Code Inspection Checklist

1. Java & Typescript

General Review Checklist

Reviewer: \Reviewed: Code lines: lines x to y of Java class z

1. Variable Declaration

- ☑ Are variable names informative
- ☑ Are variable names unique (not confusing or similar)?
- ☑ Are variable names following chosen capitalization conventions (camel case)?
- ☑ Are variables properly initialized?
- ☑ Are variables labelled as private or public based on their use?
- ☑ Is every declared variable used?
- ☑ Is there excessive use of unnecessary temporary variables?

2. Methods and method signatures including return and input types

- ☑ Do method names reflect method functionalities?
- ☑ Do method expected return values match the intended use of the return value?
- ☑ Do methods have safeguards for problematic/unexpected input?
- ☑ Is there a high cohesion between the methods within the same class?

3. Class definitions and grouping into packages (Java)

- ☑ Do object classes reflect the required elements of the program?
- ☑ Are classes placed in the appropriate packages reflecting the nature of their use?
- ☑ Are classes in different packages loosely coupled?

4. Control flow Defects

- ☑ Are Switch cases used instead of if/else blocks when appropriate?
- ☑ Are While loops successfully terminated to avoid infinite loops?
- ☑ Are control flows used efficiently in the handling of erroneous input?

- ☑ Are loop variables declared properly so that their scopes are only as big as necessary?
- ☑ Are there checks for edge cases (out of bounds) for For loops?
- ☑ Are there else blocks used for every if condition to ensure no case goes unhandled?

5. Code style & practices

- ☑ Is code consistently indented, spaced, and formatted?
- ☑ Code is well documented using inline comments and docstrings.
- ☑ Are Expensive operations minimized (shallow object copies replacing deep ones if possible)
- ☑ Are generics used where possible to improve code readability & reduce complexity?

Typescript Review Checklist

- ☑ Are type annotations and inference used?
- ☑ Are strict Null checks in place?

Use type inference, type annotation, and generics.

2. Front End (TS & ReactJS)

- ☑ Is the single responsibility principle applied to react components?
- ☑ Are container components used strictly for managing state and business logic?
- ☑ Are presentational components used for UI rendering and logic strictly?
- ☑ Are related components, styles, and assets grouped within same directory?
- ☑ Are functional components used instead of class components if possible?
- ☑ Are React hooks used to manage & control state & effects in functional components?
- ☑ Are hooks called at the top level of functional components?
- ☑ Are unnecessary re-renders avoided?
- ☑ Are local component states prioritized for UI-specific state matters?