Mid Exam

Semester 20243

Subject	: **Distributed and Parallel System**
Study Program	: Informatics
Student Name	: Berliano Keio A.T, M. Arkaan Milzam
Student ID	: 001202300158, 001202300122

Instructions for Students

1. This examination consists of 2 problems.
2. All answers to be written in answer space provided with numbers are written accordingly.
3. Students are to use proper English and are required to write neatly and clearly.
4. Any attempts on cheating and plagiarism will result in an immediate zero score.

You are being tasked to form a group of 1 - 2 students and create a project to handle both **Backend** and **Frontend** of a web application. In order to make sure everyone has the same standardized environment and softwares, you decided to implement a container-based approach using Docker.

1. Create a frontend application using any framework / platform of your choice
2. Create a backend application using any framework / platform of your choice
3. Both point number 1 and 2 must have a Dockerfile
4. Create a docker-compose.yml file to handle deployment process

To document this better, create a report that shows what kind of application you are creating including the screenshots. **The important aspect is you need to explain and put the step-by-step process of deploying the application using Docker**. Put all source code on GitHub including the report, then submit the link

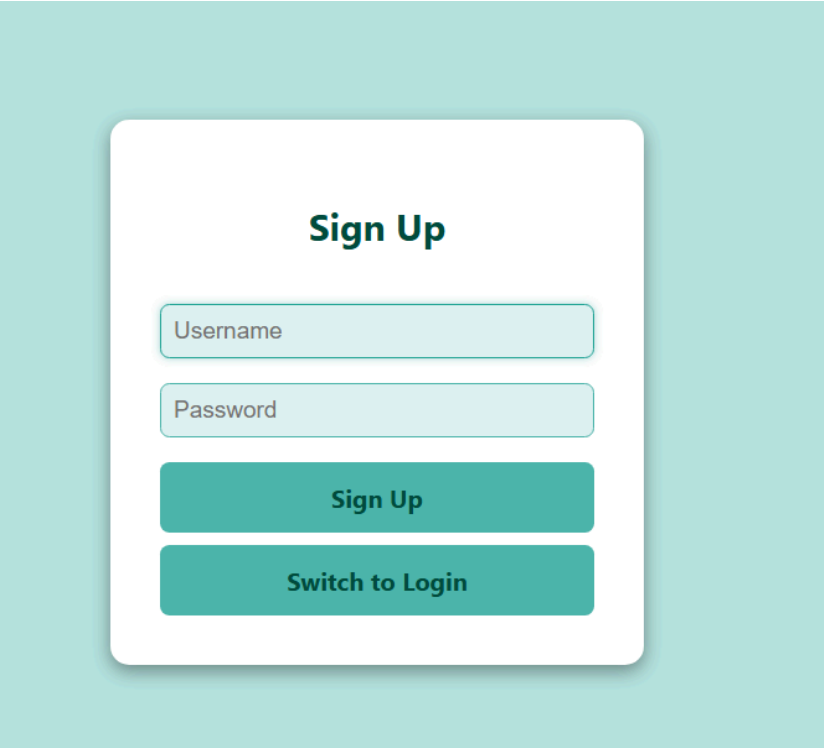
Report:
- Overview of app
- List of functions with accompanying screenshot

This is a scheduler app that helps users organize their tasks. This app is built using React as the frontend, Python (Flask framework) for the backend and Firebase as the database. Users are able to add and monitor their tasks. Users can also select the frequency of recurring tasks and organize them using this feature. Users are also able to edit and delete their task. To make tracking easier for users, this application divides each work according to its date.

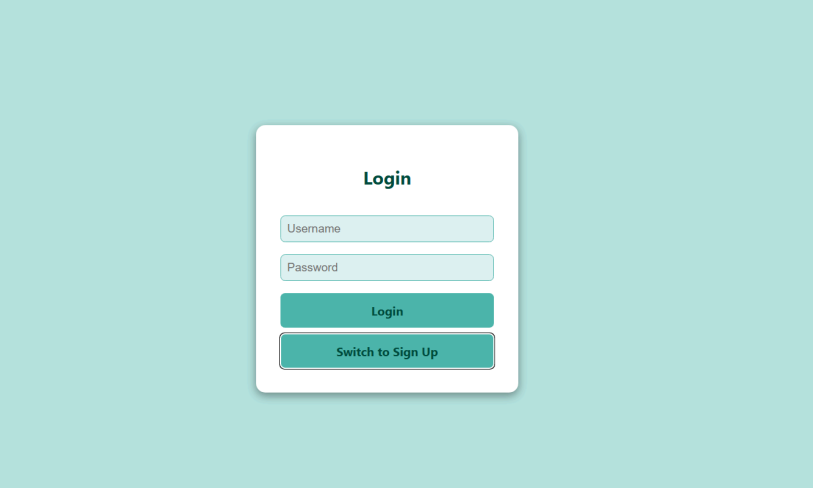
Github Link: https://github.com/Kaiki2/scheduler-app#

Screenshots of the app

1. Sign up Page

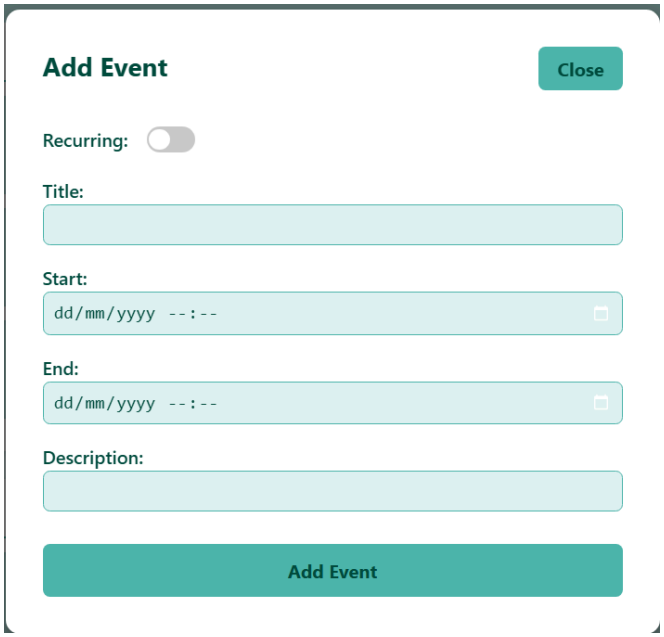


2. Login Page

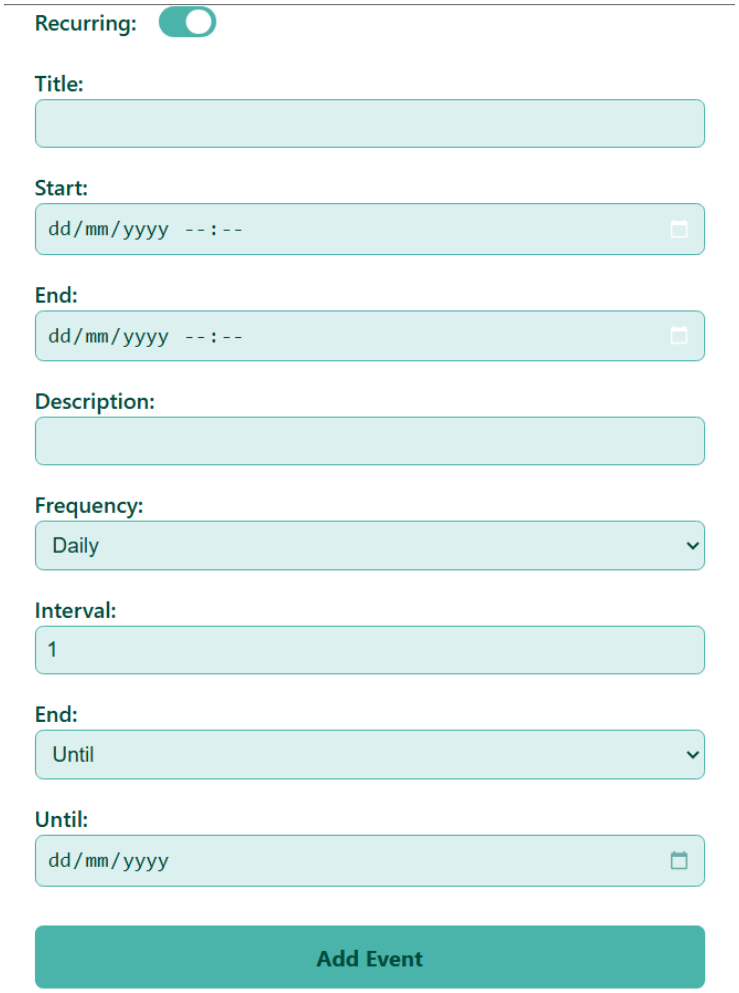


3. Dashboard

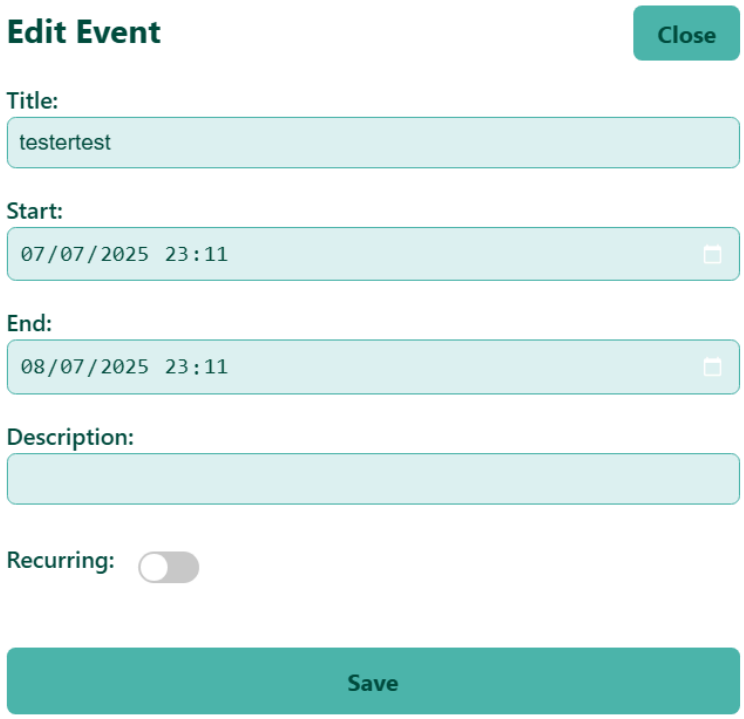
4. Add Event



5. Add Event (recurring)

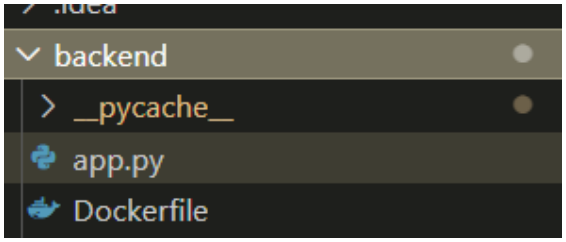


6. Edit Event



Setting up the Docker

1. Create a Docker file inside the frontend and backend folder



2. In the backend folder, create a file called “requirements.txt” to store everything that is needed to run the app

```
backend > requirements.txt
Flask
flask-cors
firebase-admin
python-dateutil
gunicorn
```

3. Frontend Docker file

```
# frontend/Dockerfile
FROM node:20

WORKDIR /app

COPY package*.json ./
RUN npm install

COPY . .

RUN npm run build

# Serve with a simple static file server
RUN npm install -g serve
CMD ["serve", "-s", "dist", "-l", "3000"]
```

This code builds the React frontend. It uses Node.js v20 and sets the working directory to /app. Then, it installs the dependencies from “package*.json”. Then, it copies the full frontend code into the image and runs the app using the command “npm run build”. Then, it will install the serve package globally and serves the static files from the “dist” folder on port 3000 using “serve”

4. Backend Docker file

```
# backend/Dockerfile
FROM python:3.11-slim

WORKDIR /app

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY . .

CMD ["flask", "run", "--host=0.0.0.0", "--port=5000"]
```

This code builds the Flask backend. It uses python version 3.11 slim for a lightweight container. It sets up the work directory in /app. Then it installs the requirements from the “requirements.txt”. Then, it copies the full backend code into the image. It then runs the Flask development server on port 5000 and binds all interfaces for Docker access.

5. Dockercompose.yml file

```
version: "3.9"

services:
  backend:
    build: ./backend
    ports:
      - "5000:5000"
    volumes:
      - ./backend:/app
    environment:
      - FLASK_APP=app.py
      - FLASK_ENV=development
```

```yaml
  frontend:
    build: ./frontend
    ports:
      - "3000:3000"
    environment:
      - NODE_ENV=production
```

This file connects the frontend and the backend. In the backend, it builds from the "./backend" directory. Then, map the port 5000 from the container to the host. It then mounts the local backend code as a volume for live development. Then it sets the environment variables to run Flask in development mode. In the frontend, it builds from the "./frontend" directory. Then, map the port 3000 from the container to the host. It then sets the Node environment to run in production mode.

How to deploy the app using Docker

1. Clone the repository from the github to your local computer.
2. Make sure all the files are present.
   IMPORTANT
   The project uses Firebase, and due to how GitHub repository security works (as well as security in general), I can't include my Firebase Service Account credentials in the repo. So, after cloning the repo, replace the contents of backend/firebase-service-account.json with the following before anything else.
   firebase-service-account.json


```
{
 "type": "service_account",
 "project_id": "scheduler-app-46237",
 "private_key_id":
"6e6caab516987c0c49f897f0580e004201f539c3",
 "private_key": "-----BEGIN PRIVATE
KEY-----\nMIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoI
BAQC29xxCWpsaCv1a\n9fXDQvJaQ9WcEECgreOZbdz+kBaqtnv/o3oXe/
WiXj6Mz+JRRUCp0tq/qg2ppnIr\nKrRjam3n8BsOahHwnkmLeL8ABwvgc
szfE1NYkoIHVq9TL5dad90xUQH7P0JIAiN5\nS+JKnTnGdr1GXRLm6JMl
OQpwqkIjq7WokF4qFw53QDw83EwY6IgLYq66ODn47p6d\nTsaczB+zm5K
```

```
AFPjtvAz2FBbq26rUNaMhM8WxvpWLP3ekb0Sc/ucs23sMwKhbwn6w\n0f
8/xS/LQlwygk23ZwWxWLoDNov7QJ5uDvwSdEzqxm/NgzrjnYL7cWDRfCz
8rv0v\nEOb1w2IxAgMBAAECggEAAzA2x3pmI/EqKtQo1xZzBssSPVIvBC
B08W2lDsZn72E3\nHjRvtQTO73LyZq2xNo7CbUCBxBpZ0FmN5ANuSbdh3
gC0wf6rxxxaW6eMIYXzPqiA\nJN4vTcFI1N6oIbxCsuaDkzT44pNUFlE7
jm+vIZ3QMyLgJ7+eYT8kivRNt/qkOimf\nw6cl69Giv5Iq5dgHHuEPTkq
eRRJCiAq5sVnJgJVB8P2ikONY3/dODaEaKGxPbnxR\ni5QLSi7SGGMdBm
hbGltxnoNE4E7GZe1AqMW+HQAufcJZMUi6OjKn2lSlClgBOm7w\nPUzMY
p8KJ3KGF/EcdYMUbl2Hn/36IsY8lMYSOo922QKBgQDd6Ck5rUXn3BamAt
wA\ne2L9BBsD2usmyyVQoSl5NOjN/U2zW4t+NZW0MVM2cZ4T0mfi2kXgo
VzXxXl4gE0P\nmMllm4q38BkTjhK/R+63Tekh5USjuCsQ5o/TrrCazxZd
DQwWxZ8Bhygr5L172PG0\nW2/vmaI2AkVGOWUZybbKwqBVuQKBgQDTE1V
tN1e5oJubb4ucAcFjwlIy1r0ttiYX\nc0gqBuiXP7ARqrwi+X1OoXgs4O
JjrGh4d117Nak2yBu9I9e/CrfEW70AEPR/J6Jf\nSUQRq+lf+3P3ayALg
gvtTJCO1n3PW9tU4QUdmLLMRBjEB7nBpSuGOZc0QbDMDNUT\nyLuXVjis
OQKBgQCRDUu9VuUmHTh0xdO3QPBAwWlDSNuDXTEEFoS8pPg0hgo/Xmy9\
nUBuOUNu67+PLwzKDF/irT+8XYA751KwKsO2czGKNv5U0sh7RCB0jlvr4
XBxVBHKq\nNnv3e+4XE5zjsD6huAYrAoUgxAGULgg4w7hUn1FoCm3r7GK
Iy9D/kt8piQKBgDb4\nlgr3mtUxStVUW5upqV/9v8KVwnchNRJuZ6DqY7
DgWU6AlLbXGxdbXhzyBIPI5vxY\nufPYqdUmkHLV3t5WQD3XVoWp5MMo4
1TFlAC2MPVaW31YXOQiXlDS2+MXYXSbi6ez\nYs2ucwJXyctj57DS/17Z
qNNnOFAsF2IrqQ9k2SWhAoGBAKqLZviGjeSg90TS2lig\nevuEkK95jDo
7zFwjM8JHMfwhsyp/+eicHh5p10gcZtWEyvz5NPqO8oCoIILlWols\nx+
VsdBZ8u8iCHqUQUPnDua3AAKq53HepAUgcxh2wj65rc6FjTWQgT1sjcTU
gQ8L1\n8Tkq1vUmJ4TH91VNBRpoiMvS\n-----END PRIVATE
KEY-----\n",
 "client_email":
"firebase-adminsdk-fbsvc@scheduler-app-46237.iam.gservice
account.com",
 "client_id": "108680784496325911680",
 "auth_uri": "https://accounts.google.com/o/oauth2/auth",
 "token_uri": "https://oauth2.googleapis.com/token",
 "auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",
 "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/fireba
se-adminsdk-fbsvc%40scheduler-app-46237.iam.gserviceaccou
nt.com",
 "universe_domain": "googleapis.com"
}
```


3. **Start up Docker Desktop**

4. Open up terminal in the root folder of the project (in this case scheduler-app), either by opening it straight from the folder or navigating to it in the terminal
6. To run the app, use the command “docker compose up -- build”
7. Once the app is online, navigate to localhost:3000 to access the app
8. To close the app, simply use CTRL + C in the terminal