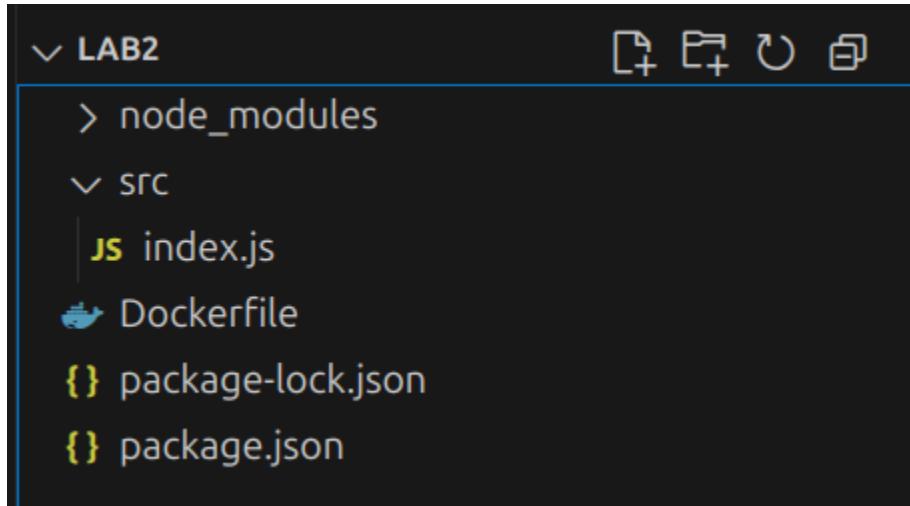


## Program 2

### Develop a Multi-stage Dockerfile for Container Orchestration

Step 1 : Create folder “lab2” , navigate to the folder in terminal and run “npm init -y” and “install express” commands.

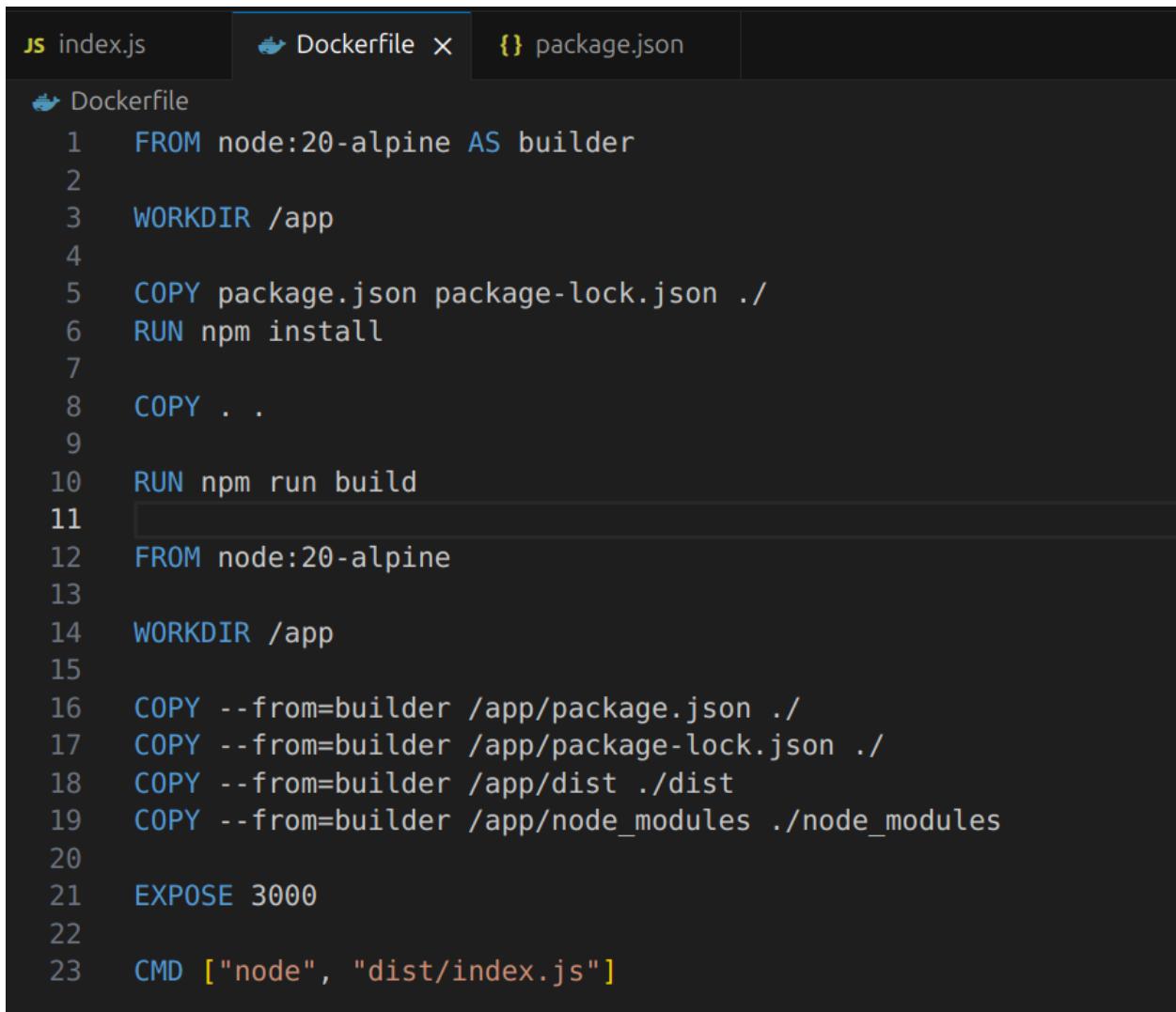


Step 2 : Create “index.js” file inside “src” folder.

```
JS index.js  X Dockerfile  {} package.json

src > JS index.js > ...
1  const express = require('express');
2  const app = express();
3  const PORT = 3000;
4
5  app.get('/', (req, res) => {
6    |   res.send('Hello from multi-stage Docker!');
7  });
8
9  app.listen(PORT, () => {
10 |   console.log(`Server running on port ${PORT}`);
11});|
```

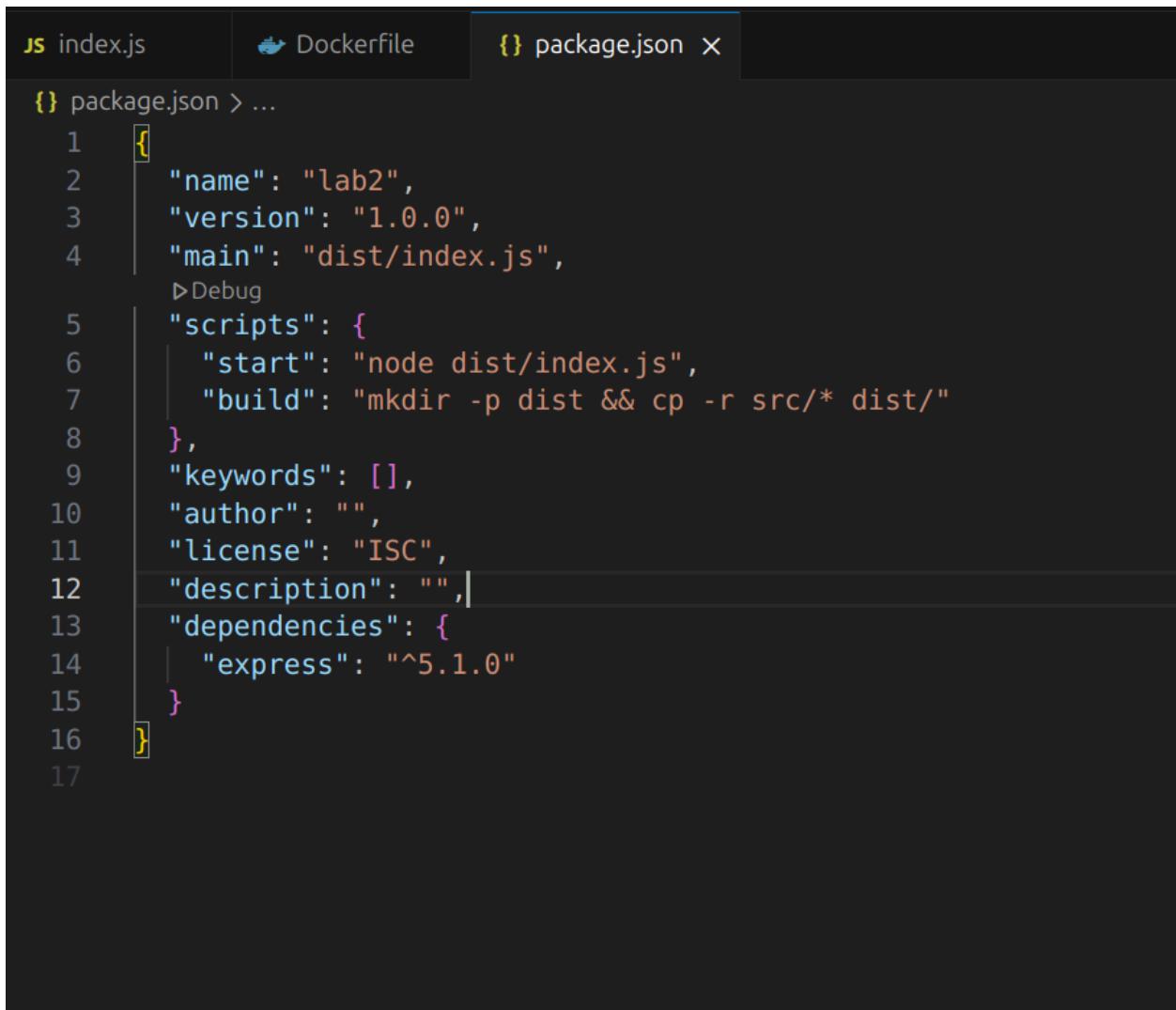
### Step 3 : Create Dockerfile



The screenshot shows a code editor interface with three tabs at the top: 'index.js', 'Dockerfile X', and 'package.json'. The 'Dockerfile' tab is currently active, displaying a Dockerfile with the following content:

```
FROM node:20-alpine AS builder
WORKDIR /app
COPY package.json package-lock.json ./
RUN npm install
COPY . .
RUN npm run build
FROM node:20-alpine
WORKDIR /app
COPY --from=builder /app/package.json ./
COPY --from=builder /app/package-lock.json ./
COPY --from=builder /app/dist ./dist
COPY --from=builder /app/node_modules ./node_modules
EXPOSE 3000
CMD [ "node", "dist/index.js" ]
```

Step 4 : Do necessary modifications in package.json file.



The screenshot shows a code editor interface with three tabs at the top: 'index.js', 'Dockerfile', and 'package.json'. The 'package.json' tab is active and selected. The code in the editor is a JSON object representing a package configuration:

```
{  
  "name": "lab2",  
  "version": "1.0.0",  
  "main": "dist/index.js",  
  "scripts": {  
    "start": "node dist/index.js",  
    "build": "mkdir -p dist && cp -r src/* dist/"  
  },  
  "keywords": [],  
  "author": "",  
  "license": "ISC",  
  "description": "",  
  "dependencies": {  
    "express": "^5.1.0"  
  }  
}
```

Step 5 : Build image

Docker build -t lab2image .

```
rv24mc094_sangeethan@sangeetha-n-ThinkPad-X1-Carbon-6th:~/Desktop/Devops_lab/lab2$ docker build -t lab2image .
[+] Building 6.2s (15/15) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 425B
=> [internal] load metadata for docker.io/library/node:20-alpine
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [builder 1/6] FROM docker.io/library/node:20-alpine@sha256:6178e78b972f79c335df281f4b7674a2d85071aae2af020ffa39f0a770265435
=> [internal] load build context
=> => transferring context: 43.88kB
=> CACHED [builder 2/6] WORKDIR /app
=> CACHED [builder 3/6] COPY package.json package-lock.json .
=> CACHED [builder 4/6] RUN npm install
=> [builder 5/6] COPY .
=> [builder 6/6] RUN npm run build
=> CACHED [stage-1 3/6] COPY --from=builder /app/package.json .
=> CACHED [stage-1 4/6] COPY --from=builder /app/package-lock.json .
=> [stage-1 5/6] COPY --from=builder /app/dist ./dist
=> [stage-1 6/6] COPY --from=builder /app/node_modules ./node_modules
=> => exporting layers
=> => writing image sha256:28b5ff338c8c99ed54520f734cd083af1e88787ca2046f69c41b2c6fce8f2ef4
=> => naming to docker.io/library/lab2image
=> docker:default
```

## Step 6 : Run Container

Docker run -p 3000:3000 lab2image

```
rv24mc094_sangeethan@sangeetha-n-ThinkPad-X1-Carbon-6th:~/Desktop/Devops_lab/lab2$ docker run -p 3000:3000 lab2image
Server running on port 3000
```

## Step 7 : Test Container

