**SHL Assessment Recommender Web App – Project Summary**

**Objective**

To design a smart and user-friendly web application that recommends **SHL individual tests** based on **natural language queries** such as job descriptions or skill requirements making assessment selection **intelligent**, **context-aware**, and **easy to use**.

**How I Tackled It**

**1. Frontend (React + Tailwind CSS)**

* Developed using **React (Vite)** for blazing-fast performance and modular architecture.
* Used **Tailwind CSS** to build a clean, responsive, and accessible UI without writing custom CSS.
* Key Components:
  + 🔹 QueryForm.jsx:
    - Takes free-form natural language input (e.g., *“Looking for a finance analyst test for graduates”*).
    - Built with form validation, focus states, and smooth UX transitions.
    - Sends data asynchronously via fetch() to the backend.
  + 🔹 ResultsDisplay.jsx:
    - Dynamically displays test recommendations in a clean list with title, skill, level, and estimated duration.
    - Styled for readability and interactivity (hover, spacing, mobile-friendliness).

**2.Backend (FastAPI + LLM Intelligence)**

* The **/recommend** API endpoint, built with **FastAPI**, is deployed on **Railway.app** for easy and scalable hosting.
* This endpoint integrates a **Large Language Model (LLM)** to:
  + Parse and understand unstructured job descriptions.
  + Convert queries into vector embeddings.
  + Match semantic content with SHL assessment metadata using **cosine similarity**.
* LLM adds depth and intelligence — understanding what the user *means*, not just what they *type*.

**3.Tech Stack Summary**

| **Tech / Tool** | **Role** |
| --- | --- |
| **React.js (Vite)** | Frontend framework for SPA |
| **Tailwind CSS** | Utility-first CSS for styling |
| **FastAPI** | Lightweight, async Python backend |
| **Railway.app** | Deployment platform for backend |
| **LLM (e.g., OpenAI)** | Semantic understanding of user queries |
| **Fetch API (JS)** | Client-server communication |

**4. Key Highlights & Contributions**

* **LLM-Driven Query Intelligence**: Users can enter real-world job descriptions and get personalized test suggestions — no dropdowns, no filters, just plain English.
* **End-to-End Development**: Built both the client and backend from scratch, ensuring tight integration and smooth data flow.
* **Beautiful UI/UX**: Focused on accessibility, responsiveness, and simplicity while ensuring functionality.
* **Deployed & Production-Ready**: App is hosted and live, with a real inference engine backing recommendations.

**5. What I Achieved**

* **Smart Recommendations:** Job descriptions are parsed and matched to the most appropriate SHL tests.
* **Beautiful, Functional UI:** Minimal yet impactful design that’s easy to use and understand.
* **Fast & Scalable:** Lightweight React build with scalable API calls—ready for real-world use.
* **Ownership:** Took full ownership of frontend logic, styling, API integration, and testing.

**6. Final Thoughts**

This project reflects not just technical know-how, but a strong focus on **user experience**, **modular architecture**, and **real-world problem solving**. From building responsive UI components to integrating a hosted ML backend, every part was crafted with care, precision, and a deep understanding of the product goal.