AeroAspire

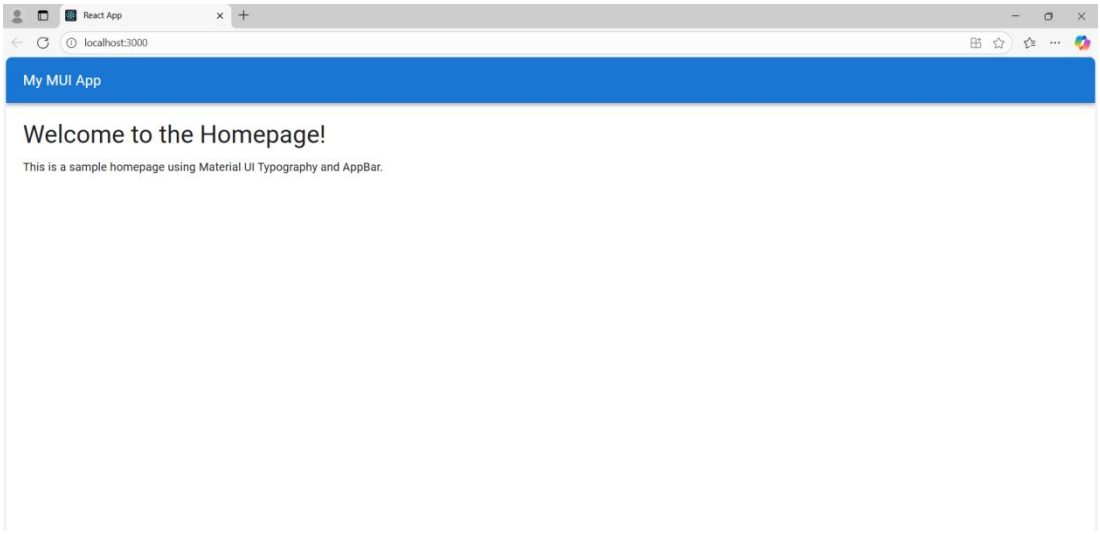
SDE Intern

Goutham V

Week 2 – Day1 (29th September)

Task:

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



Steps I Followed

1. **Installed Node.js and npm**

- Checked if Node.js and npm were installed using node -v and npm -v.
- If not installed, downloaded from the Node.js official website.

2. **Created a new React project**

- Ran this command in the terminal:

  bash
  ‘npx create-react-app my-react-app’

- Entered the project name when prompted.

3. **Installed Material UI packages**

- Ran:

  bash
  ‘npm install @mui/material @emotion/react @emotion/styled @mui/icons-material’

4. **Set up the folder structure**

- Inside src/, created these folders:
- components/ (for reusable UI components)
- pages/ (for main pages like Home)
- assets/ (for images and static files)
- Main files:
- App.js (main app component)
- index.js (entry point)

5. **Created the first components**

- In src/components/Header.js, made a Header using MUI AppBar and Typography.
- In src/pages/Home.js, made a Home page using MUI Typography and Box.
- Imported and used these components in App.js.

6. **Ran the development server**

- Started the app with:bash‘npm start’
- Opened the browser at http://localhost:3000 to see the homepage.

Reflection,

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

- When you scaffold a project with Vite, it creates a folder structure with:
    - index.html: The main HTML entry point.
    - src/: Contains source files, including main.jsx or main.tsx which bootstraps the React app.
    - node_modules/: For installed dependencies.
    - vite.config.js: Configuration for Vite.
    - package.json & package-lock.json: For project metadata and dependencies.
- **Development flow**: You run a local dev server with live reload (vite command), Vite serves modules as native ES modules dynamically without bundling.
- **Build flow**: When ready for production, running vite build bundles the code and assets into a dist/ folder optimized for deployment.
- This setup allows fast, efficient development with instant feedback and production-ready builds.

---

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

- **Bundling**: Combining many JavaScript files and dependencies into a single or few files for faster loading in browsers.
- **Hot Module Replacement (HMR)** or **Hot Reloading**: When you make changes to code, only updated files reload in the browser instead of a full page refresh.
- **Vite speeds up development by**:
    - Serving source files directly using native ES modules during development, avoiding full bundling and build.
    - Using fast, native ESM support in browsers for instant page update.
    - Using efficient dependency pre-bundling for production builds.

- Providing quick hot reloads to reflect your changes immediately, improving developer productivity.

---

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

- React apps are built with **components**, reusable pieces that manage their own UI and logic.
- Components can be **parent or child**: parents contain child components nested inside them.
- Each component returns JSX—HTML-like markup to render UI elements.
- **Props** are the way parents pass **data** or **parameters** to children, making components more flexible and reusable.
- For example, a Parent component can include a Child component and pass a name prop: <Child name="Goutham" />.
- Children receive props as input and use them to render dynamic content.
- This hierarchical, component-based structure helps to build scalable and maintainable UI.