This project sounds like an excellent opportunity to showcase your skills in system design, optimization, and software engineering!

Project Setup

GitHub Repository:

Create a well-organized repository with separate folders for source code, documentation, and any resources like diagrams or datasets. Include a `README.md` to explain the project's purpose, features, and setup instructions.

System Design

Simulate Customer Arrival:

Use a randomized function to generate customers with attributes like service time and priority level (e.g., VIP, Corporate, Normal). Store this data in a queue or list for processing.

Agent Scheduling:

Implement scheduling algorithms:

- Round Robin Scheduling, rotate tasks among agents equally.
- Priority Scheduling, serve VIP and Corporate customers first.
- Shortest Job Next, handle tasks with shorter service times first.

Define each agent's workload limit and availability status, keep track of tasks assigned to each agent.

Real-Time Monitoring:

- Build a dashboard or logging system to update agent availability and workload status every 5 seconds.
- Provide insights for managers to optimize staffing.

Performance Metrics

Calculate and display:

Average Customer Wait Time, time spent waiting before service begins.

Agent Utilization Rate, percentage of time agents are actively working.

Fairness in Task Distribution, measure the task balance among agents.

Containerization with Docker

Write a `Docker file` to containerize your application. Example sections:

FROM python:3.9-slim

WORKDIR /app

COPY . .

RUN pip install -r requirements.txt

CMD ["python", "main.py"]

Test your application within a containerized environment to ensure consistency across deployments.

CI/CD Pipeline

GitHub Actions Workflow

- Automate build and test processes for each commit.
- Push Docker images to Docker Hub.
- Deploy the application to a cloud platform (e.g., AWS).

Create a `.github/workflows` directory and add workflow YAML files. Here's a basic snippet:

```yaml

**name: CI/CD Pipeline**

**on: [push]**

**jobs:**

**build:**

**runs-on: ubuntu-latest**

**steps:**

**- name: Checkout code**

**uses: actions/checkout@v2**

**- name: Set up Python**

**uses: actions/setup-python@v3**

**with:**
```

```
        python-version: '3.9'
    - name: Install dependencies
      run: |
        pip install -r requirements.txt
    - name: Run tests
      run: pytest
  deploy:
    runs-on: ubuntu-latest
```

steps

step 1:

Build Docker image

Copy and run:

docker build -t resource-scheduler:latest .

step 2:

Push Docker image to Docker Hub

env:

DOCKER_USERNAME: ${{ secrets.DOCKER_USERNAME }}

DOCKER_PASSWORD: ${{ secrets.DOCKER_PASSWORD }}

Copy and run:

echo "$DOCKER_PASSWORD" | docker login -u "$DOCKER_USERNAME" --password-stdin

docker push resource-scheduler:latest

Documentation & Presentation

Keep an updated Software Requirements Specification (SRS) document that outlines:

- Functional and non-functional requirements.
- System design and architecture.

Prepare a final presentation to showcase:

- Project overview and objectives.
- Demo of the scheduler's functionality.
- Key challenges and solutions.

Version Control

Use Git effectively:

Commit often with clear messages.

Track individual contributions via branches and pull requests.

Would you like detailed guidance on a specific aspect of this project, such as implementing one of the scheduling algorithms or setting up the CI/CD pipeline?