1. **Developers Should Use the Following Tools in the Development Environment:**

*a)* [[Prettier]](https://prettier.io/): *A tool that automatically formats code, ensuring consistent style across the project.*

*b)* [[ESLint]](https://eslint.org/): *A powerful tool for identifying and fixing common issues in JavaScript code.*

 *C) InFeature* [[Huskey]](https://github.com/typicode/husky%22%20%5Cl%20%22readme): is used for setting up Git hooks, which automate tasks like running tests or linting code before commits or pushes, ensuring code quality throughout development.

**Optional:**

*a)* [[Git Lens]](https://marketplace.visualstudio.com/items?itemName=eamodio.gitlens): *A powerful extension for Visual Studio Code, enhancing Git workflow by providing advanced features and insights within the editor***.**

**JS and JSX Standardization:**

1. Form handling should be managed by **control components**. Developers should use libraries**(Formik and Yup) or formData** as per project requirements.
2. JSX file names should **begin** with **a capital letter (pascal case)**, while **folder** and **JS** file names should be in **camel case**.
3. Developers should use **‘let and const’** to declare variables, functions, etc. The variable names should be in **camelCase**.
4. Developers should use **custom hooks/function** if the functionality is repeated**.**
5. Usage of **ES6** functions and keywords as much as we can.
6. Usage of **Ternary operator** (ES6) if condition matched.
7. Error handling **(try-catch)** compulsory.
8. **Meaningful** names for variables and functions, and avoid spelling errors.
9. **Explanatory comments for complex logic/algorithms.**
10. Developers should follow **React and software development** best practices**:**
	1. *Developers should adhere to the* ***DRY*** *(Don't Repeat Yourself) methodology****.***
	2. *Developers should* ***split*** *the code according to the requirements.*

**Html and Css Standardization:**

1. Custom className should be in **block/snake case**.
	1. *Ex: ’nav\_hero\_section’,’fotter\_section’*
2. Developers should double-check classNames before writing styles or using in-built styles to ensure there are **no CSS regressions**.
3. Avoid Using **0px** Instead of **0** in stylesheet.

*Reason - totally unnecessary which increases your file sizes for no reason.*

1. Avoid **inline Style/CSS** at max.
2. Use Global **Generic style classes** as much as we can.

*Reason - shorten the codebase, easy to debug and fair usage of code*

*For example - for align center we can create and use class like below -*

***.tc {text-align: center}***

1. Avoid **Internal style** code, it affects **SEO** friendly websites.
2. Use **Block v/s Inline** HTML elements wisely and accordingly.
3. Usage of **Semantic Tags** instead of repeating Div and span again and again for **better visibility** in code and **SEO** prospective too.
4. Don’t use**!important*;***in Code anywhere.

*Reason - It’s completely damage the priority level of your style, if its very urgent then use it for component level only, don’t ever use !Important in global style.*

1. **Should be Generic Code.**
	1. Toaster Options
	2. Logout functionality
	3. API calls error handling - middleware
	4. Helper functions
	5. !important in css - minimum
2. Usage of **Alt** Attribute in **Img**

*Best practice for SEO friendly website and in case Image is broken from server.*

**Git and Build Standardization:**

1. **PR merging Process:**



1. Developers must ensure that all comments are **removed** before **raising** a **PR**. If commented code is **necessary** for the feature, developers should **provide** an explanatory **comment.**
2. *Implementation of* ***CI/CD*** *for the development server* ***reduces*** *developer time in generating builds and deployments, thus enhancing developer* ***productivity****.*
3. **Standardization of Git Branches, Commits, and PR Messages:**

a) Branches should be named in a structured manner to convey their purpose:

 **- Case 1:** *For bug fixes, the branch name should start with* ***'bug-fix/****' followed by the* ***feature, module, or ticket ID****.*

 **- Case 2***: For UI development, the branch name should start with* ***'UI/'*** *followed by the* ***feature, module, or ticket ID****.*

**- Case 3:** *For API integration, the branch name should start with* ***'API/'*** *followed by the* ***feature, module, or ticket ID****.*

**- Case 4:** For fixing critical bugs, the branch name should start with '**hot- bug-fix/'** followed by the **feature, module or ticket ID**.

*b) Developers should provide a description for every PR.*

*c) Developers should include proper comments with each PR.*