# ***Experiment no 5.1***

# ***Aim***

To containerize a React.js application using **Docker Multi-Stage Build**, ensuring smaller image size, faster builds, and optimized deployment.

# **Theory**

### What is Dockerizing a React App?

Dockerizing means packaging a React app with all dependencies inside a Docker image so it can run anywhere.

React apps are frontend apps built using **Node.js** (for building) and usually served by a web server like **Nginx** in production.

### What is Multi-Stage Build?

* A **multi-stage build** in Docker lets you use multiple stages in a single Dockerfile.
* You can install heavy dependencies in the first stage (build stage) and copy only the final optimized build into the second stage (production stage).
* This reduces image size and improves efficiency.

### Why Multi-Stage Build for React?

1. React needs **Node.js** to build the production bundle.
2. But for serving static files, we don’t need Node.js — **Nginx is enough.**
3. Multi-Stage lets us:
   * Stage 1: Use Node.js to build.
   * Stage 2: Use Nginx to serve only the compiled /build folder.

# ***Code:-***

### Project Structure

my-react-app/

│── Dockerfile

│── package.json

│── src/

│── public/

### Dockerfile (Multi-Stage)

# ---------- Stage 1: Build ----------

FROM node:18 AS build

# Set working directory

WORKDIR /app

# Copy package.json and install deps

COPY package\*.json ./

RUN npm install

# Copy source code

COPY . .

# Build production-ready React app

RUN npm run build

# ---------- Stage 2: Production ----------

FROM nginx:alpine

# Copy built React app from Stage 1

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

# Expose port

EXPOSE 80

# Start Nginx

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

### Commands to Run

# Step 1: Build Docker image

docker build -t react-docker-app .

# Step 2: Run the container

docker run -p 8080:80 react-docker-app

Now, open http://localhost:8080 → You’ll see your React app running in a container.

# ***Learning Outcomes***

By the end of this, you will:

1. Understand **how to Dockerize a React app** for deployment.
2. Learn **Multi-Stage Build** to optimize image size.
3. Gain hands-on skills in **serving React with Nginx** instead of Node.js.
4. Recognize the difference between **development dependencies (Node.js)** **and production needs (Nginx).**
5. Be able to deploy the containerized React app to **any environment (local, cloud, Kubernetes, etc.).**