**Step 1 — Project Scaffold**  
In this step I created the initial system scaffold. I set up a FastAPI backend with endpoints for health checks, requirements retrieval, requirement matching, and smart report generation. I also defined Pydantic models for business inputs and regulatory rules, and added a simple matching engine and AI report stub. On the frontend, I built a minimal RTL-friendly questionnaire form in Hebrew, which collects business size, seating, and features, then displays matched requirements and a generated report. To support development and testing, I prepared a sample JSON dataset with representative regulatory items (gas usage, meat handling, delivery, seating limits). Finally, I added configuration files (requirements.txt, .env.example), a run script, and a README with setup instructions. This gave me a complete end-to-end skeleton using sample data, ready for later integration with real PDF parsing and AI report generation.  
  
**Step 1 — Data Processing (PDF → JSON)**  
In this step I implemented the foundation for handling the regulatory data. I created a parser script (backend/process\_pdf.py) that reads the official licensing PDF (backend/data/18-07-2022\_4.2A.pdf) and automatically converts it into structured JSON (backend/data/requirements.json). The script uses pdfplumber to extract the raw text, cleans and splits it into logical requirement blocks, and assigns each block a structured schema with fields such as id, category, title, description, priority, and conditions. The backend API was updated to prefer this generated JSON while falling back to a sample dataset if needed. This ensures the system can run locally on any machine without absolute paths and provides a clean, machine-readable knowledge base that later steps (matching engine and AI report generation) will rely on.  
  
**Step 2 — Digital Questionnaire (extended inputs)**  
In this step I expanded the questionnaire to capture a richer set of business attributes directly from the user. In addition to the core fields (business size and seating capacity), the form now collects the business area in square meters, the number of staff per shift, and a wide range of operational features such as alcohol serving, outdoor seating, music/entertainment, smoking area, kitchen type (hot/cold), dietary focus (dairy, fish, vegan), late-night operation, takeaway, and technical/hygiene items like grease traps, ventilation hoods, fire extinguishers, refrigeration, allergen notes, accessibility, signage, pest control, waste separation, and valid gas checks. These inputs are validated in the backend against an allowed features list, and the matching engine was extended to handle numeric thresholds (min/max area, min/max staff) alongside features. I also added sample requirements demonstrating how these new questions trigger specific rules (e.g., alcohol license, hot-kitchen ventilation, staff-based restroom requirements). This makes the questionnaire much closer to real-world regulation and enables more accurate, personalized compliance reports.

**Step 3 — Matching Engine & Results Dashboard**  
In this step I implemented the matching engine that takes the business profile (size, seating capacity, area, staff per shift, and selected features) and filters the processed requirements dataset to only the relevant items. The engine supports size\_any, numeric thresholds (min/max\_seats, min/max\_area\_sqm, min/max\_staff), and feature logic (features\_any, features\_all, features\_none). Each matched requirement includes an explanation (reasons) describing why it applies, and results are sorted by priority (High → Medium → Low). To make the output more usable, I added a **statistics dashboard** that summarizes matches by category and priority, along with **filtering controls** so users can quickly narrow down the list to the most important obligations. Together, this transforms the raw matching results into a clear, navigable, and actionable compliance view.

**Step 4 — AI Smart Report**  
In the final step I integrated an AI-powered report that transforms the matched regulatory requirements into a clear, personalized document for the business owner. The backend builds a structured Hebrew prompt from the business profile and the relevant requirements, and if an API key is provided it generates a live report using an LLM (e.g., GPT). Without a key, the system falls back to a mock report so the demo always runs locally. In the UI, the report is presented in a clean modal titled **“דוח חכם – AI Report”**, with the project name highlighted (“Licensure Buddy IL – דוח רישוי עסק”). The modal shows a **business summary section** (size, seating, features) followed by the generated report text. At the bottom there are action buttons for **Copy** and **Download (.txt)**, and a badge indicating whether the output came from a real model or mock mode. This design makes the AI feature immediately visible and easy to use, while giving the recruiter a professional, end-to-end impression of the system.