Style Guide

File and Folder Structure

```
MoCode
|-- .firebase/
|     |-- hosting.cache
|-- nodes_modules/
|-- public/
|     |-- 404.html
|     |-- favicon.ico
|     |-- index.html
|     |-- manifest.json
|     |-- robots.txt
|-- src/
|     |-- components/
|     |     |-- MyComponent/
|     |           |-- ConfirmStatus.css
|     |           |-- ConfirmStatus.js
|     |           |-- Footer.css
|     |           |-- Footer.js
|     |           |-- NavBar.css
|     |           |-- NavBar.js
|     |           |-- NewUserInfo.css
|     |           |-- NewUserInfo.js
|     |           |-- ProblemHistory.css
|     |           |-- ProblemHistory.js
|     |           |-- ProblemRec.css
|     |           |-- ProblemRec.js
|     |           |-- SignIn.css
|     |           |-- SignIn.js
|     |           |-- Stats.css
|     |           |-- Stats.js
|     |           |-- Timer.css
|     |           |-- Timer.js
|     |-- extra/
|     |     |-- profile_files/
|     |           |-- bundle.js
|     |           |-- js
|     |           |-- profile_icon.svg
|     |     |-- svgs and images
|     |-- pages/
|     |     |-- HomePage/
|     |           |-- Home.js
|     |           |-- Home.css
```

```
|     |           |-- Profile.js
|     |           |-- Profile.css
|     |-- App.css
|     |-- App.test.js
|     |-- AuthContext.js
|     |-- firebase.js
|     |-- index.css
|     |-- index.js
|     |-- problem.json
|     |-- reportWebVitals.js
|     |-- scraper.py
|     |-- setupTests.js
|     |-- Signin.test.js
|-- .firebaserc
|-- .gitignore
|-- firebase.json
|-- LICENSE
|-- package-lock.json
|-- package.json
|-- README.md
```

- With the way our files and folders are structured, we created a “pages” folder for the different pages that are present such as the Home and Profile pages
- There is another folder for components that focuses on making sure that the different components are divided up and easily accessible to use.
- There is an “extra” folder that provides resources for the other code such as images and more
- All unit tests are in `src/`

Naming Conventions

- Use PascalCase for component names
    - Make sure the names are specific to what the function does
- Use camelCase for variable and function names.

Component Structure

- Use functional components by default.
- Use arrow functions for component declarations.

```
// Functional Component
const MyComponent = () => {
  // ...
};
```

Here is an example of how we formatted the functional components:

```
const handleRecommendClick = async () => {
  // ...
}
```

With the function names and more, we used our naming conventions and focused on making sure that it followed the formatting conventions.

Spacing

- Four space indentation
- One space before and after operators
- One space after commas

Comments

- Provide comments for complex logic or non-trivial code.
- Keep comments up-to-date to reflect code changes.
- Keep comments focusing on what the code does
    - Easier for future coders to understand the code