Workflows, Code Conventions, and Quality Assurance

Workflows

Work due is documented using well-named and tagged GitLab Issues. Team members are to create short-lived feature branches that address a specific Issue, and open Merge Requests (MRs) to bring changes back to `main`. MRs must reference the Issue being addressed so they can be closed automatically. Issue and MR titles must be imperative sentences (e.g., Fix db config), and branch names should be in lower-kebab-case (e.g., `hotfix/db-config`). Importantly, branches may only branch off `main`. (Atlassian, n.d.)

Commits should be atomic, easily revertible, separable by scope, and must follow the Conventional Commits specification (Conventional Commits, 2023)

Code conventions

Code consistency is enforced across the team using ESLint and Prettier. The CI pipeline is designed to fail if linting fails, and it is recommended to run `npm run lint` before committing code.

Code must be written in a functional style (e.g., React functional components with hooks are to be used instead of class components) with a focus on immutability (i.e., `const` variables are preferred over `let`). The arrow function style is encouraged, but other forms may be used at the discretion of the author if they improve code readability.

Components, types, and interfaces must use `UpperCamelCase`, variables must use `lowerCamelCase`, and files should be named using `lower-kebab-case` (Mozilla, 2023)

Comments are encouraged where needed and should address the "why" behind a block of code.

What/how [X]	Why [✓]

`// setting user field to false`
`user.admin = false;`
	`// field may be undefined if user has`
	`// just signed up`
	`user.admin = false;`

Quality assurance

To minimise the risk of committing buggy code to main, a MR can be merged only once ALL these requirements are met:

- Passing CI/CD pipeline
- Approval from at least one member who has not authored/contributed to the MR
- Approval from the a11y champion, if not an author/contributor
- Resolution of all threads in the MR and linked issue

Team members are expected to test their code and account for edge cases. On the front-end, this could involve testing the UI against illegal/malformed states. On the server-side, team members must account for malicious/missing API request parameters and fail gracefully. Writing unit tests is encouraged but not expected or required.

More importantly, access to data and resources must be delegated with the principle of least privilege in mind, i.e., no access > admins > authenticated users > public access.

Further reading

Atlassian, n.d. *Git Feature Branch Workflow.* [Online]
Available at: https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow

Conventional Commits, 2023. *Specification.* [Online]
Available at: https://www.conventionalcommits.org/en/v1.0.0/#summary

Mozilla, 2023. *Guidelines for styling JavaScript code.* [Online]
Available at: https://developer.mozilla.org/docs/MDN/Writing_guidelines/Writing_style_guide/Code_style_guide/JavaScript