Docker One-Page (Project Overview)

Purpose

Docker in this project provides a reproducible development environment for both backend and frontend, so all team members run the same stack with the same dependencies.

Compose Services

File: `/projectRoot/docker-compose.yml`

There are 2 services:

1. `backend`
2. `frontend`

Key behavior:

1. `backend` runs Django migrations, seeds roles, then starts server on `8000`.
2. `frontend` runs Vite dev server on `5173`.
3. `frontend` depends on `backend`.

Ports

1. Backend: `8000:8000`
2. Frontend: `5173:5173`

Local access:

1. API: `http://localhost:8000`
2. App UI: `http://localhost:5173`

Volumes

1. Backend mount:

```
- ./backend:/app
```

2. Frontend mounts:

```
- ./frontend:/app
- /app/node_modules
```

Why:

1. Live-reload development without rebuilding for each code change.
2. Node modules stay inside container path to avoid host/container mismatch.

Commands

Run full stack:

```
cd /projectRoot
docker compose up
```

Run only one service:

```
cd /projectRoot/backend
docker compose up
cd /projectRoot/frontend
docker compose up
```

Stop:

```
docker compose down
```

Logs:

```
docker compose logs -f backend
docker compose logs -f frontend
```

Benefits

1. Same runtime for all developers.
2. Quick onboarding.
3. Cleaner dependency management.
4. Better CI/CD compatibility.

Current Dev vs Production Note

Current setup is development-oriented (`runserver` , Vite dev mode).
For production, use built frontend static assets + production WSGI/ASGI server for Django.