Semester Project

Course Instructor
Sir Zaheer Sani

Submitted By
Yahya Rashid (22i-1413)

Members:

Saim Zia (22i-2661)

Yahya Rashid (22i-1413)

Date:
13-5-2025

Spring 2025

Department of Computer Science

FAST – National University of Computer & Emerging Sciences

Table of Content

Gym Management System - Project Report

1. Operating System Selection and Environment Setup

Operating System Justification

- **Selected OS**: Ubuntu 24.04.1 LTS
- **Justification**:
    1. Native Docker Support: Linux provides native Docker support, eliminating the need for virtualization layers
    2. Performance: Direct hardware access results in better performance for containers and Kubernetes
    3. Package Management: Advanced package management through apt makes tool installation straightforward
    4. Community Support: Extensive community support for DevOps tools and containerization
    5. Resource Efficiency: Lower overhead compared to running on Windows with WSL or VMs

Advantages

1. Better Performance:
    - Native container support without virtualization
    - Direct hardware access results in better performance for containers and Kubernetes
    - Faster build and deployment times
2. Tool Compatibility:
    - Native support for Docker and Kubernetes tools
    - Better compatibility with DevOps tools
    - Seamless integration with CI/CD pipelines
3. Development Workflow:
    - Faster container builds and deployments
    - Better resource utilization
    - More efficient local development experience

Challenges Encountered

1. Initial Setup Learning Curve:
    - Required familiarity with Linux commands
    - Understanding of system permissions
    - Configuration of development environment
2. System Configuration:
    - Manual configuration of network settings
    - Setting up proper permissions for Docker and Kubernetes
    - Managing system resources

Environment Specifications

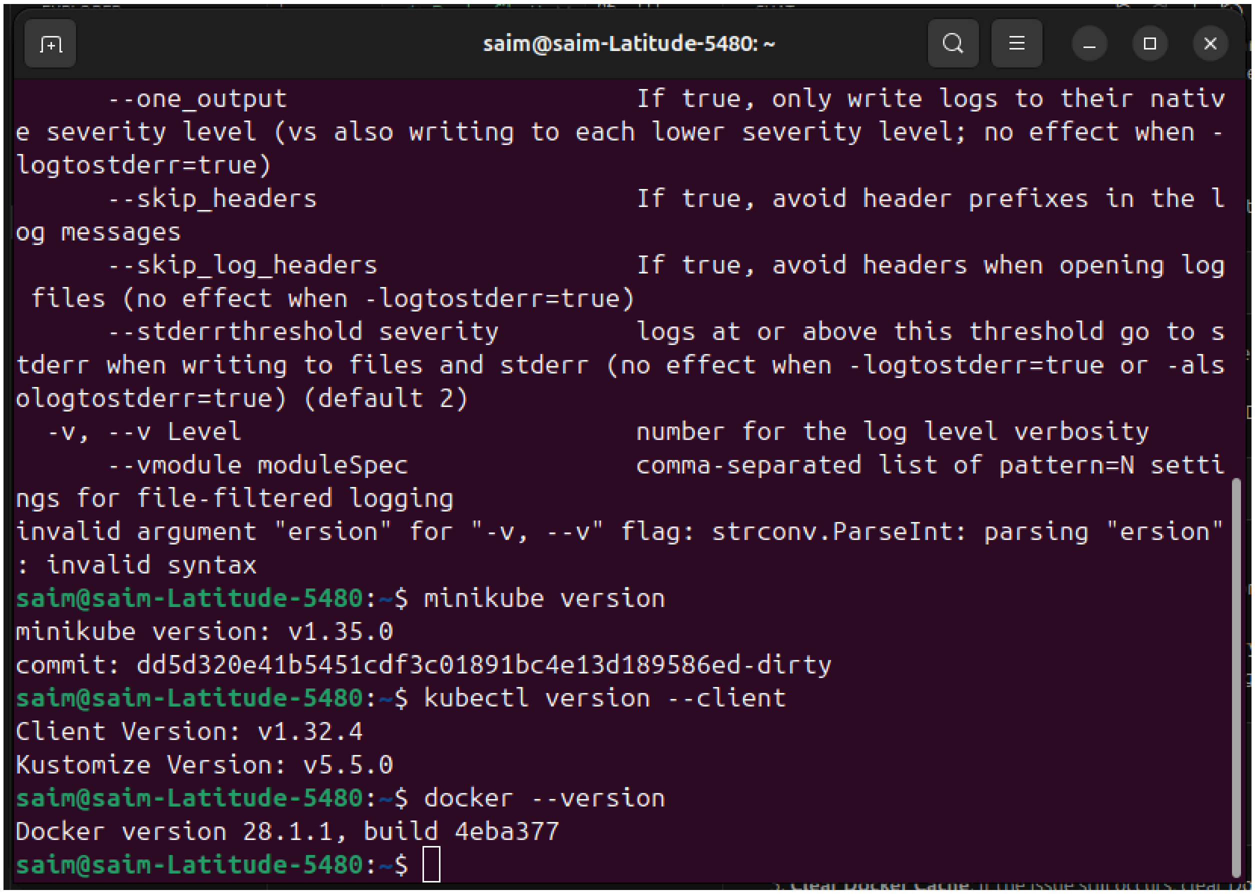
- **OS Version**: Ubuntu 24.04.1 LTS (Noble)
- **Kernel Version**: 6.11.0-25-generic
- **Docker Version**: 28.1.1
- **Minikube Version**: v1.35.0
- **Kubectl Version**: v1.32.4 (Client), v1.32.0 (Server)

[Screenshot: System Information showing OS, Docker, and Kubernetes versions]

2. Step-by-Step Implementation

2.1 Environment Setup

```
# Install Docker
sudo apt update
sudo apt install docker.io
sudo systemctl start docker
sudo systemctl enable docker
sudo usermod -aG docker $USER

# Install Minikube
curl -LO
https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube

# Install kubectl
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
sudo install kubectl /usr/local/bin/kubectl
```

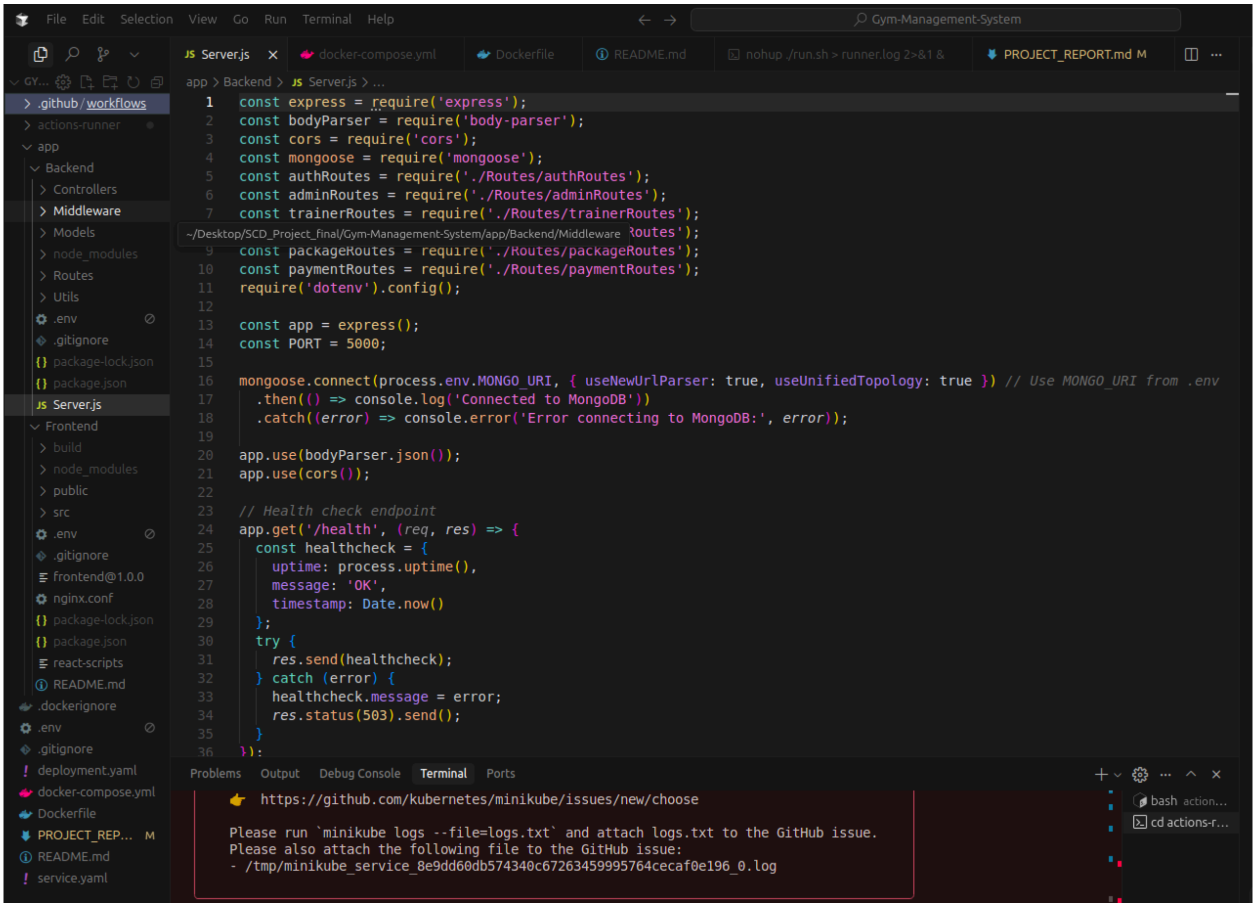


```
      --one_output                      If true, only write logs to their nativ
e severity level (vs also writing to each lower severity level; no effect when -
logtostderr=true)
      --skip_headers                    If true, avoid header prefixes in the l
og messages
      --skip_log_headers                If true, avoid headers when opening log
 files (no effect when -logtostderr=true)
      --stderrthreshold severity        logs at or above this threshold go to s
tderr when writing to files and stderr (no effect when -logtostderr=true or -als
ologtostderr=true) (default 2)
  -v, --v Level                         number for the log level verbosity
      --vmodule moduleSpec              comma-separated list of pattern=N setti
ngs for file-filtered logging
invalid argument "ersion" for "-v, --v" flag: strconv.ParseInt: parsing "ersion"
: invalid syntax
saim@saim-Latitude-5480:~$ minikube version
minikube version: v1.35.0
commit: dd5d320e41b5451cdf3c01891bc4e13d189586ed-dirty
saim@saim-Latitude-5480:~$ kubectl version --client
Client Version: v1.32.4
Kustomize Version: v5.5.0
saim@saim-Latitude-5480:~$ docker --version
Docker version 28.1.1, build 4eba377
saim@saim-Latitude-5480:~$ 
```

2.2 Application Development

- Created a full-stack Gym Management System using:
  - Frontend: React.js
  - Backend: Node.js/Express.js
  - Database: MongoDB

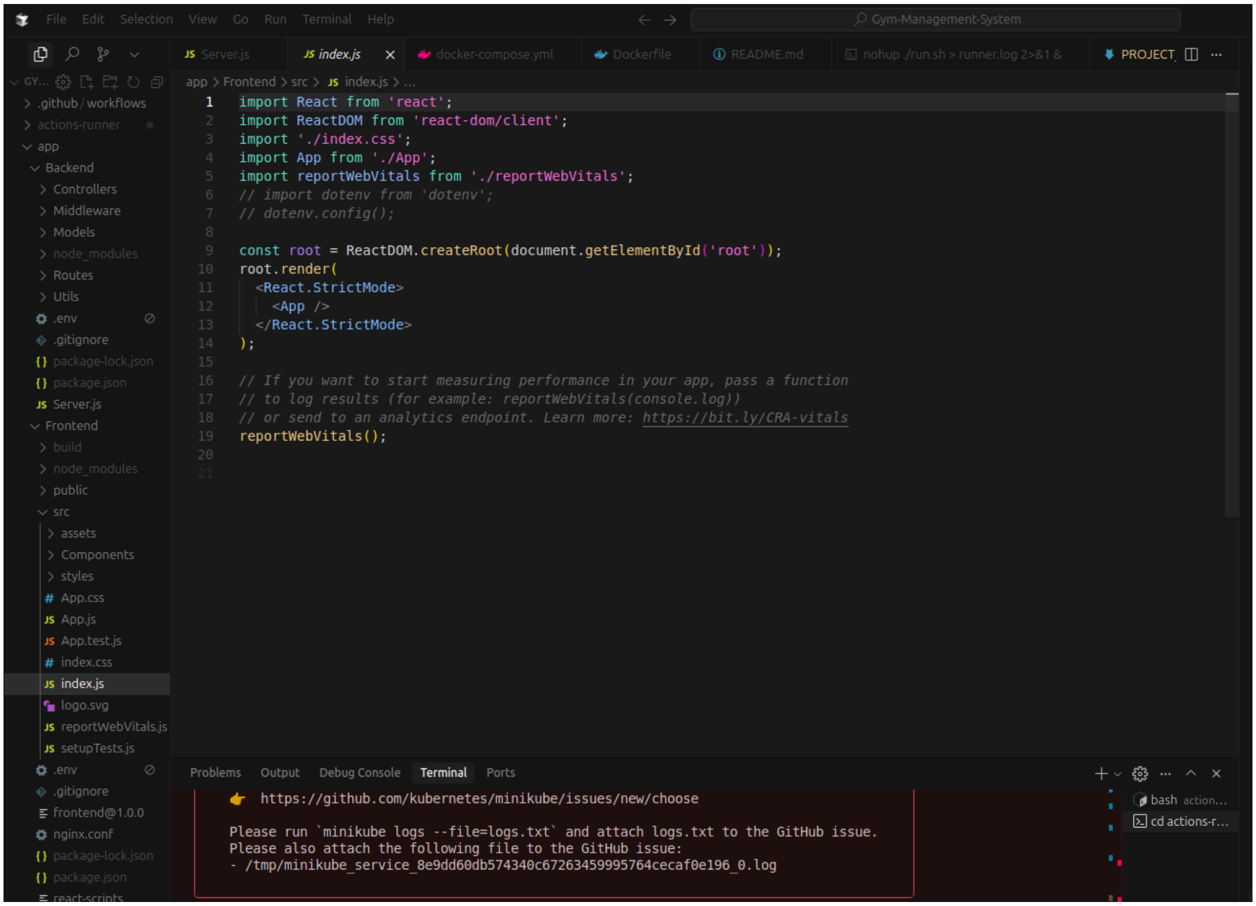
- /app/backend/server.js



```
const express = require('express');
const bodyParser = require('body-parser');
const cors = require('cors');
const mongoose = require('mongoose');
const authRoutes = require('./Routes/authRoutes');
const adminRoutes = require('./Routes/adminRoutes');
const trainerRoutes = require('./Routes/trainerRoutes');
                                          Routes');
const packageRoutes = require('./Routes/packageRoutes');
const paymentRoutes = require('./Routes/paymentRoutes');
require('dotenv').config();

const app = express();
const PORT = 5000;

mongoose.connect(process.env.MONGO_URI, { useNewUrlParser: true, useUnifiedTopology: true }) // Use MONGO_URI from .env
  .then(() => console.log('Connected to MongoDB'))
  .catch((error) => console.error('Error connecting to MongoDB:', error));

app.use(bodyParser.json());
app.use(cors());

// Health check endpoint
app.get('/health', (req, res) => {
  const healthcheck = {
    uptime: process.uptime(),
    message: 'OK',
    timestamp: Date.now()
  };
  try {
    res.send(healthcheck);
  } catch (error) {
    healthcheck.message = error;
    res.status(503).send();
  }
});
```

```
👉  https://github.com/kubernetes/minikube/issues/new/choose

Please run `minikube logs --file=logs.txt` and attach logs.txt to the GitHub issue.
Please also attach the following file to the GitHub issue:
- /tmp/minikube_service_8e9dd60db574340c67263459995764cecaf0e196_0.log
```

- /app/frontend/src/index.js



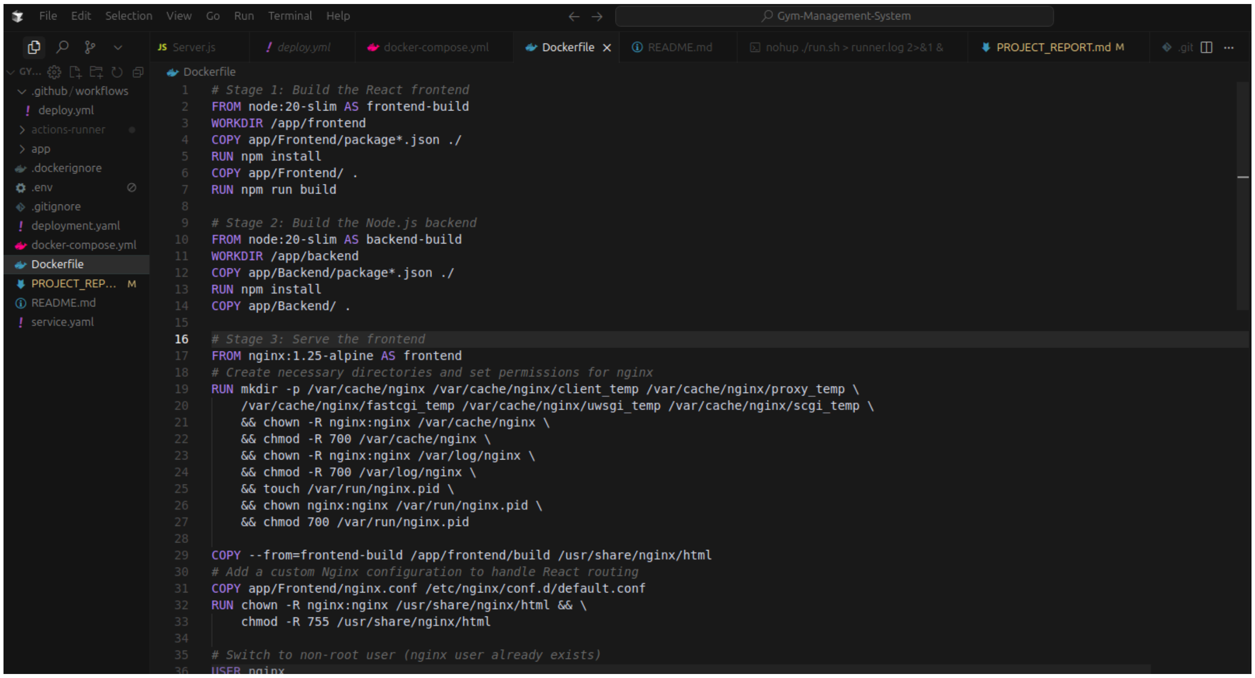
```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
// import dotenv from 'dotenv';
// dotenv.config();

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

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

```
    👉  https://github.com/kubernetes/minikube/issues/new/choose

    Please run `minikube logs --file=logs.txt` and attach logs.txt to the GitHub issue.
    Please also attach the following file to the GitHub issue:
    - /tmp/minikube_service_8e9dd60db574340c67263459995764cecaf0e196_0.log
```

2.3 Containerization

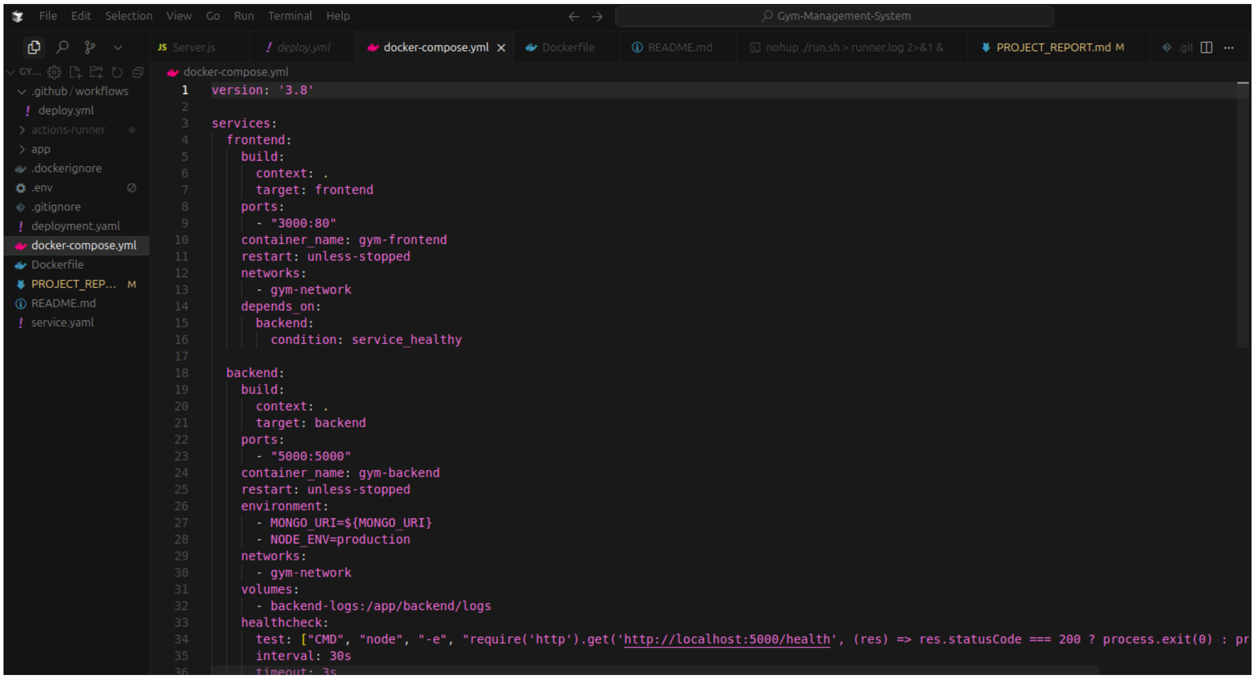
```
# Build Docker images
docker build --target frontend -t saim814/gym-frontend:latest .
docker build --target backend -t saim814/gym-backend:latest .
```

- Dockerfile



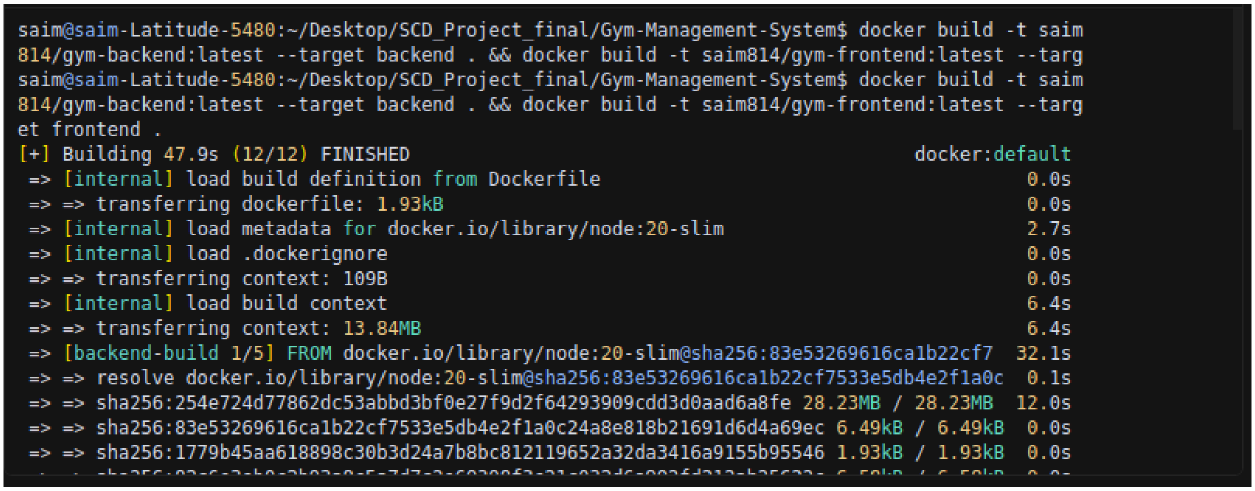
```dockerfile
# Stage 1: Build the React frontend
FROM node:20-slim AS frontend-build
WORKDIR /app/frontend
COPY app/Frontend/package*.json ./
RUN npm install
COPY app/Frontend/ .
RUN npm run build

# Stage 2: Build the Node.js backend
FROM node:20-slim AS backend-build
WORKDIR /app/backend
COPY app/Backend/package*.json ./
RUN npm install
COPY app/Backend/ .

# Stage 3: Serve the frontend
FROM nginx:1.25-alpine AS frontend
# Create necessary directories and set permissions for nginx
RUN mkdir -p /var/cache/nginx /var/cache/nginx/client_temp /var/cache/nginx/proxy_temp \
    /var/cache/nginx/fastcgi_temp /var/cache/nginx/uwsgi_temp /var/cache/nginx/scgi_temp \
    && chown -R nginx:nginx /var/cache/nginx \
    && chmod -R 700 /var/cache/nginx \
    && chown -R nginx:nginx /var/log/nginx \
    && chmod -R 700 /var/log/nginx \
    && touch /var/run/nginx.pid \
    && chown nginx:nginx /var/run/nginx.pid \
    && chmod 700 /var/run/nginx.pid

COPY --from=frontend-build /app/frontend/build /usr/share/nginx/html
# Add a custom Nginx configuration to handle React routing
COPY app/Frontend/nginx.conf /etc/nginx/conf.d/default.conf
RUN chown -R nginx:nginx /usr/share/nginx/html && \
    chmod -R 755 /usr/share/nginx/html

# Switch to non-root user (nginx user already exists)
USER nginx
```

- Docker Compose file



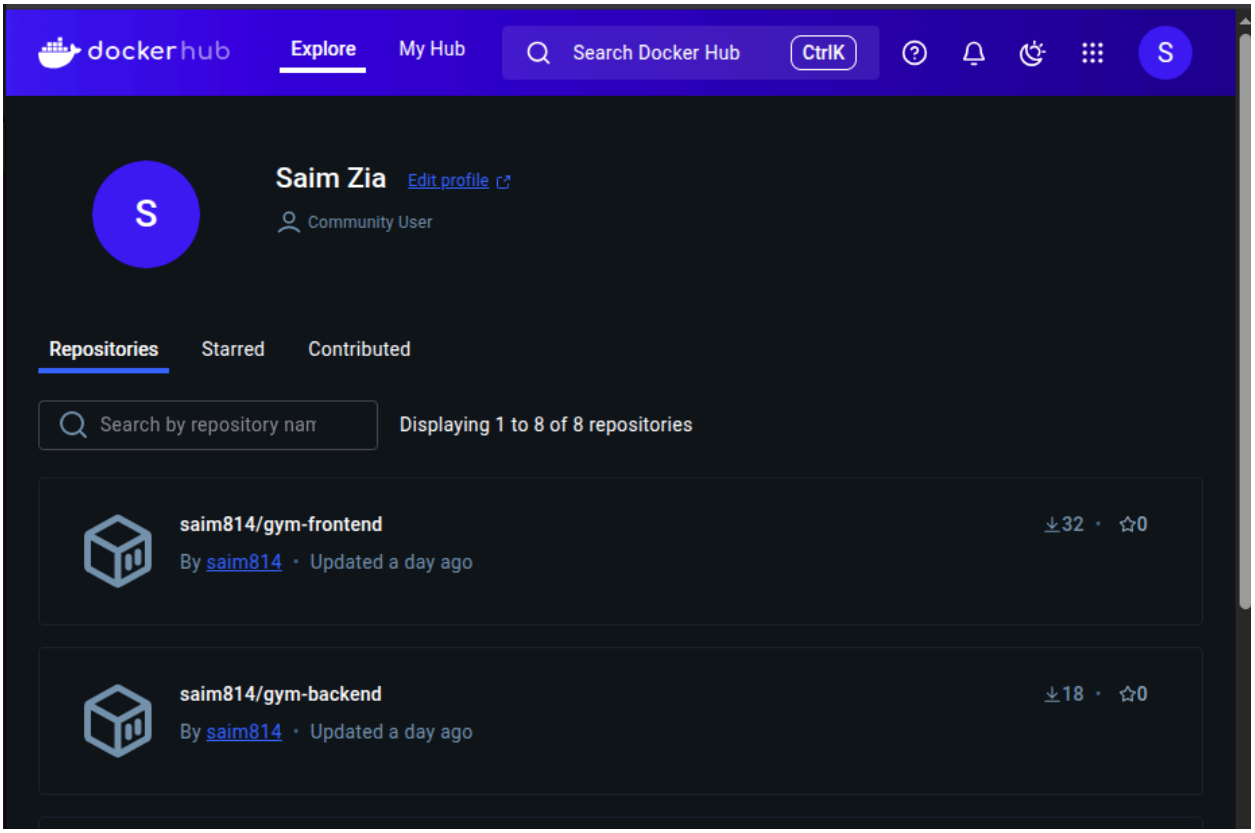
```yaml
version: '3.8'

services:
  frontend:
    build:
      context: .
      target: frontend
    ports:
      - "3000:80"
    container_name: gym-frontend
    restart: unless-stopped
    networks:
      - gym-network
    depends_on:
      backend:
        condition: service_healthy

  backend:
    build:
      context: .
      target: backend
    ports:
      - "5000:5000"
    container_name: gym-backend
    restart: unless-stopped
    environment:
      - MONGO_URI=${MONGO_URI}
      - NODE_ENV=production
    networks:
      - gym-network
    volumes:
      - backend-logs:/app/backend/logs
    healthcheck:
      test: ["CMD", "node", "-e", "require('http').get('http://localhost:5000/health', (res) => res.statusCode === 200 ? process.exit(0) : pr
      interval: 30s
      timeout: 3s
```

- Docker images being built



```
saim@saim-Latitude-5480:~/Desktop/SCD_Project_final/Gym-Management-System$ docker build -t saim
814/gym-backend:latest --target backend . && docker build -t saim814/gym-frontend:latest --targ
saim@saim-Latitude-5480:~/Desktop/SCD_Project_final/Gym-Management-System$ docker build -t saim
814/gym-backend:latest --target backend . && docker build -t saim814/gym-frontend:latest --targ
et frontend .
[+] Building 47.9s (12/12) FINISHED                                              docker:default
 => [internal] load build definition from Dockerfile                                       0.0s
 => => transferring dockerfile: 1.93kB                                                     0.0s
 => [internal] load metadata for docker.io/library/node:20-slim                            2.7s
 => [internal] load .dockerignore                                                          0.0s
 => => transferring context: 109B                                                          0.0s
 => [internal] load build context                                                          6.4s
 => => transferring context: 13.84MB                                                       6.4s
 => [backend-build 1/5] FROM docker.io/library/node:20-slim@sha256:83e53269616ca1b22cf7  32.1s
 => => resolve docker.io/library/node:20-slim@sha256:83e53269616ca1b22cf7533e5db4e2f1a0c  0.1s
 => => sha256:254e724d77862dc53abbd3bf0e27f9d2f64293909cdd3d0aad6a8fe 28.23MB / 28.23MB  12.0s
 => => sha256:83e53269616ca1b22cf7533e5db4e2f1a0c24a8e818b21691d6d4a69ec 6.49kB / 6.49kB  0.0s
 => => sha256:1779b45aa618898c30b3d24a7b8bc812119652a32da3416a9155b95546 1.93kB / 1.93kB  0.0s
```

- Docker images list showing built images



Saim Zia

Community User

Displaying 1 to 8 of 8 repositories

saim814/gym-frontend
By saim814 · Updated a day ago
32 · 0

saim814/gym-backend
By saim814 · Updated a day ago
18 · 0

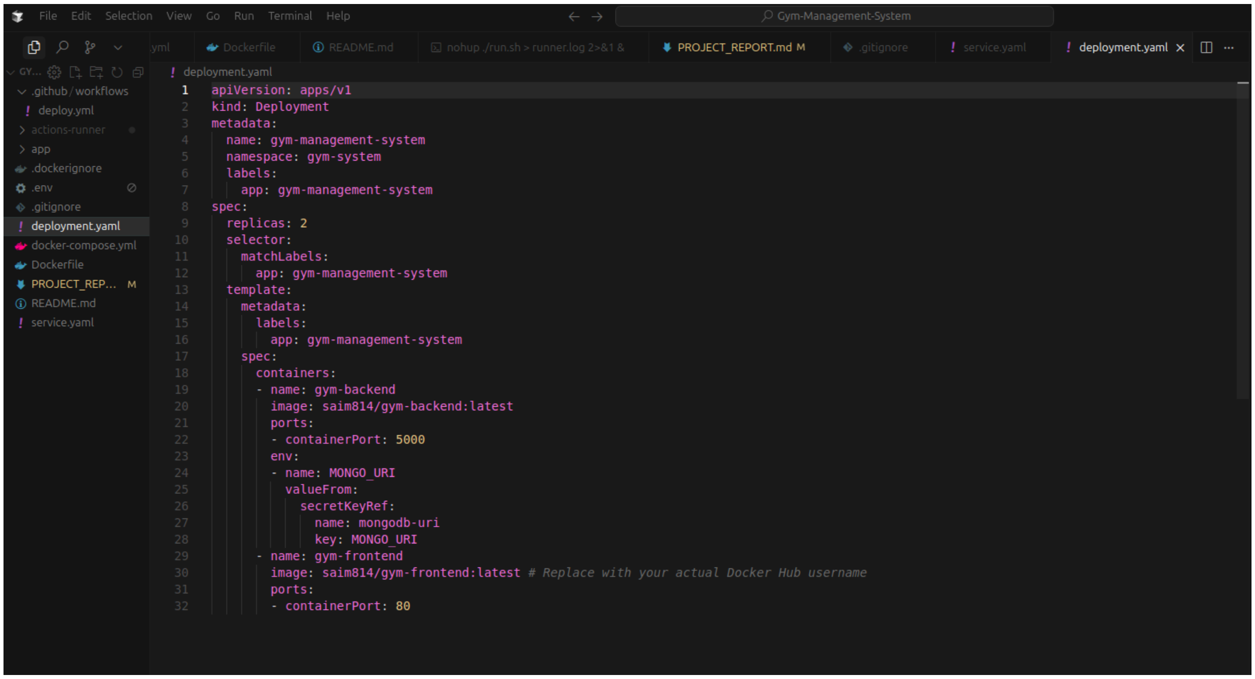
2.4 Kubernetes Deployment

Current Cluster Status:

```
# Node Information
kubectl get nodes -o wide
[Output: Node details]
```

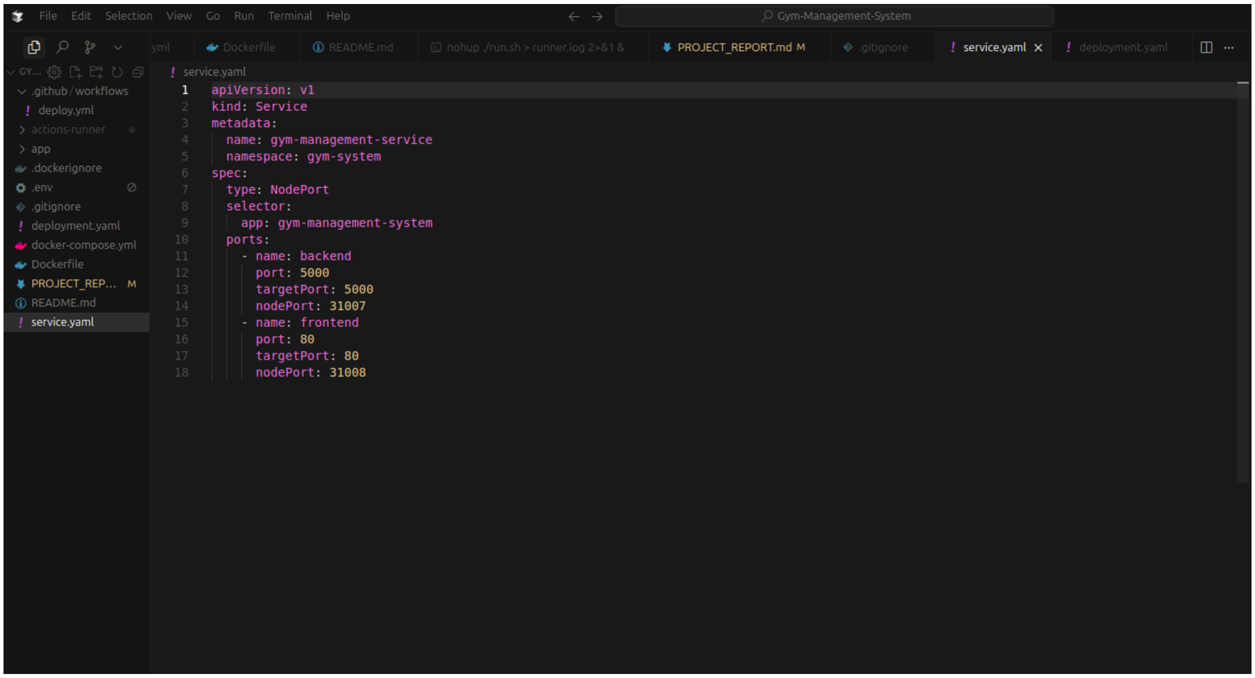
```
# Pod Information
kubectl get pods -o wide -n gym-system
[Output: Pod details]

# Service Information
kubectl get services -o wide -n gym-system
[Output: Service details]
```

- Deployment.yaml



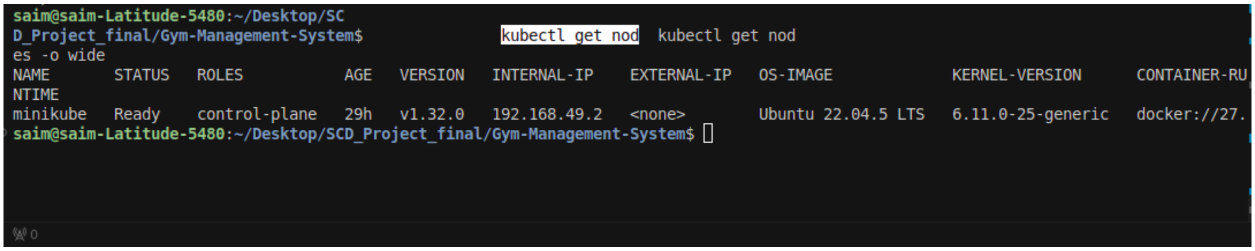
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gym-management-system
  namespace: gym-system
  labels:
    app: gym-management-system
spec:
  replicas: 2
  selector:
    matchLabels:
      app: gym-management-system
  template:
    metadata:
      labels:
        app: gym-management-system
    spec:
      containers:
      - name: gym-backend
        image: saim814/gym-backend:latest
        ports:
        - containerPort: 5000
        env:
        - name: MONGO_URI
          valueFrom:
            secretKeyRef:
              name: mongodb-uri
              key: MONGO_URI
      - name: gym-frontend
        image: saim814/gym-frontend:latest # Replace with your actual Docker Hub username
        ports:
        - containerPort: 80
```

- service.yaml



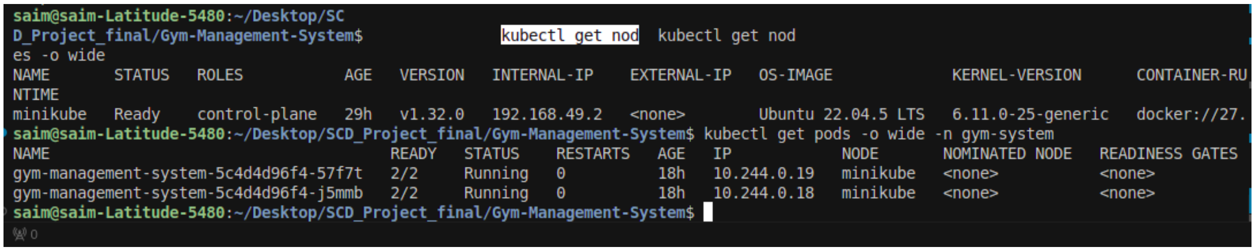
```yaml
apiVersion: v1
kind: Service
metadata:
  name: gym-management-service
  namespace: gym-system
spec:
  type: NodePort
  selector:
    app: gym-management-system
  ports:
    - name: backend
      port: 5000
      targetPort: 5000
      nodePort: 31007
    - name: frontend
      port: 80
      targetPort: 80
      nodePort: 31008
```

[Screenshot: Kubernetes nodes status]



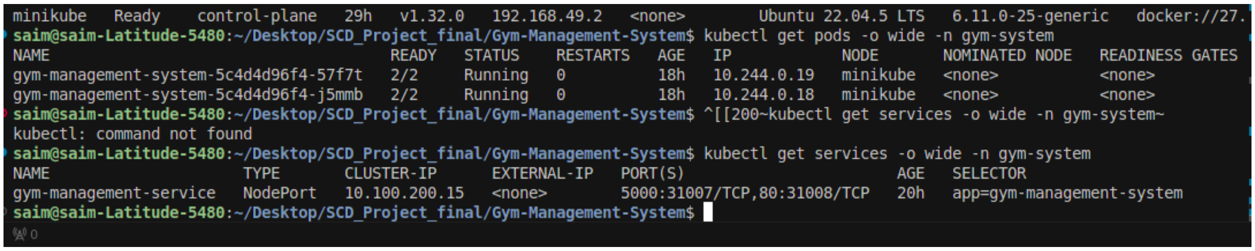
```
saim@saim-Latitude-5480:~/Desktop/SC
D_Project_final/Gym-Management-System$                kubectl get nod  kubectl get nod
es -o wide
NAME       STATUS   ROLES           AGE   VERSION   INTERNAL-IP    EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION     CONTAINER-RU
NTIME
minikube   Ready    control-plane   29h   v1.32.0   192.168.49.2   <none>        Ubuntu 22.04.5 LTS   6.11.0-25-generic   docker://27.
saim@saim-Latitude-5480:~/Desktop/SCD_Project_final/Gym-Management-System$
```

[Screenshot: Running pods in the cluster]



```
saim@saim-Latitude-5480:~/Desktop/SC
D_Project_final/Gym-Management-System$                kubectl get nod  kubectl get nod
es -o wide
NAME       STATUS   ROLES           AGE   VERSION   INTERNAL-IP    EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION     CONTAINER-RU
NTIME
minikube   Ready    control-plane   29h   v1.32.0   192.168.49.2   <none>        Ubuntu 22.04.5 LTS   6.11.0-25-generic   docker://27.
saim@saim-Latitude-5480:~/Desktop/SCD_Project_final/Gym-Management-System$ kubectl get pods -o wide -n gym-system
NAME                                     READY   STATUS    RESTARTS   AGE   IP            NODE       NOMINATED NODE   READINESS GATES
gym-management-system-5c4d4d96f4-57f7t   2/2     Running   0          18h   10.244.0.19   minikube   <none>           <none>
gym-management-system-5c4d4d96f4-j5mmb   2/2     Running   0          18h   10.244.0.18   minikube   <none>           <none>
saim@saim-Latitude-5480:~/Desktop/SCD_Project_final/Gym-Management-System$
```

[Screenshot: Services configuration]

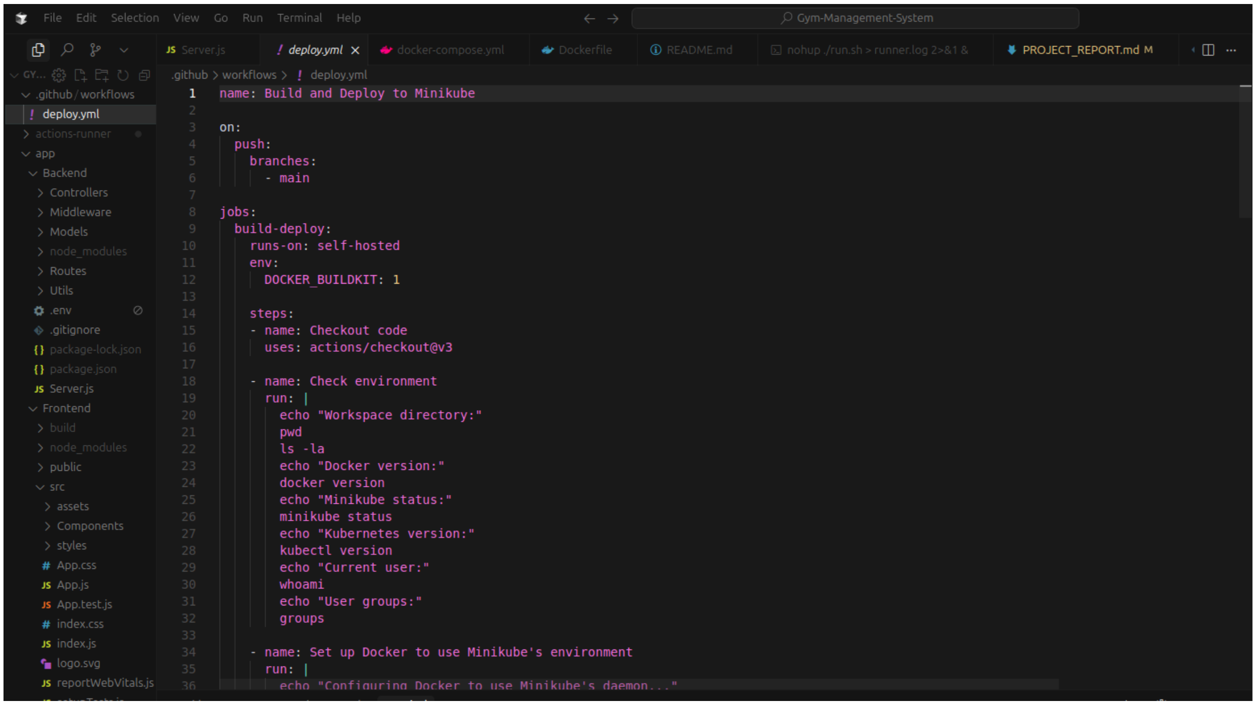


```
minikube   Ready    control-plane   29h   v1.32.0   192.168.49.2   <none>        Ubuntu 22.04.5 LTS   6.11.0-25-generic   docker://27.
saim@saim-Latitude-5480:~/Desktop/SCD_Project_final/Gym-Management-System$ kubectl get pods -o wide -n gym-system
NAME                                     READY   STATUS    RESTARTS   AGE   IP            NODE       NOMINATED NODE   READINESS GATES
gym-management-system-5c4d4d96f4-57f7t   2/2     Running   0          18h   10.244.0.19   minikube   <none>           <none>
gym-management-system-5c4d4d96f4-j5mmb   2/2     Running   0          18h   10.244.0.18   minikube   <none>           <none>
saim@saim-Latitude-5480:~/Desktop/SCD_Project_final/Gym-Management-System$ ^[[200~kubectl get services -o wide -n gym-system~
kubectl: command not found
saim@saim-Latitude-5480:~/Desktop/SCD_Project_final/Gym-Management-System$ kubectl get services -o wide -n gym-system
NAME                     TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)                       AGE   SELECTOR
gym-management-service   NodePort   10.100.200.15   <none>        5000:31007/TCP,80:31008/TCP   20h   app=gym-management-system
saim@saim-Latitude-5480:~/Desktop/SCD_Project_final/Gym-Management-System$
```

2.5 CI/CD Setup with GitHub Actions

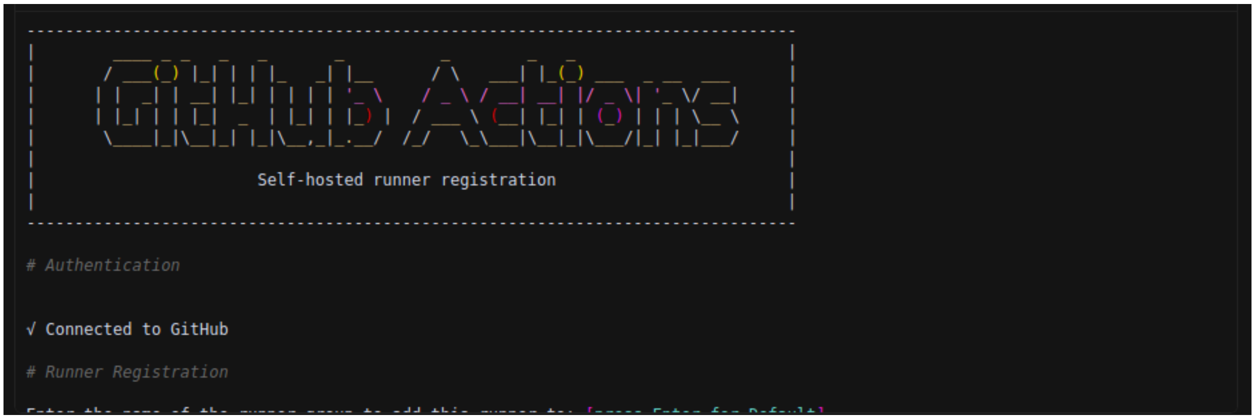
- Created `.github/workflows/deploy.yml`
- Configured self-hosted runner
- Set up Docker Hub authentication

[Screenshot: GitHub Actions workflow configuration]

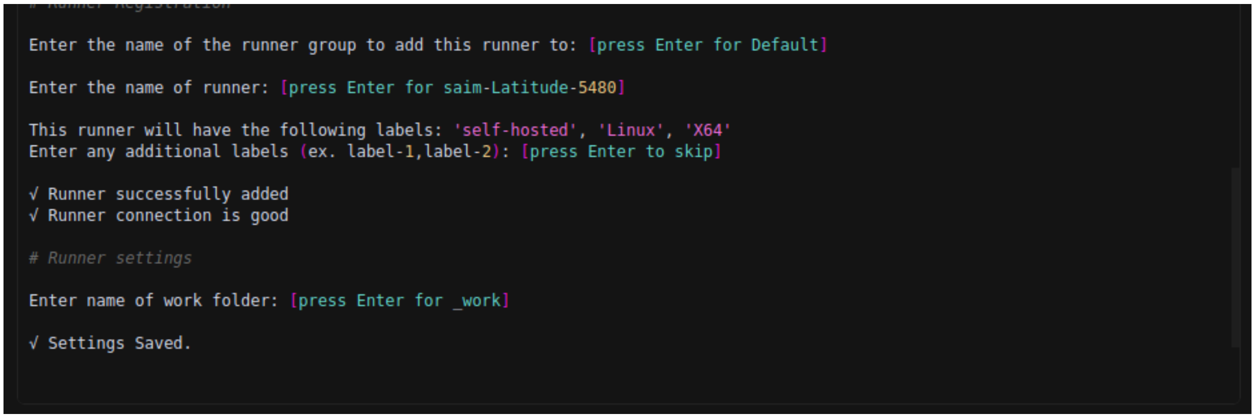


```yaml
name: Build and Deploy to Minikube

on:
  push:
    branches:
      - main

jobs:
  build-deploy:
    runs-on: self-hosted
    env:
      DOCKER_BUILDKIT: 1

    steps:
    - name: Checkout code
      uses: actions/checkout@v3

    - name: Check environment
      run: |
        echo "Workspace directory:"
        pwd
        ls -la
        echo "Docker version:"
        docker version
        echo "Minikube status:"
        minikube status
        echo "Kubernetes version:"
        kubectl version
        echo "Current user:"
        whoami
        echo "User groups:"
        groups

    - name: Set up Docker to use Minikube's environment
      run: |
        echo "Configuring Docker to use Minikube's daemon..."
```

[Screenshot: GitHub Actions runner setup]

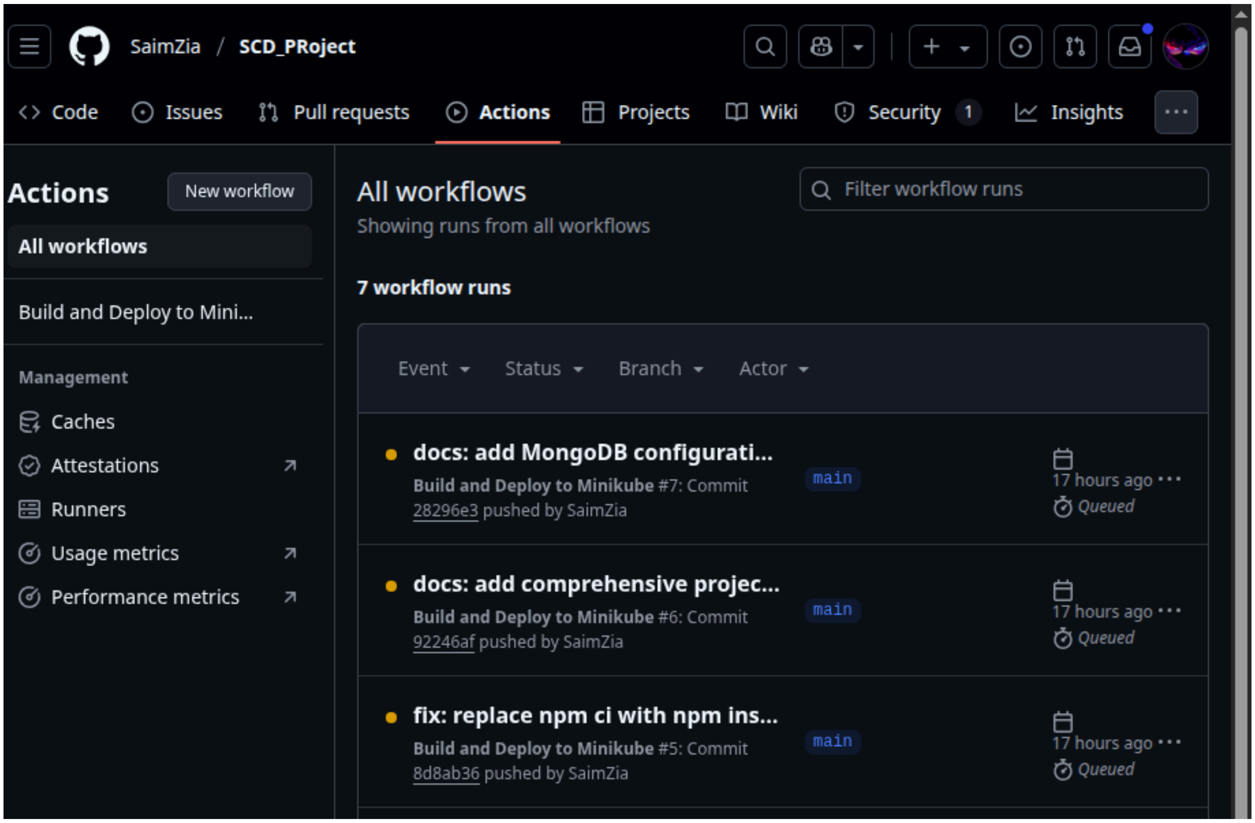


```
--------------------------------------------------------------------------------
|        ____ _ _   _   _       _          _        _   _                      |
|       / ___(_) |_| | | |_   _| |__      / \   ___| |_(_) ___  _ __  ___      |
|      | |  _| | __| |_| | | | | '_ \    / _ \ / __| __| |/ _ \| '_ \/ __|     |
|      | |_| | | |_|  _  | |_| | |_) |  / ___ \ (__| |_| | (_) | | | \__ \     |
|       \____|_|\__|_| |_|\__,_|_.__/  /_/   \_\___|\__|_|\___/|_| |_|___/     |
|                                                                              |
|                       Self-hosted runner registration                        |
|                                                                              |
--------------------------------------------------------------------------------

# Authentication


√ Connected to GitHub

# Runner Registration

Enter the name of the runner group to add this runner to: [press Enter for Default]
```



```
# Runner Registration

Enter the name of the runner group to add this runner to: [press Enter for Default]

Enter the name of runner: [press Enter for saim-Latitude-5480]

This runner will have the following labels: 'self-hosted', 'Linux', 'X64'
Enter any additional labels (ex. label-1,label-2): [press Enter to skip]

√ Runner successfully added
√ Runner connection is good

# Runner settings

Enter name of work folder: [press Enter for _work]

√ Settings Saved.
```

[Screenshot: Successful workflow run]

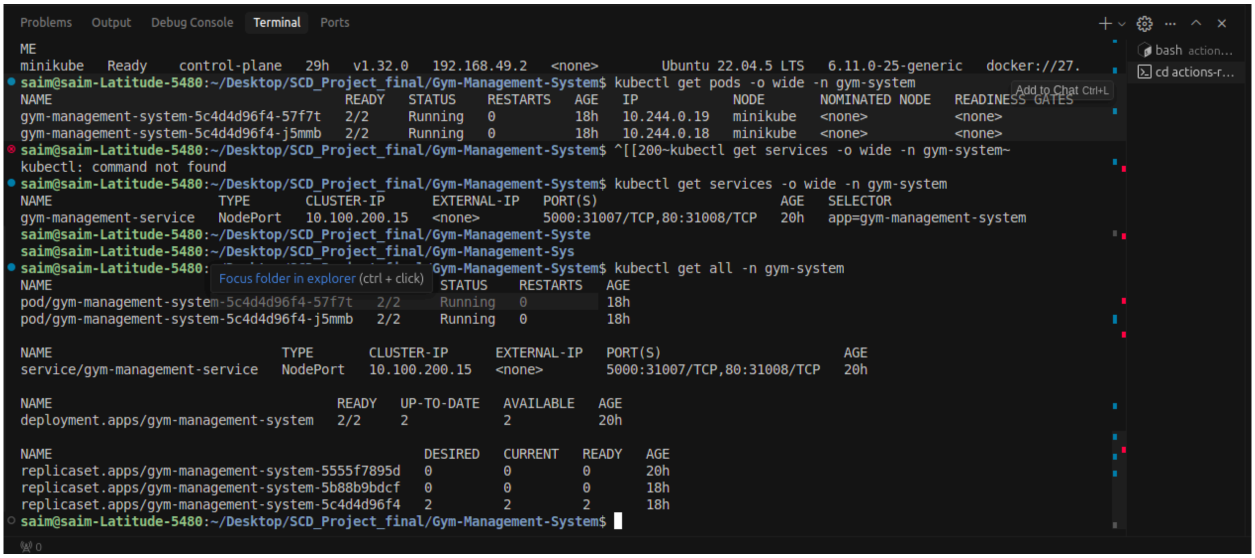


All workflows
Showing runs from all workflows

7 workflow runs

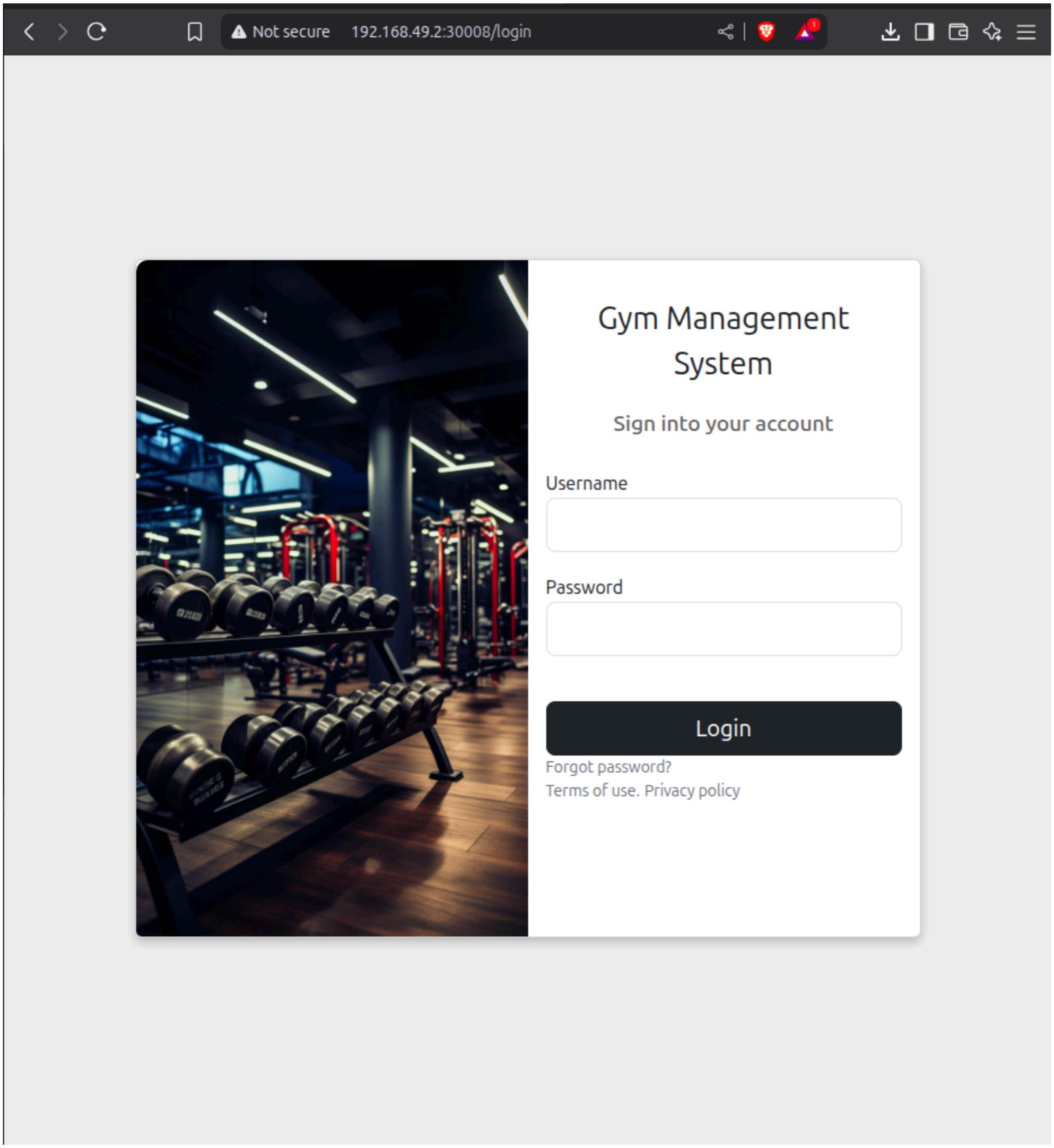
docs: add MongoDB configurati... **Build and Deploy to Minikube** #7: Commit 28296e3 pushed by SaimZia	main	17 hours ago Queued
docs: add comprehensive projec... **Build and Deploy to Minikube** #6: Commit 92246af pushed by SaimZia	main	17 hours ago Queued
fix: replace npm ci with npm ins... **Build and Deploy to Minikube** #5: Commit 8d8ab36 pushed by SaimZia	main	17 hours ago Queued

-
  ```
  # Check all resources
  kubectl get all -n gym-system
  ```



```
ME
minikube   Ready    control-plane   29h   v1.32.0   192.168.49.2   <none>        Ubuntu 22.04.5 LTS   6.11.0-25-generic   docker://27.
saim@saim-Latitude-5480:~/Desktop/SCD_Project_final/Gym-Management-System$ kubectl get pods -o wide -n gym-system
NAME                                       READY   STATUS    RESTARTS   AGE   IP            NODE       NOMINATED NODE   READINESS GATES
gym-management-system-5c4d4d96f4-57f7t   2/2     Running   0          18h   10.244.0.19   minikube   <none>           <none>
gym-management-system-5c4d4d96f4-j5mmb   2/2     Running   0          18h   10.244.0.18   minikube   <none>           <none>
saim@saim-Latitude-5480:~/Desktop/SCD_Project_final/Gym-Management-System$ ^[[200~kubectl get services -o wide -n gym-system~
kubectl: command not found
saim@saim-Latitude-5480:~/Desktop/SCD_Project_final/Gym-Management-System$ kubectl get services -o wide -n gym-system
NAME                     TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)                        AGE   SELECTOR
gym-management-service   NodePort   10.100.200.15   <none>        5000:31007/TCP,80:31008/TCP   20h   app=gym-management-system
saim@saim-Latitude-5480:~/Desktop/SCD_Project_final/Gym-Management-Syste
saim@saim-Latitude-5480:~/Desktop/SCD_Project_final/Gym-Management-Sys
saim@saim-Latitude-5480:                              Gym-Management-System$ kubectl get all -n gym-system
NAME                                                             STATUS    RESTARTS   AGE
pod/gym-management-system-5c4d4d96f4-57f7t   2/2     Running   0          18h
pod/gym-management-system-5c4d4d96f4-j5mmb   2/2     Running   0          18h

NAME                             TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)                        AGE
service/gym-management-service   NodePort   10.100.200.15   <none>        5000:31007/TCP,80:31008/TCP   20h

NAME                                    READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/gym-management-system   2/2     2            2           20h

NAME                                               DESIRED   CURRENT   READY   AGE
replicaset.apps/gym-management-system-5555f7895d   0         0         0       20h
replicaset.apps/gym-management-system-5b88b9bdcf   0         0         0       18h
replicaset.apps/gym-management-system-5c4d4d96f4   2         2         2       18h
saim@saim-Latitude-5480:~/Desktop/SCD_Project_final/Gym-Management-System$ 
```

Running Website:



Not secure 192.168.49.2:30008/login

Gym Management System

Sign into your account

Username

Password

Login

Forgot password?

Terms of use. Privacy policy

3. Issues Faced and Solutions

Issue 1: Docker Permission Issues

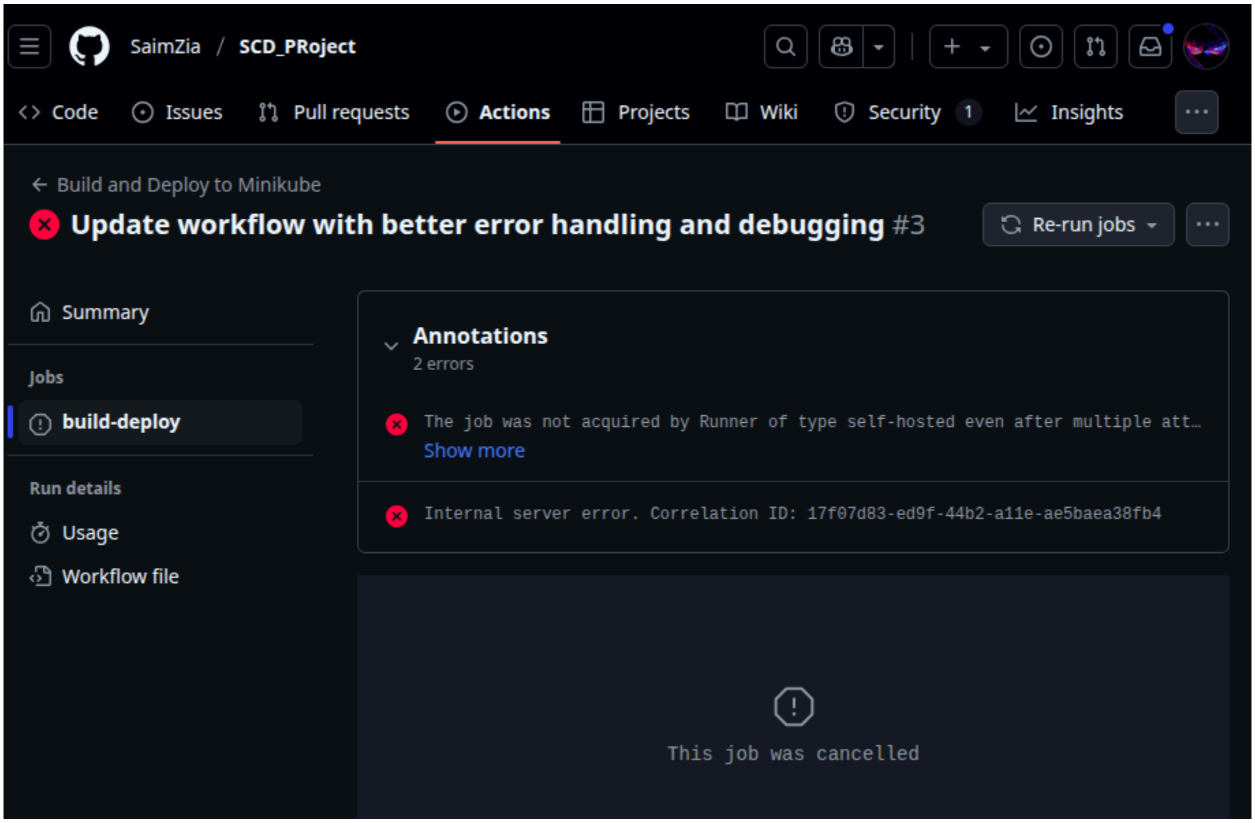
Problem: Permission denied while trying to connect to Docker daemon **Solution**: Added user to docker group

Issue 2: Minikube Node Port Access

Problem: Unable to access application through NodePort **Solution**: Used minikube service command

Issue 3: GitHub Actions Runner Connection

Problem: Runner not picking up jobs **Solution**: Reconfigured runner with new token [Screenshot: Runner connection issue]



Update workflow with better error handling and debugging #3

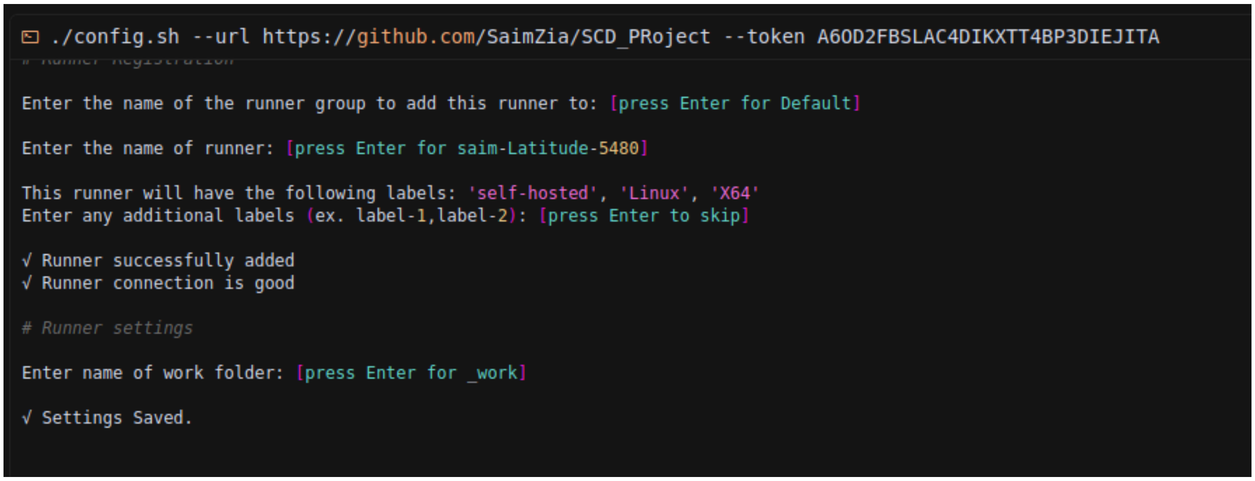
Annotations
2 errors

```
The job was not acquired by Runner of type self-hosted even after multiple att…
```

```
Internal server error. Correlation ID: 17f07d83-ed9f-44b2-a11e-ae5baea38fb4
```

```
This job was cancelled
```

[Screenshot: Successful runner connection]

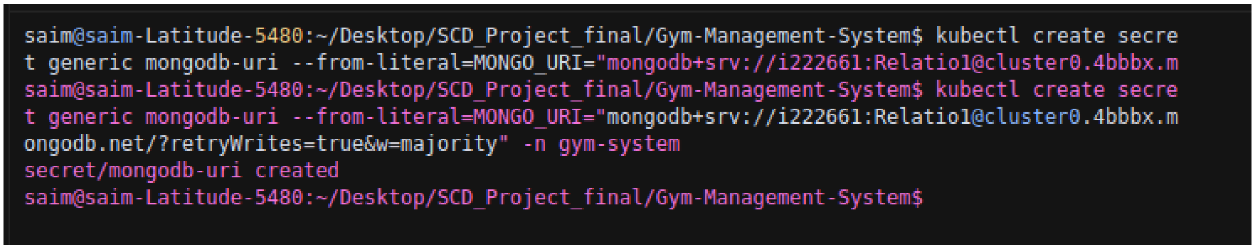


```
./config.sh --url https://github.com/SaimZia/SCD_PRoject --token A6OD2FBSLAC4DIKXTT4BP3DIEJITA
# Runner Registration

Enter the name of the runner group to add this runner to: [press Enter for Default]

Enter the name of runner: [press Enter for saim-Latitude-5480]

This runner will have the following labels: 'self-hosted', 'Linux', 'X64'
Enter any additional labels (ex. label-1,label-2): [press Enter to skip]

√ Runner successfully added
√ Runner connection is good

# Runner settings

Enter name of work folder: [press Enter for _work]

√ Settings Saved.
```

Issue 4: MongoDB Connection Issues

Problem: Backend unable to connect to MongoDB **Solution**: Created proper Kubernetes secret

[Screenshot: Working database connection]



```
saim@saim-Latitude-5480:~/Desktop/SCD_Project_final/Gym-Management-System$ kubectl create secre
t generic mongodb-uri --from-literal=MONGO_URI="mongodb+srv://i222661:Relatio1@cluster0.4bbbx.m
saim@saim-Latitude-5480:~/Desktop/SCD_Project_final/Gym-Management-System$ kubectl create secre
t generic mongodb-uri --from-literal=MONGO_URI="mongodb+srv://i222661:Relatio1@cluster0.4bbbx.m
ongodb.net/?retryWrites=true&w=majority" -n gym-system
secret/mongodb-uri created
saim@saim-Latitude-5480:~/Desktop/SCD_Project_final/Gym-Management-System$
```

Issue 5: Docker Build Context Issues

Problem: Slow builds due to large context **Solution**: Implemented proper .dockerignore

4. Project Running Instructions

4.1 Starting from Scratch

1. Clone the repository:

```
git clone https://github.com/SaimZia/SCD_PRoject.git
cd SCD_PRoject
```

2. Start Minikube cluster:

```
# Start Minikube
minikube start

# Verify cluster is running
minikube status
kubectl cluster-info
```

3. Configure Docker to use Minikube's Docker daemon:

```
eval $(minikube docker-env)
```

4. Set up MongoDB:

```
# Create MongoDB secret (replace with your MongoDB URI)
kubectl create namespace gym-system
kubectl create secret generic mongodb-uri \

--from-literal=MONGO_URI="mongodb+srv://your-username:your-password@your-cluster-url" \
  -n gym-system
```

5. Build Docker images:

```
# Build frontend image
docker build --target frontend -t saim814/gym-frontend:latest .

# Build backend image
docker build --target backend -t saim814/gym-backend:latest .

# Verify images are built
docker images | grep saim814
```

6. Deploy to Kubernetes:

```
# Apply Kubernetes configurations
kubectl apply -f deployment.yaml -n gym-system
kubectl apply -f service.yaml -n gym-system

# Verify deployments and services
kubectl get deployments -n gym-system
kubectl get services -n gym-system
kubectl get pods -n gym-system
```

7. Set up GitHub Actions Runner (for CI/CD):

```
# Create and navigate to actions-runner directory
mkdir actions-runner && cd actions-runner

# Download runner package
curl -o actions-runner-linux-x64-2.323.0.tar.gz -L
https://github.com/actions/runner/releases/download/v2.323.0/actions-ru
nner-linux-x64-2.323.0.tar.gz

# Extract runner
tar xzf ./actions-runner-linux-x64-2.323.0.tar.gz

# Configure runner (replace TOKEN with your GitHub runner token)
./config.sh --url https://github.com/SaimZia/SCD_PRoject --token
YOUR_TOKEN

# Start runner
./run.sh
```

4.2 Local Deployment with Minikube

1. Get service URLs:

```
# Get URLs for both frontend and backend services
minikube service gym-management-service --url -n gym-system
```

2. Access the application:

   – Frontend UI: Access through the NodePort URL (port 30008)
   – Backend API: Access through the NodePort URL (port 30007)

4.3 Accessing the Application

1. Get service URLs:

```
# Get URLs for both frontend and backend services
minikube service gym-management-service --url -n gym-system
```

2. Access the application:

   - Frontend UI: Access through the NodePort URL (port 30008)
   - Backend API: Access through the NodePort URL (port 30007)

4.4 Monitoring and Debugging

1. Check pod status:

```
# View pod status and logs
kubectl get pods -n gym-system
kubectl describe pods -n gym-system
kubectl logs -n gym-system <pod-name> -c gym-frontend
kubectl logs -n gym-system <pod-name> -c gym-backend
```

2. Check service status:

```
kubectl get services -n gym-system
kubectl describe service gym-management-service -n gym-system
```

3. Monitor resources:

```
# Monitor CPU and memory usage
kubectl top pods -n gym-system
kubectl top nodes
```

Conclusion

The project successfully implements a containerized full-stack application with automated deployment using GitHub Actions and Kubernetes. The use of Linux as the development environment proved beneficial for native container support and better resource utilization. The combination of React.js, Node.js, and MongoDB provides a scalable and maintainable solution for gym management.

Running Website Locally with Minikube

1. Start Minikube tunnel (in a separate terminal):

```
# Start minikube tunnel to enable LoadBalancer services
sudo minikube tunnel
```

2. Get the Minikube IP:

```
# Get Minikube IP address
minikube ip
```

3. Access the website:

```
# Get the NodePort URLs
minikube service gym-management-service --url -n gym-system

# Or use these commands to automatically open in browser
minikube service gym-management-service -n gym-system
```

4. Quick start commands (all-in-one):

```
# Start everything from scratch
minikube start
eval $(minikube docker-env)
kubectl create namespace gym-system
kubectl apply -f deployment.yaml -n gym-system
kubectl apply -f service.yaml -n gym-system

# Wait for pods to be ready
kubectl wait --for=condition=ready pod -l app=gym-management-system -n
gym-system --timeout=180s

# Open the website
minikube service gym-management-service -n gym-system
```

5. Development workflow commands:

```
# View the website without opening browser
minikube service list -n gym-system

# Get specific URLs
echo "Frontend URL: http://$(minikube ip):30008"
echo "Backend API URL: http://$(minikube ip):30007"

# Monitor the application
kubectl get pods -n gym-system -w

# View logs in real-time
kubectl logs -f -l app=gym-management-system -n gym-system
```

6. Useful debugging commands:

```
# Check if services are running
kubectl get all -n gym-system

# Check pod logs
kubectl logs -f deployment/gym-management-system -n gym-system -c
gym-frontend
kubectl logs -f deployment/gym-management-system -n gym-system -c
gym-backend

# Check pod details
kubectl describe pod -l app=gym-management-system -n gym-system
```

```
# Check service endpoints
kubectl get endpoints -n gym-system
```

7. Stop the application:

```
# Stop the services
kubectl delete -f service.yaml -n gym-system
kubectl delete -f deployment.yaml -n gym-system

# Stop Minikube (optional)
minikube stop
```

Common Local Development Tasks

1. Rebuild and redeploy after code changes:

```
# Rebuild Docker images
eval $(minikube docker-env)
docker build --target frontend -t saim814/gym-frontend:latest .
docker build --target backend -t saim814/gym-backend:latest .

# Restart the deployment
kubectl rollout restart deployment gym-management-system -n gym-system
```

2. View application logs:

```
# Frontend logs
kubectl logs -f -l app=gym-management-system -c gym-frontend -n
gym-system

# Backend logs
kubectl logs -f -l app=gym-management-system -c gym-backend -n
gym-system
```

3. Access the application directly:

```
# Get NodePort URLs
MINIKUBE_IP=$(minikube ip)
echo "Frontend: http://$MINIKUBE_IP:30008"
echo "Backend API: http://$MINIKUBE_IP:30007"
```

4. Quick health check:

```
# Check all resources
kubectl get all -n gym-system

# Check pod health
kubectl describe pods -n gym-system | grep -A 5 "Events:"
```

```
# Check service endpoints
kubectl get endpoints -n gym-system
```