MID EXAM REPORT

Project Title	**:**	**Presuniv Event Management**
Class	**:**	**IT 3**
Subject	**:**	**Distributed Parallel and System**

By:

Joy Adelia Sihombing	001202300077
Zelika Hannah	001202300067

INFORMATICS STUDY PROGRAM
FACULTY OF COMPUTER SCIENCE
PRESIDENT UNIVERSITY

INTRODUCTION

Presuniv Event Management website is a digital platform designed to manage, promote, and streamline access to events organized by President University. This platform was established to consolidate the structured dissemination of event-related information for students, with the aim of providing a single, credible source for the university community and external stakeholders. It combines user-oriented design with modern web technologies and containerized deployment through Docker, ensuring reliability, scalability, and operational efficiency.

OBJECTIVE

The primary goal of this project is to develop a comprehensive event management system that supports the effective organization, promotion, and coordination of events at President University. The system is intended to unify all event-related activities into a single, user-friendly digital platform, offering a more modern and efficient approach to event management.

This platform aims to enhance the user experience by making it easier for students and staff to discover and participate in campus events. It also assists university departments, faculties, and student organizations by simplifying the planning and management processes, while reducing administrative tasks through automation.

In addition to its practical functions, the system contributes to improving the university's professional image by ensuring all events are presented in a consistent and polished manner. By consolidating information and streamlining processes, the project ultimately seeks to improve coordination, increase engagement, and foster a stronger sense of community across the university.

FEATURES

The Presuniv Event website offers a set of essential features designed to support efficient event management and user engagement. These core functionalities include:

- **Event Listing and Discovery**
  A centralized and well-organized interface that displays upcoming events. Users can explore events based on categories, dates, and target audiences, ensuring relevant and efficient navigation.

- **Detailed Event Profiles**
  Each event is presented through a dedicated page that includes key details such as the event's purpose, schedule, speaker profiles, venue, and offering a complete overview to prospective participants.

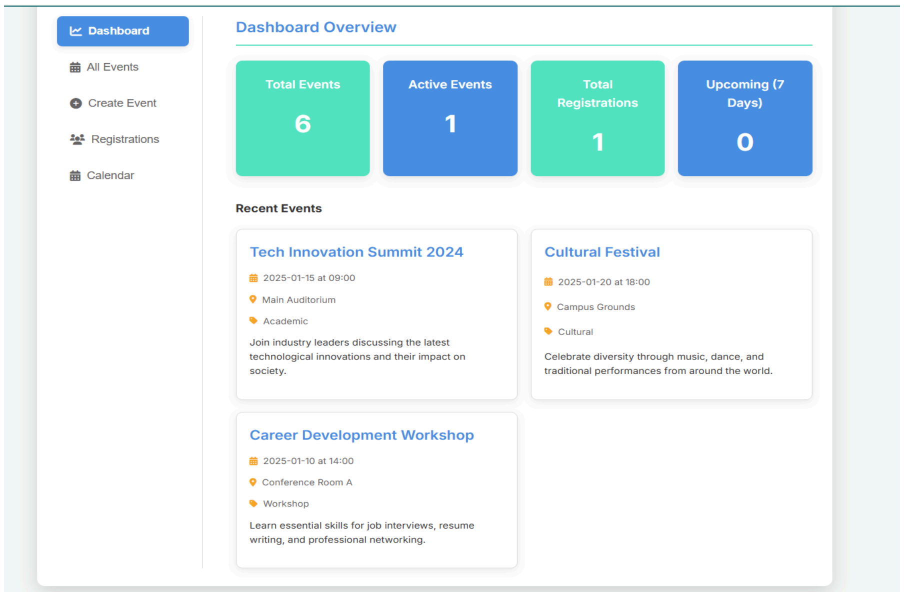
- **Registration and Ticketing System**
  The system allows users to register for events easily, whether free or ticketed. It supports secure online payments, generates digital tickets, and sends automated confirmation emails.

- **Event Submission for Internal Organizations**
  Authorized university departments and student organizations can submit their event details through a dedicated portal. Submissions are reviewed by administrators before being published, ensuring quality and consistency across the platform.

- **Advanced Search and Filtering**
  A search engine equipped with filters for keywords, event types, and dates helps users find specific events quickly and efficiently.

RESULT

Following its deployment, the Presuniv Event Website provides a seamless and user-friendly platform for browsing, submitting, and registering for events at President University. The system is responsive, functionally complete, and easily deployable across various environments with minimal configuration. As a result, the website serves as a reliable and modern hub for event-related activities, enhancing communication, engagement, and participation across the university community.



Dashboard Overview

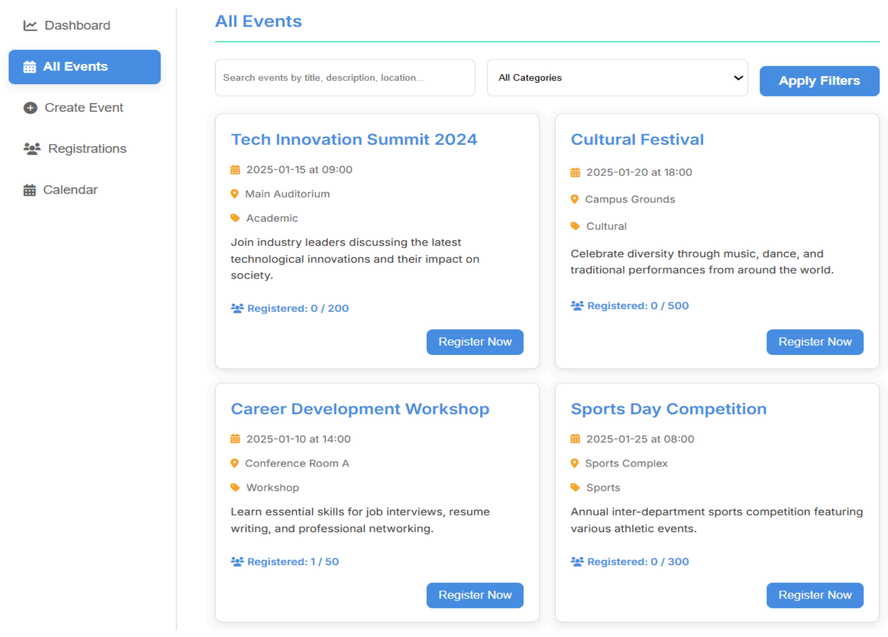
Total Events	Active Events	Total Registrations	Upcoming (7 Days)
6	1	1	0

Recent Events

Tech Innovation Summit 2024
2025-01-15 at 09:00
Main Auditorium
Academic
Join industry leaders discussing the latest technological innovations and their impact on society.

Cultural Festival
2025-01-20 at 18:00
Campus Grounds
Cultural
Celebrate diversity through music, dance, and traditional performances from around the world.

Career Development Workshop
2025-01-10 at 14:00
Conference Room A
Workshop
Learn essential skills for job interviews, resume writing, and professional networking.



All Events

Tech Innovation Summit 2024
2025-01-15 at 09:00
Main Auditorium
Academic
Join industry leaders discussing the latest technological innovations and their impact on society.
Registered: 0 / 200
Register Now

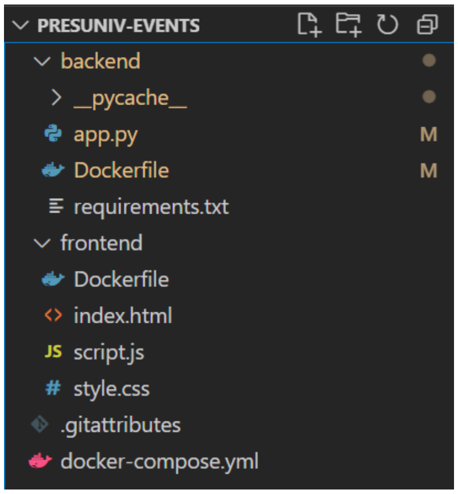
Cultural Festival
2025-01-20 at 18:00
Campus Grounds
Cultural
Celebrate diversity through music, dance, and traditional performances from around the world.
Registered: 0 / 500
Register Now

Career Development Workshop
2025-01-10 at 14:00
Conference Room A
Workshop
Learn essential skills for job interviews, resume writing, and professional networking.
Registered: 1 / 50
Register Now

Sports Day Competition
2025-01-25 at 08:00
Sports Complex
Sports
Annual inter-department sports competition featuring various athletic events.
Registered: 0 / 300
Register Now

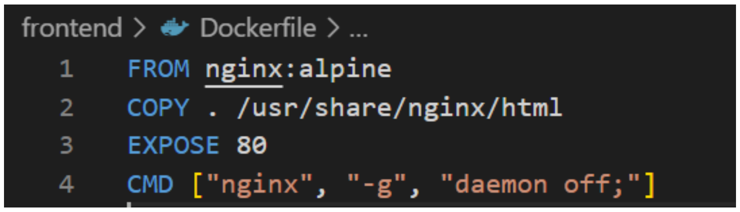
Step-by-Step: Deploying the Application Using Docker

Step 1
Make a project structure, the application directory should be organized well



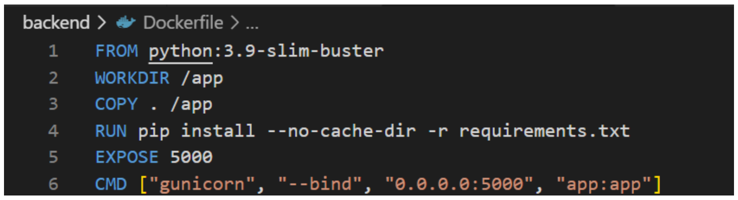
```
PRESUNIV-EVENTS
  backend
    __pycache__
    app.py                 M
    Dockerfile             M
    requirements.txt
  frontend
    Dockerfile
    index.html
    script.js
    style.css
  .gitattributes
  docker-compose.yml
```

Step 2
Create a Dockerfile for both Frontend and Backend. The Dockerfile contains instructions for building the application container image.



frontend > Dockerfile > ...

```dockerfile
FROM nginx:alpine
COPY . /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

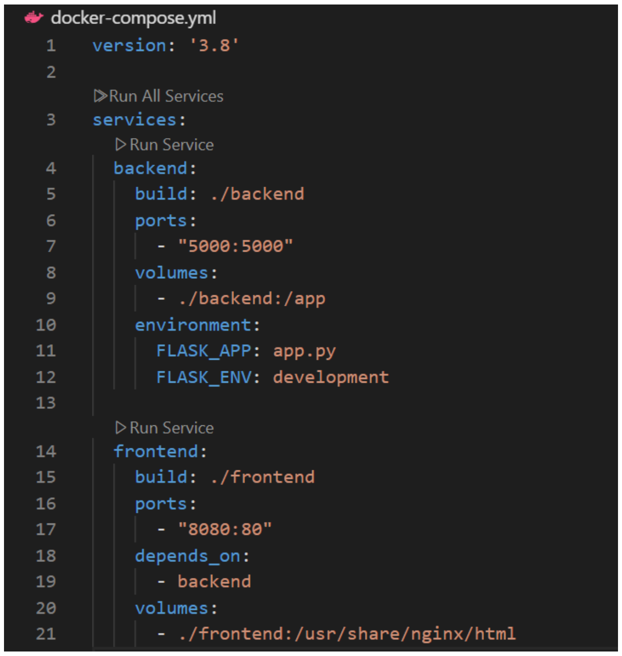


backend > Dockerfile > ...

```dockerfile
FROM python:3.9-slim-buster
WORKDIR /app
COPY . /app
RUN pip install --no-cache-dir -r requirements.txt
EXPOSE 5000
CMD ["gunicorn", "--bind", "0.0.0.0:5000", "app:app"]
```

Step 3

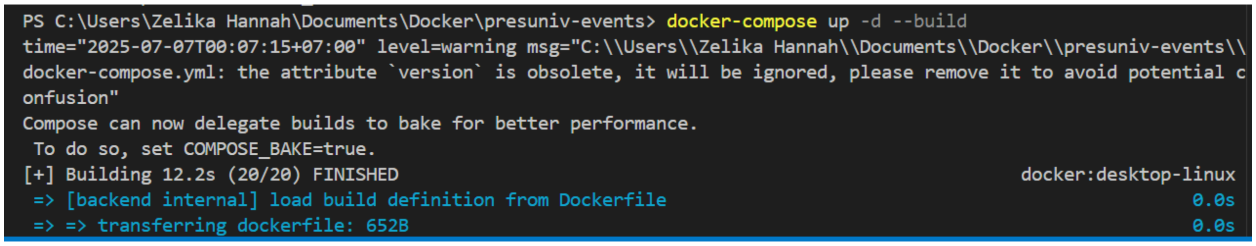
Define the docker-compose.yml file to define how different containers work together.



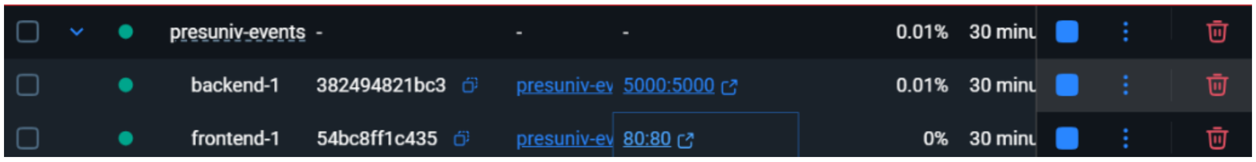
```yaml
version: '3.8'

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

  frontend:
    build: ./frontend
    ports:
      - "8080:80"
    depends_on:
      - backend
    volumes:
      - ./frontend:/usr/share/nginx/html
```

Step 4

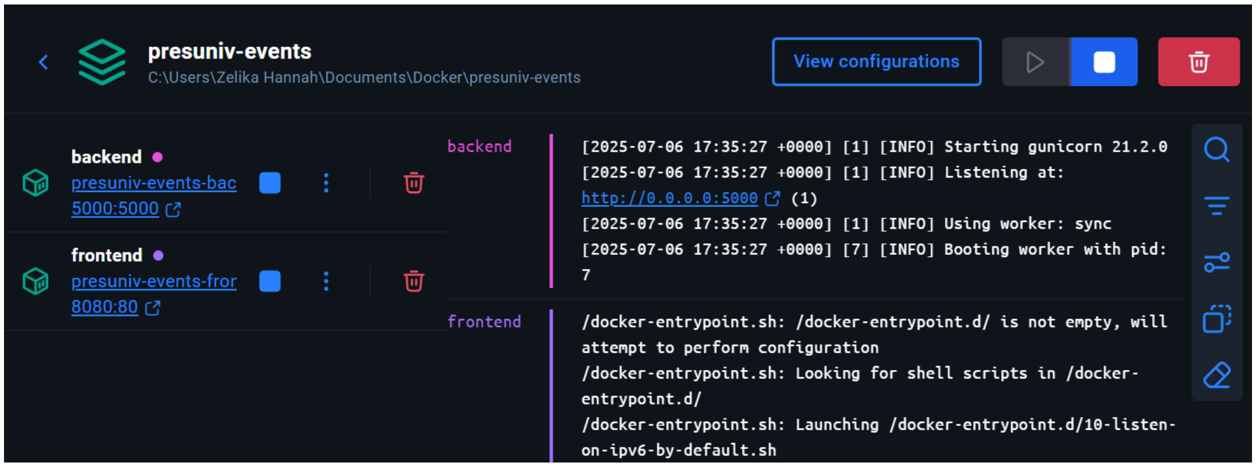
Build and run the application



```
PS C:\Users\Zelika Hannah\Documents\Docker\presuniv-events> docker-compose up -d --build
time="2025-07-07T00:07:15+07:00" level=warning msg="C:\\Users\\Zelika Hannah\\Documents\\Docker\\presuniv-events\\
docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential c
onfusion"
Compose can now delegate builds to bake for better performance.
 To do so, set COMPOSE_BAKE=true.
[+] Building 12.2s (20/20) FINISHED                                                    docker:desktop-linux
 => [backend internal] load build definition from Dockerfile                                           0.0s
 => => transferring dockerfile: 652B                                                                   0.0s
```



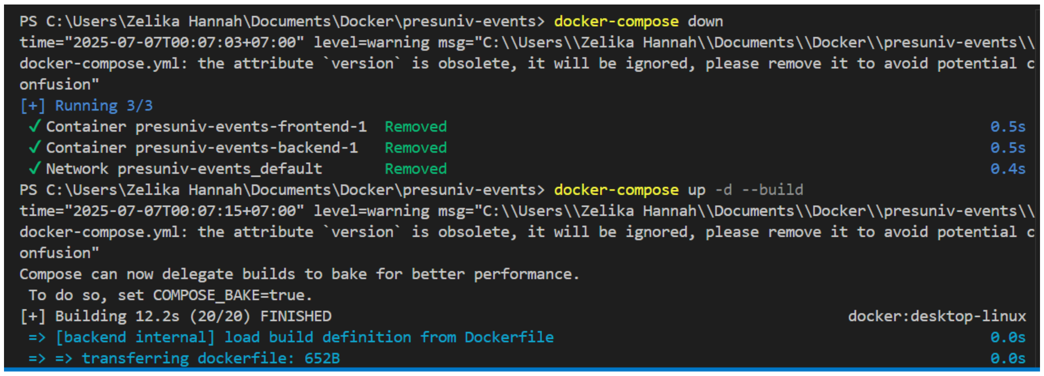
```
presuniv-events  -                   -                   -           0.01%  30 minu
   backend-1     382494821bc3   presuniv-ev  5000:5000               0.01%  30 minu
   frontend-1    54bc8ff1c435   presuniv-ev  80:80                   0%     30 minu
```



```
backend   [2025-07-06 17:35:27 +0000] [1] [INFO] Starting gunicorn 21.2.0
          [2025-07-06 17:35:27 +0000] [1] [INFO] Listening at:
          http://0.0.0.0:5000 (1)
          [2025-07-06 17:35:27 +0000] [1] [INFO] Using worker: sync
          [2025-07-06 17:35:27 +0000] [7] [INFO] Booting worker with pid:
          7
frontend  /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will
          attempt to perform configuration
          /docker-entrypoint.sh: Looking for shell scripts in /docker-
          entrypoint.d/
          /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-
          on-ipv6-by-default.sh
```

Step 5

If there are any changes or updates to the code, you can modify the source files and then use the command `docker-compose down`, followed by `docker-compose up -d --build`, to rebuild and restart the application with the latest changes.



```
PS C:\Users\Zelika Hannah\Documents\Docker\presuniv-events> docker-compose down
time="2025-07-07T00:07:03+07:00" level=warning msg="C:\\Users\\Zelika Hannah\\Documents\\Docker\\presuniv-events\\
docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential c
onfusion"
[+] Running 3/3
 ✓ Container presuniv-events-frontend-1  Removed                                                            0.5s
 ✓ Container presuniv-events-backend-1   Removed                                                            0.5s
 ✓ Network presuniv-events_default       Removed                                                            0.4s
PS C:\Users\Zelika Hannah\Documents\Docker\presuniv-events> docker-compose up -d --build
time="2025-07-07T00:07:15+07:00" level=warning msg="C:\\Users\\Zelika Hannah\\Documents\\Docker\\presuniv-events\\
docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential c
onfusion"
Compose can now delegate builds to bake for better performance.
 To do so, set COMPOSE_BAKE=true.
[+] Building 12.2s (20/20) FINISHED                                                       docker:desktop-linux
 => [backend internal] load build definition from Dockerfile                                                0.0s
 => => transferring dockerfile: 652B                                                                        0.0s
```

FINAL REFLECTION

I. Team Contribution

Name	Contribution
Joy Adelia	● Frontend Developer ● Reported bugs and coordinated fixes ● Fill in the report file
Zelika Hannah	● Backend Developer ● Deployed the website ● Fill in the report file

II. Learning Outcomes

1. **Implementing Docker for deployment**
   Our team learned how to containerize web applications using Docker, allowing us to deploy the project consistently across various environments. We explored the creation of Dockerfile and docker-compose.yml, which taught us how to automate the deployment of multi-service applications using containers.

2. **Designing a full-stack web application**
   We gained practical experience in building a full-stack application from front-end interface design to back-end. This included the integration of user interactions, data validation, database communication, and server-side logic for registration and ticketing systems.

3. **Using NGINX**
   We learned to configure NGINX to act as a reverse proxy, which improved the performance, scalability, and security of our web application. This also gave us better control over routing traffic to the application containers.

4. **Understanding and Resolving CORS (Cross-Origin Resource Sharing)**
   A critical learning outcome was understanding why browsers block cross-origin requests by default and how CORS policies work. We learned to configure Flask-cors on the backend to securely allow requests from the frontend running on a different *origin* (domain/port), effectively resolving common "Network error: Failed to fetch" issues.

5. **Handling real-world development workflow**
   Throughout the development cycle, we practiced version control using Git, and communicated effectively to align front-end and back-end work. This experience has improved our ability to work collaboratively in a simulated professional environment.

III. Challenges

1. **Port Conflicts During Docker Deployment**
   One significant initial challenge was resolving port conflicts, especially with ports 80 and 5000, which were often already in use by other applications on the host system (like XAMPP or other application instances). This caused Docker containers to fail to bind the necessary ports, resulting in the error message Bind for 0.0.0.0:<port> failed: port is already allocated. We resolved this by identifying and stopping the conflicting processes or by changing the port mappings in docker-compose.yml.

2. **Cross-Origin Communication Difficulties (CORS)**
   Even though both the backend and frontend were running on localhost, using different ports caused browsers to enforce CORS policies, leading to the Network error: Failed to fetch. We overcame this by adding and configuring Flask-Cors on the backend to allow requests from the frontend's origin.

3. **Time Constraints and Task Management**
   Balancing multiple tasks such as UI design, database configuration, and deployment within a limited timeframe was challenging. At times, tasks were delayed due to overlapping responsibilities. This highlighted the need for better task distribution and the importance of detailed planning before development begins.

IV. Github Link

Github: [Presuniv Events Web]