Optimizing Deployment Speed and Reliability with DevOps

PHASE 3 - SOLUTION DEVELOPMENT AND TESTING

College Name: Dr. Sri Sri Sri Shivakumara Mahaswamy College of Engineering

Group Members:

- **Name: Binod Khatri Chatri**
  CAN ID Number: CAN_35805323
- **Name: Ashwini J**
  CAN ID Number: CAN_35777960
- **Name: Kavana M**
  CAN ID Number: CAN_35778518
- **Name: Shashi kumar M**
  CAN ID Number: CAN_35805208

SOLUTION DEVELOPMENT:

This phase deals with the implementation of the designed pipeline using source code and automation tools. Goal is to Set up the application (React + Node.js), containerize it with Docker, and prepare it for automated builds in a CI/CD pipeline.

Steps:

1. Set up the application:

Client (React): inside /client

Server (Express): inside /server

1.1 React App (Frontend)

to create clint : npx create-react-app client

1.2 Express App (Backend)

In server/ folder:

```
npm init -y
```

```
npm install express
```

server/index.js:

```
const express = require("express");
const app = express();
const PORT = process.env.PORT || 5000;

app.get("/", (req, res) => res.send("Server is running"));

app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

2. Create Dockerfiles

2.1 Frontend: client/Dockerfile

```
# Build stage
FROM node:18 as build
WORKDIR /app

RUN npm install
RUN npm run build

# Production stage
FROM nginx:alpine
COPY --from=build /app/build /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

2.2 Backend: server/Dockerfile

```
FROM node:18
WORKDIR /app
COPY . .
RUN npm install
```

```
EXPOSE 5000
CMD ["node", "index.js"]
```

3. Docker Compose (Optional but recommended)

docker-compose.yml:

```
version: "3.8"
services:
  client:
    build: ./client
    ports:
      - "3000:80"
    depends_on:
      - server

  server:
    build: ./server
    ports:
      - "5000:5000"
```

Run: docker-compose up --build

4. Set Up GitHub Actions (CI/CD)

.github/workflows/deploy.yml:

```
name: CI/CD Pipeline

on:
  push:
    branches: [ main ]

jobs:
  build-and-test:
    runs-on: ubuntu-latest
```

```
  services:
    docker:
      image: docker:20.10.16
      options: --privileged

  steps:
  - uses: actions/checkout@v3

  - name: Set up Docker Buildx
    uses: docker/setup-buildx-action@v3

  - name: Build Client Docker Image
    run: docker build -t devops-client ./client

  - name: Build Server Docker Image
    run: docker build -t devops-server ./server

  - name: Run containers
    run: docker-compose up -d

  - name: Wait for services to be ready
    run: sleep 10

  - name: Test Server
    run: curl http://localhost:5000
```

5. Add Development Scripts

client/package.json:

```
"scripts": {
  "start": "react-scripts start",
  "build": "react-scripts build",
```

```
  "test": "react-scripts test"
}
```

server/package.json:

```
"scripts": {
  "start": "node index.js",
  "test": "echo \"Add backend tests here\" && exit 0"
}
```

6. .dockerignore / .gitignore

client/.dockerignore and server/.dockerignore:

```
node_modules
build
npm-debug.log
```

containerized both apps

can run the whole system with one command (docker-compose up)

Can integrate CI/CD with GitHub Actions to build and test the containers on every push