

- We use Git extensively for version control.
- We use GitHub for source code hosting.
- We utilise GitHub Actions, which is a GitHub facility for CI and general automation for source code.

Source code push to github -> Create PR -> auto build in CI system -> run tests -> lint -> performance

https://intellij-support.jetbrains.com/hc/en-us/community/posts/206810905-Enabling-automatic-Git-Fetch

https://plugins.jetbrains.com/plugin/7499-gittoolbox