Docker Setup Documentation

Step-by-Step Guide

1. Clone the Repository

Open your terminal and run:
- git clone https://github.com/KHRISTMAE/Valid8-Attendance-Recognition-System
- Cd C:\Users\Wisdom Seed\Valid8-Attendance-Recognition-System

2. Create Dockerfile for Backend
Location: C:\Users\Wisdom Seed\Valid8-Attendance-Recognition-System\backend updated

```
# Use official Python base image
FROM python:3.10-slim

# Set working directory inside the container
WORKDIR /app

# Install build tools and CMake
RUN apt-get update && apt-get install -y \
    cmake \
    build-essential \
    libglib2.0-0 \
    libsm6 \
    libxext6 \
    libxrender-dev \
    && rm -rf /var/lib/apt/lists/*

# Copy the requirements.txt file into the container
COPY requirements.txt .

# Install the Python dependencies
RUN pip install --default-timeout=100 --no-cache-dir -r requirements.txt

# Copy all backend source code into the container
COPY . .

# Expose the port FastAPI will run on
EXPOSE 8000

# Command to run the FastAPI app with hot reload
CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8000", "--reload"]
```

3. Create Dockerfile for Frontend

Location: C:\Users\Wisdom Seed\Valid8-Attendance-Recognition-System\frontend updated

```
# Use official Node.js image
FROM node:20

# Set working directory inside the container
WORKDIR /app

# Copy only dependency files first to leverage Docker caching
COPY package.json package-lock.json ./

# Install dependencies using npm
RUN npm install
```

```
# Copy the rest of your app code
COPY . .

# Expose the port used by Vite dev server
EXPOSE 5173

# Start Vite dev server
CMD ["npm", "run", "dev"]
```

4. Create docker-compose.yml File

Location: C:\Users\Wisdom Seed\Valid8-Attendance-Recognition-System

```
version: '3.9'

services:
  backend:
    build: ./backend
    container_name: backend
    ports:
      - "8000:8000"
    environment:
      - DATABASE_URL=postgresql://postgres:new_secure_password123!@100.70.139.24:5432/fastapi_db
    volumes:
      - ./backend:/app
    restart: unless-stopped

  frontend:
    build: ./frontend
    container_name: frontend
    ports:
      - "5173:5173"
    depends_on:
      - backend
    working_dir: /app
    volumes:
      - ./frontend:/app
      - /app/node_modules      # Avoids syncing host node_modules
    command: npm run dev
     restart: unless-stopped
```

5. Build and Run the Containers

Run the following command in your project directory:

```
docker-compose up --build
```

6. Access the Services

- Frontend: http://localhost:3000
- Backend API Docs: http://localhost:8000/docs
- PostgreSQL DB: Port 54327. Stop the Services

To stop and remove containers:
```
docker-compose down
```

To stop and remove containers with volumes:
```
docker-compose down -v
```

Environment Variables (Optional)
Create a .env file with:

```
POSTGRES_USER=valid8_user
POSTGRES_PASSWORD=valid8_password
POSTGRES_DB=valid8_db
```

And reference it in docker-compose.yml with:

```
env_file:
- .env
```

NOTE:

- Make sure **Docker Desktop** or **Docker Engine** is properly installed and running on your machine.
- Ensure **Docker Compose** is installed (if using older Docker setups, as newer Docker Desktop includes it by default).
- If you're using **Windows**, run your terminal as **Administrator**.

On the first run, Docker might take a few minutes to download images (like python:3.11-slim, nginx:alpine, and postgres:14).

If you encounter **port conflicts** (e.g., if ports 5432, 8000, or 3000 are already in use), you'll need to adjust the ports section in your docker-compose.yml.

The **PostgreSQL database data is saved in a Docker volume** named db_data, so it persists even after stopping the containers.

You can view running containers using:

```
docker ps
```

And stop them individually using:

```
docker stop <container_name>
```