Guidelines for developers to write UI code:-

1. **Code Structure and Organization**

   Component Structure: Components should be modular, reusable, and follow a consistent naming convention.All New file should be in functional components(React Hooks).
   File Structure: The file and folder organization should be logical and aligned with the project.

   Example:-

```
src/
```

```
├── components/
│   ├── Header/
│   │   ├── Header.js
│   │   └── Header.css
│   ├── UserProfile/
│   │   ├── UserProfile.js
│   │   └── UserProfile.css
├── App.js
└── index.js
```

   SOURCE:- File Structure – React

2. **Code Readability**

- **Code Comments:** Comments should be clear and meaningful.
- **Naming Conventions:** Variables, functions, and components should have clear and descriptive names.
- **Components**: Use PascalCase for React components. For example, Header.js, UserProfile.js.
- **Files**: Use camelCase for non-component files like services, utils, hooks, etc. For example, apiService.js, useAuth.js.

   SOURCE: JSX In Depth – React

- **Log Removal:** After Unit test,Remove console.log which are printing value of the variables.

   [debug log,errorlog and inf log] - with or without source map -- Need to do POC on this

3.Performance

- **Rendering Optimization**: Use memoization (React.memo, useMemo, useCallback) where appropriate to avoid unnecessary re-renders.
- **Component Splitting:** Large components should be split into smaller, manageable components.file should not exceed more than 1000 lines

  NOTE:-there are no strict guidelines or hard limits on the number of lines of code in a file
- **Code Splitting:** Use dynamic imports and lazy loading to improve initial load time.
- **State Management:** Ensure efficient state management. Use React's Context API, Redux, or similar libraries as needed.
- **Avoid Multiple API Calls:**- In Network,Cross-check whether same API's are calling multiple times

4.**Error Handling**

- **Prop/state Validation:** check null ,undefined ,empty string or length if it is array variable.- create a validation function in helpers.js and use it
- **Try-Catch Blocks:** Use try-catch blocks for asynchronous operations and API calls [for all functions].

5.Testing

- **Code Coverage**: Aim for both high code data and low code data coverage with tests.
  Manual Testing: Conduct manual testing to catch visual or usability issues.on your local check all ut cases and make PR
- **Test Your Component:** Once your component is implemented, test it thoroughly to ensure it behaves as expected.(try to test use cases using testing libraries like Jest and Enzyme)

  Note - Someone needs to do POC for this

6.Security

- **Secure Communication:** Use HTTPS for all API calls and ensure secure token storage.
- **Environment Variables:** Keep sensitive information out of source code and use environment variables
- **Encryption/Decryption:** All sensitive Data should be encrypted

7.Dependencies and Build Configuration

- **Dependency Management:** Keep dependencies up to date and avoid unused dependencies.
- **Bundle Size:** Optimize the bundle size using tools like Webpack or Rollup.
- **Production Build:** Confirm that the code is built for production with proper minification and optimization.

8.Version Control and Collaboration

- **Git Practices:** Ensure proper use of version control, including meaningful commit messages and branches.[git standards]
- **Code Reviews:** Conduct thorough code reviews to catch issues and maintain quality using JSlist in vscode or using NPM.

  NOTE:- DOING POC on this need some time to implement.
- **Collaboration Tools:** Use JIRA collaboration tools to track issues, pull requests, and code discussions.

9.**Documentation**

Component Documentation: Provide documentation for each component, describing its purpose and usage.

NOTE :- And also below points need to be follow:-

1. we can stop using class based components
2. stop usage of var ,Instead we have to use let and const
3. Replace for loop with for in loop and for of loop
4. call all apis using redux . stop using buildaxios or local api calls
5. reusability of functions to be checked and moved to helpers
6. resuability of code to be checked in css