AeroAspire - SDE Intern

JOHN NIKHIL G

Week 2 – Day 1 (September 29)

Task/Assignment :

- Scaffold app;
- Setup basic folder structure;
- Create homepage with MUI Typography and AppBar;

Questions/Reflections :

1. What files/folders does Vite generate and how does its dev/build workflow function?

When initializing a React app with **Vite**, you typically see this setup:

index.html – The root HTML file, placed outside src, used as the main entry point.

src/ – Contains React source files like main.jsx and App.jsx.

package.json – Holds scripts, dependencies, and project metadata.

vite.config.js – Configuration file for customizing how Vite behaves.

node_modules/ – Populated when dependencies are installed.

dist/ – Output folder created after running npm run build, with optimized, production-ready assets.

Development phase:

Running npm run dev spins up a fast development server that uses ES modules directly for near-instant startup and updates.

Build phase:

Running npm run build bundles and optimizes everything into the dist/ directory, ready for deployment.

---

2. What are bundling and hot reloading, and how does Vite enhance the developer experience?

Bundling: The process of collecting multiple JavaScript, CSS, and assets and packaging them into efficient bundles for faster delivery to the browser.

Hot Reloading (HMR): Updates only changed modules within the running app, so you see edits in real time without refreshing the whole page.

Vite advantages:

Uses **native ES modules** in the browser during development, skipping time-consuming pre-bundling.

Applies **HMR**, so changes take effect almost instantly.

For production, Vite hands off to Rollup to create highly optimized bundles.

---

3. How are React components organized in terms of hierarchy and props?

React components are isolated and reusable building blocks of UI.

Components often form a **tree structure** where a parent component can include one or more children.

Information or functions can be passed **downward through props**, giving children access to dynamic data or parent-driven behavior.

This design allows easy reusability, clearer separation of concerns, and better scalability of applications.