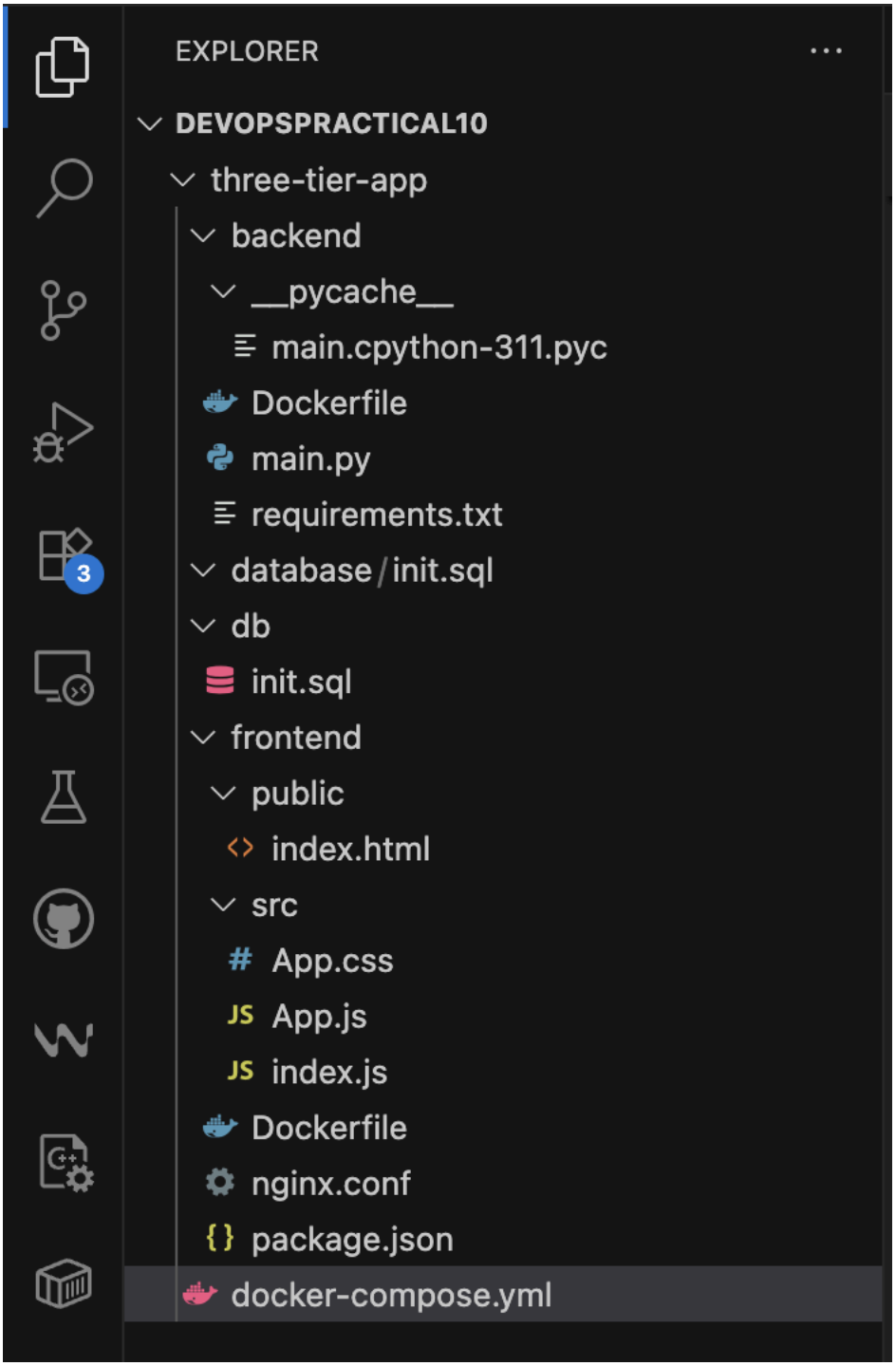
Practical 10

Aim:



Dockerfile:

```
# Build stage
FROM node:18-alpine as build
WORKDIR /app
# Copy package files
COPY package*.json ./
```

```
# Install dependencies
RUN npm install

# Copy source code
COPY . .

# Build the app
RUN npm run build

# Production stage
FROM nginx:alpine

# Copy built app from build stage
COPY --from=build /app/build /usr/share/nginx/html

# Copy custom nginx config
COPY nginx.conf /etc/nginx/nginx.conf

# Expose port
EXPOSE 80

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

requirements.txt:

```
fastapi==0.104.1
uvicorn==0.24.0
psycopg2-binary==2.9.7
pydantic==2.5.0
```

init.sql:

```
-- Initialize the database with sample data
CREATE TABLE IF NOT EXISTS users (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Insert some sample data
INSERT INTO users (name, email) VALUES
    ('John Doe', 'john.doe@example.com'),
    ('Jane Smith', 'jane.smith@example.com'),
    ('Bob Johnson', 'bob.johnson@example.com')
ON CONFLICT (email) DO NOTHING;
```

Index.html:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta name="description" content="3-Tier Application Demo" />
    <title>3-Tier Application</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

index.js:

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';
import './App.css';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

Dockerfile:

```
# Build stage
FROM node:18-alpine as build

WORKDIR /app

# Copy package files
COPY package*.json ./

# Install dependencies
RUN npm install

# Copy source code
COPY . .
```

```
# Build the app
RUN npm run build

# Production stage
FROM nginx:alpine

# Copy built app from build stage
COPY --from=build /app/build /usr/share/nginx/html

# Copy custom nginx config
COPY nginx.conf /etc/nginx/nginx.conf

# Expose port
EXPOSE 80

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

Nginx.conf:

```
events {
    worker_connections 1024;
}

http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    server {
        listen 80;
        server_name localhost;

        root /usr/share/nginx/html;
        index index.html;

        # Handle React Router
        location / {
            try_files $uri $uri/ /index.html;
        }

        # Cache static assets
        location ~* \.(js|css|png|jpg|jpeg|gif|ico|svg)$ {
            expires 1y;
            add_header Cache-Control "public, immutable";
        }
    }
}
```

Package.json:

```
{
  "name": "3tier-frontend",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.16.4",
    "@testing-library/react": "^13.3.0",
    "@testing-library/user-event": "^13.5.0",
    "axios": "^1.6.0",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  },
  "proxy": "http://localhost:8000"
}
```

Docker-compose.yml:

```
version: '3.8'

services:
```

```
# PostgreSQL Database
db:
  image: postgres:15-alpine
  container_name: 3tier-db
  environment:
    POSTGRES_DB: appdb
    POSTGRES_USER: user
    POSTGRES_PASSWORD: password
  ports:
    - "5432:5432"
  volumes:
    - postgres_data:/var/lib/postgresql/data
  networks:
    - app-network
  healthcheck:
    test: ["CMD-SHELL", "pg_isready -U user -d appdb"]
    interval: 30s
    timeout: 10s
    retries: 3

backend:
  build: ./backend
  container_name: 3tier-backend
  environment:
    DATABASE_URL: postgresql://user:password@db:5432/appdb
  ports:
    - "8000:8000"
  depends_on:
    db:
      condition: service_healthy
  networks:
    - app-network
  volumes:
    - ./backend:/app

frontend:
  build: ./frontend
  container_name: 3tier-frontend
  environment:
    REACT_APP_API_URL: http://backend:8000
  ports:
    - "3000:80"
  depends_on:
    backend:
      condition: service_started
  networks:
    - app-network
```

```
volumes:
  postgres_data:

networks:
  app-network:
    driver: bridge
```

ON TERMINAL:
docker compose up --build

SCREENSHOTS: