Frontend Coding Standard

Overview

These guidelines, principles and standards allow us to:

- Consistent code quality across all front-end project
- Multiple developers can work on a single code base at the same time in the same way
- Write code that is less prone to errors and regressions and easier to understand and debug
- Write code that supports reuse
- This is must-read for all developers (internal and external) working on front-end project

Code Formatting and Style

- Tab size should be 2. Use space instead of tabs.
- The code format is governed by `prettier`

File Naming

- Types, interfaces and classes should be `CamelCase`
- Functions and variables should be `camelCase`
- React function component should be `CamelCase`
- Constant should be in `PASCAL_CASE`
- Folder and file name should be `kebab-case` unless the file is a react component

React Component

- React component should be the default export of its own file, the file name should be the same with the component. e.g. a component called `RecordingList` should be the default export of the file `RecordingList.tsx` and imported like this `import RecordingList from './somepath/RecordingList'`
- If a react component is a fully functional page, it should be put in `views` folder, and its name should end with `View`, e.g. `FooView`; if it is a component that can be reused, it should be put in `components` folder

Function Codes Structure

- API related function should be in `api.ts`
- Other utility functions should be in `utils.ts`
- The interface for backend api schema should be in `entity.ts`

Error Handling

- All API calls and other functions that can raise errors should have proper error handling
- Use try-catch statements to catch or handle runtime errors

Comment

- Comment is used when code cannot explain itself clearly. Concise and explicit languages should be used.
- For util and api functions, a JSDoc should be added to explain how the function works.