**Architecture for Web UI**

**1. Frontend (Client Interface)**

* A lightweight **React** or **Streamlit** (simpler) app:
  + **Census Input Form**: upload or manually enter daily census.
  + **Staffing Grid Editor**: editable table for RN/NA ratios by census + season.
  + **Resources Page**: manage staff, FTEs, availability, and leave (dropdown for names to avoid typos).
  + **Shifts Page**: configure shifts per role (drag/drop or dropdown-based).
  + **Run Planning Button**: triggers calculation.
  + **Reports Tab**: download Excel or view charts inline (shortages highlighted).

**2. Backend (Core Logic)**

* **Python FastAPI** or **Flask** service that:
  + Loads census, staffing grid, resources, shifts (from DB or config).
  + Runs the **same pipeline you already built** (staffing plan → position control → schedule).
  + Returns results as JSON to frontend.
  + Optionally exports an **Excel or PDF report** on demand.

**3. Database / Storage**

* Replace Excel input sheets with:
  + **SQLite/Postgres** for staff + census + resources.
  + Or keep them in **CSV/YAML configs** (if client wants to stay file-based).
* Results can be stored historically (so client can view past schedules).

**4. Deployment**

* **Dockerized app** → host on client’s intranet or a cloud server (AWS, Azure, GCP).
* Authentication (if needed) with simple login.