Task:

Get the application Up and running in the docker compose .

Soln:

Docker Full Stack application

Frontend

1. Create a react app and install axios in it
2. Create docker file for react app
3.
```
FROM node:latest
WORKDIR /app
COPY . .
RUN npm install
EXPOSE 3000
CMD [ "npm", "start" ]
```
4. Create dockerignore to ignore the node_modules and .gitignore
5.
```
Dockerfile docker-frontend 1, U        .dockerignore U
ocker-frontend > .dockerignore
1    node_modules
```

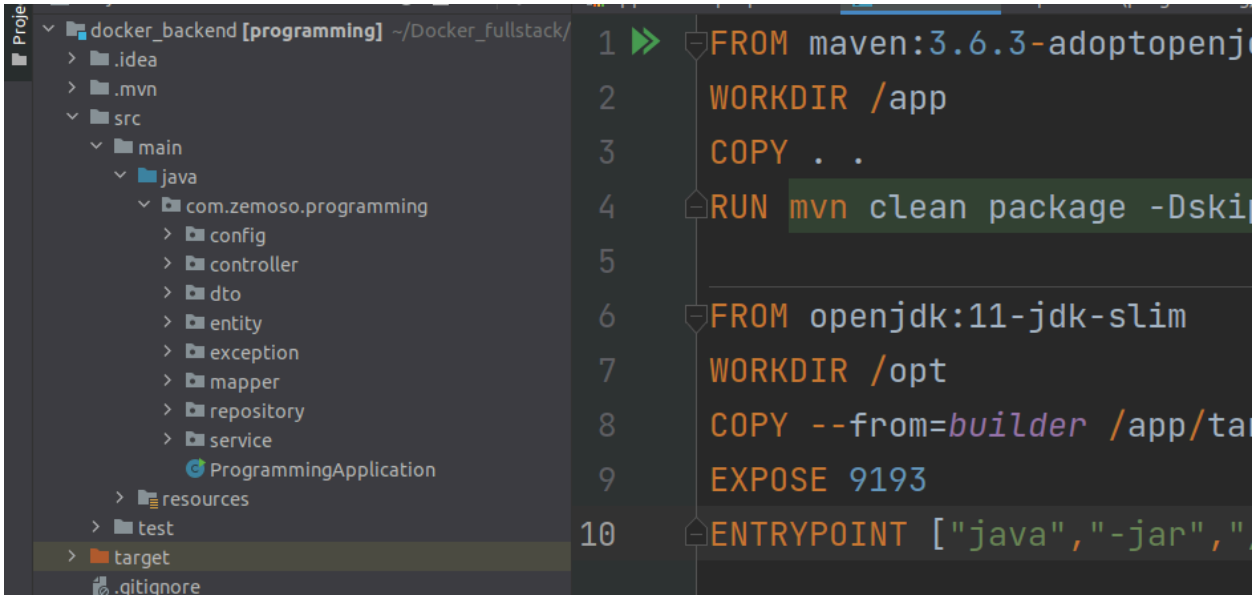
Backend

1. Create a spring boot project

2. Create dockerfile for backend

```
FROM maven:3.6.3-adoptopenjdk-11 AS builder
WORKDIR /app
COPY . .
RUN mvn clean package -DskipTests

FROM openjdk:11-jdk-slim
WORKDIR /opt
COPY --from=builder /app/target/*.jar /opt/app.jar
EXPOSE 9193
ENTRYPOINT ["java","-jar","/opt/app.jar"]
```

3.
4. We have to create multistage docker file because first stage we have to download dependencies and create jar file and second step we need to run the jar file. It will also reduce the image size as only the jar file will be present on image (no dependencies , build tools etc)
5. Create at one controller endpoint to see the api result
6. 

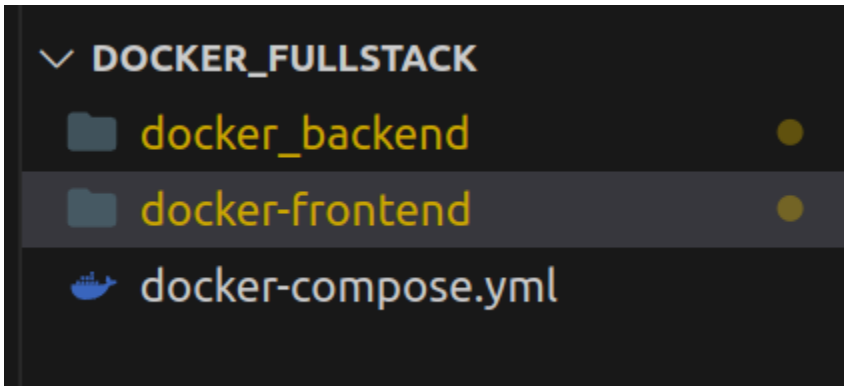
Database:

1. Create normal data base in root connection
2. Write the credential in src/java/resources in spring

3.
```
application.properties   Dockerfile   pom.xml (programming)
server.port=9193
spring.datasource.url=jdbc:mysql://database:3306/todo
spring.datasource.username=root
spring.datasource.password=Test@1234

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
#spring.jpa.hibernate.ddl-auto=update
spring.jpa.hibernate.ddl-auto=create
```
4. Here localhost is replaced by database because database is a name of service which will run prior to backend and backend is dependent on it

Docker Compose File Creation

1. Create the docker compose file outside the frontend/backend folder
2. 

3.

docker-frontend 1, U | .dockerignore U | Dockerfile docker_backend | docker-compose.yml

docker-compose.yml

```yaml
version: '3'
services:
  database:
    image: mysql:latest
    environment:
      - MYSQL_ROOT_PASSWORD=Test@1234
      - MYSQL_DATABASE=todo
    ports:
      - 3307:3306
    volumes:
      - todo-list-mysql-data:/var/lib/mysql
    networks:
      - main_network

  backend:
    build:
      dockerfile: Dockerfile
      context: ./docker_backend
    depends_on:
      - database
    restart: always
    environment:
      - SPRING_DATASOURCE_URL=jdbc:mysql://database:3306/todo
      - SPRING_DATASOURCE_USERNAME=root
```

4.

docker-frontend 1, U | .dockerignore U | Dockerfile docker_backend | docker-compose.yml

docker-compose.yml

```yaml
    environment:
      - SPRING_DATASOURCE_URL=jdbc:mysql://database:3306/todo
      - SPRING_DATASOURCE_USERNAME=root
      - SPRING_DATASOURCE_PASSWORD=Test@1234
    ports:
      - 9193:9193
    volumes:
      - backend_volume:/opt
    networks:
      - main_network
  frontend:
    build:
      dockerfile: Dockerfile
      context: ./docker-frontend
    depends_on:
      - backend
    ports:
      - 3000:3000
    networks:
      - main_network
    volumes:
      - ./docker-frontend:/app
volumes:
  todo-list-mysql-data:
```

5.

```yaml
volumes:
  todo-list-mysql-data:
  backend_volume:
networks:
  main_network:
```

First we have to start the database service with password and database name

Then start the backend service which depednss on database and place the env variables defined on database (spring boot)

6. **Then we have to start the frontend end service whic depends on backend**