AeroAspire - SDE Intern

Gokul Krishna S

Week 2 – Day 1 (September 29)

Task/Assignment :

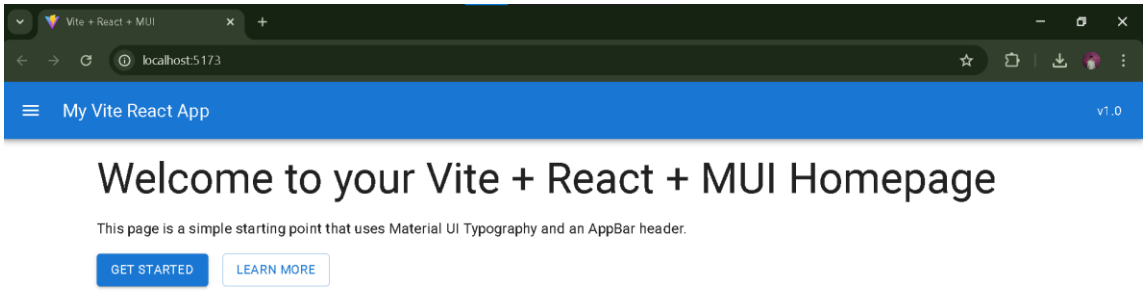
Scaffold app;
setup basic folder structure;
create homepage with MUI Typography and AppBar ;

To scaffold a React app with Vite, establish a folder structure, and create a homepage using MUI's Typography and AppBar Run the following in the terminal :

- npm create vite@latest my-react-app -- --template react
- cd my-react-app
- npm install
- npm install @mui/material @emotion/react @emotion/styled
- npm run dev

Inside the scr folder structure :

- src/
- components/
  AppBarHeader.jsx
- pages/
  Home.jsx
  App.jsx
  main.jsx



Questions/Reflections :

1. What files/folders does Vite produce and what is the build / dev flow?

   When creating a React app with Vite, it generates the following basic structure:

   - index.html (root HTML file outside src)
   - src/ folder containing main.jsx, App.jsx
   - package.json (scripts and dependencies)
   - vite.config.js (Vite configuration)
   - node_modules/ (after install)
   - Public/build assets on npm run build in dist/ folder.

   The dev flow uses npm run dev—it starts a fast development server using Vite. Build flow uses npm run build which bundles and optimizes the app for production in the dist/directory.

2. What is bundling / hot reloading. How does Vite help speed up development?

    - Bundling combines multiple files and assets into optimized bundles for efficient loading and deployment. Hot reloading instantly updates the browser when you change code, without restarting the app—so changes appear instantly.
    - Vite is extremely fast because it serves source files via native ES modules during development and only bundles for production.
    - Hot Module Replacement (HMR) is almost instant—making iterative development far faster than traditional bundlers like webpack.

3. Describe how React components are structured: what parent/child relationship, what props?

    - React components are modular JavaScript functions or classes. Components can nest other components—this creates parent/child relationships. Parents pass props (custom data and functions) to children.
    - Props let children display dynamic info or trigger parent logic. Each component manages its own logic, UI, and can be reused, keeping code organized and scalable.