Dockerize a React Application with Multi-Stage Build

Objective

Learn how to create a production-ready Docker image for a React application using a multi-stage Docker build. This helps reduce image size, separate build dependencies from runtime, and prepare the app for deployment.

Task Description

1. Build a simple React application (for example, created using Create React App).
2. Write a multi-stage Dockerfile:
- Use a Node.js image as the first stage to install dependencies and build the React app.
- Use an Nginx image as the second stage to serve the compiled static files.
3. Add a .dockerignore file to exclude unnecessary files.
4. Build the Docker image locally and run it to test if the React app is correctly served on localhost.
5. Verify that the final image size is smaller than including all dev dependencies.

Code

Dockerfile

```
# Stage 1: Build the React app
FROM node:18 AS build

WORKDIR /app

COPY package*.json ./
RUN npm install

COPY . .
RUN npm run build

# Stage 2: Serve the app using Nginx
FROM nginx:alpine

COPY --from=build /app/build /usr/share/nginx/html

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

.dockerignore

```
node_modules
build
.dockerignore
Dockerfile
.git
.gitignore
```

Expected Output

- A working Docker container that serves the React app at http://localhost.
- Optimized Docker image with significantly smaller size.

- Clear separation between build and production stages in the Dockerfile.