Medical Transport & Equipment System – Setup Instructions

This guide explains how to run the Flask application and PostgreSQL database, and how to restore the database using the provided backup file.

1. Install Requirements

Make sure you have the following installed:
- Python 3.9 or higher
- PostgreSQL + pgAdmin (either locally or via Docker)
- pip (Python package manager)

Install the required Python packages by running:

```
pip install flask psycopg2-binary
```

2. Database Setup Options

You can set up PostgreSQL in one of two ways:
OPTION A – Using Docker (Recommended for portability)
If you don't have PostgreSQL or pgAdmin installed locally, you can use Docker to run them in containers:

1. Create a file called `docker-compose.yml` in your project folder with the following content:

```
version: '3'
services:
  db:
    image: postgres:13
    environment:
      POSTGRES_USER: your_username
      POSTGRES_PASSWORD: your_password
      POSTGRES_DB: mydatabase
    ports:
      - "5432:5432"
    volumes:
      - pgdata:/var/lib/postgresql/data

  pgadmin:
    image: dpage/pgadmin4
    environment:
      PGADMIN_DEFAULT_EMAIL: admin@example.com
      PGADMIN_DEFAULT_PASSWORD: admin
    ports:
      - "5050:80"
    depends_on:
      - db
```

```
volumes:
  pgdata:
```

2. Replace `your_username` and `your_password` with values of your choice.

3. Start the containers:

```
docker-compose up -d
```

4. Access pgAdmin at:

```
http://localhost:5050
Login:
Email: admin@example.com
Password: admin
```

<u>OPTION B</u> – Use PostgreSQL Installed Locally
If you already have PostgreSQL and pgAdmin installed on your computer (not in Docker), you can skip the `docker-compose.yml` file completely.
Make sure your PostgreSQL server is running on:

```
Host: localhost
Port: 5432
```

You can continue directly to restoring the database in pgAdmin.

3. Restore the Database Using pgAdmin

1. Open pgAdmin (either from Docker or locally).

2. Connect to the server using:
   - Host: `localhost` (or `db` if using Docker)
   - Username: the value you set as POSTGRES_USER
   - Password: the value you set as POSTGRES_PASSWORD

3. Locate the database `mydatabase` (or create it if missing).

4. Right-click the database → Restore → choose the provided backup file (e.g., `backup.sql`)

5. Select Format: "Custom" or "Plain" (depending on the file type) → Click Restore.

4. Update Connection Details in app.py

Open `app.py` and update the following function with your actual DB credentials:

```
def get_db_connection():
    return psycopg2.connect(
        host='localhost',
        port=5432,
        user='your_username',
        password='your_password',
        dbname='mydatabase'
    )
```

5. Run the Flask Application

In your project folder, run:

```
python app.py
```

Then open your browser at:

```
http://localhost:5000
```

6. Navigating the Web Application

From there, you can navigate between the different sections using the top navigation bar.

Home Page
- Shows a list of volunteer birthdays that occur during the current week.
- Provides a general description of the system.
- Includes links to all other sections of the application.

Volunteers
- View a list of all volunteers along with their personal details and assigned type: Driver, Service Assistant, or Transport Assistant.
- Add a new volunteer by filling in general details and type-specific fields.
- Edit or delete existing volunteers.
- Displays the top 3 most active volunteers for the current month (based on past rides).

Patients
- View all registered patients and their information, including disability status and assigned medical equipment.
- Add new patients.
- Edit or delete existing patient records.
- Patients assigned to a ride cannot be deleted due to database constraints.

Rides
- View all scheduled rides including date, time, patient, driver, optional assistant, and destination.
- Add new rides by selecting from existing records.
- Edit the ride's date and pickup time.
- Delete rides when allowed by foreign key constraints.
- Includes a button to automatically assign available assistants to future rides.

Borrowed Equipment
- View a list of all borrowed equipment records, including patient, product, borrow date, and service center.
- Add new borrow records.
- Mark items as returned by updating their status.
- Only records marked as returned can be deleted.

Expired Borrowed Equipment
- Accessible only from within the "Borrowed Equipment" page.
- Displays borrowed items that have not been returned and are linked to products with expired warranties.
- Includes patient name, phone number, product name, and warranty expiration date.

Notes:

- If you use Docker: use `db` as the host when connecting from pgAdmin inside the container.
- If you installed PostgreSQL locally: use `localhost` as the host.
- The system includes management of volunteers, patients, rides, and equipment loans.