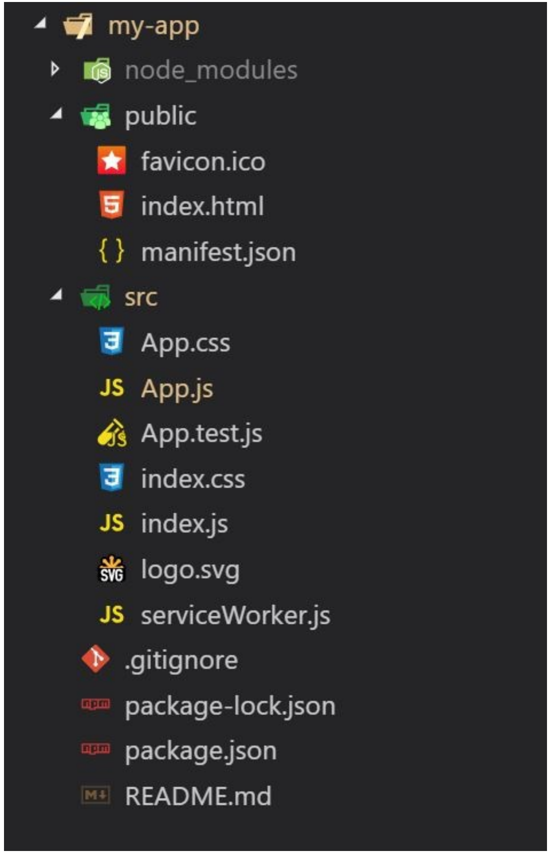
Directory Structure



```
my-app
  node_modules
  public
    favicon.ico
    index.html
    manifest.json
  src
    App.css
    App.js
    App.test.js
    index.css
    index.js
    logo.svg
    serviceWorker.js
  .gitignore
  package-lock.json
  package.json
  README.md
```

Directory Structure

- **my-app** – This is your Project Name
  - node_modules – It contains all packages and dependencies installed.
  - public -
    - favicon.ico – It's a favicon for website
    - index.html – This file holds the HTML template of our app.
    - manifest.json - manifest.json provides metadata used when your web app is installed on a user's mobile device or desktop.
  - src
    - App.css – It's a css file related to App.js but its global.
    - App.js – It's Parent component of your react app
    - App.test.js – Its Test environment
    - index.css - It's a css file related to index.js but its global
    - index.js – It's the JavaScript entry Point.
    - logo.svg – React logo file
    - serviceWorker.js – It can help to access website offline.
- .gitignore – It is used when you want to ignore git push.
- package-lock.json – Its version control package json file
- package,json – All dependencies mentioned in this file.
- README.md – It readme file

For the project to build, these files must exist with exact filenames

public Folder

If you put a file into the public folder, it will not be processed by Webpack (Webpack does not read the public folder; it will only read the files inside the src folder). Instead it will be copied into the build folder untouched.

To reference assets in the public folder, you need to use a special variable called PUBLIC_URL.

Only files inside the public folder will be accessible by %PUBLIC_URL% prefix.

```
<link rel="shortcut icon" href="%PUBLIC_URL%/favicon.ico">
```

When you run npm run build, Create React App will substitute %PUBLIC_URL% with a correct absolute path so your project works even if you use client-side routing or host it at a non-root URL.

public Folder

- You need a file with a specific name in the build output, such as manifest.webmanifest.
- You have thousands of images and need to dynamically reference their paths.
- You want to include a small script like pace.js outside of the bundled code.
- Some library may be incompatible with Webpack and you have no other option but to include it as a <script> tag.