Interview Tasks

Introduction

Thank you for taking the time to complete our technical assessment. This project is designed to give us insight into your skills across the entire stack, from backend services and machine learning to frontend implementation and architectural thinking.

A Note on Your Time

We value your time and have designed this task to be completed in stages. We expect the **Core Tasks (1 and 2)** to take approximately **6-8 hours**. Please focus on quality, clarity, and testing over completing every single feature. The **Advanced Task (Task 3)** and the bonus features in Task 2 are optional and provide an opportunity for you to showcase additional skills if you have the time.

<u>**Task 1**</u>: Housing Price Prediction Model API (Core Task)

Objective:
Build, containerise, and deploy a simple regression model that predicts housing prices based on provided features (dataset attached).

Requirements:

- Api Endpoints:
    - predict - Accepts housing features and returns price predictions. Both single and batch numbers
    - model-info - Returns model coefficients and performance metrics
    - health - Simple health check endpoint

Technical Constraints:

- Python 3.12+
- FastAPI
- Scikit-learn

Deliverables:

1- Source code in github
2- Dockerfile
3- Ability to show this live during interview via (Swagger/OpenAPI)

Task 2: Multi-Application Next.js Portal (Core Task)

Objective:
Create a unified Next.js portal that hosts two independent applications with different backend technologies, both capable of interacting with the ML model from Task 1.

Portal Structure Requirements:

1- Unified Navigation and Layout:
   a. Implement a shared layout with navigation between applications
   b. Use Next.js App Router for routing between and within applications
   c. Create a consistent design system across both applications
   d. Properly handle loading and error states at the layout level

2- App 1: Property Value Estimator (Python Backend)
   a. Frontend:
      i. Create a form for inputting property details (all fields from the model)
      ii. Implement client-side validation with appropriate error messages
      iii. Display prediction results in both tabular format and visual chart
      iv. Implement a history feature showing previous estimates
      v. Create a comparison view to analyse multiple properties side-by-side
   b. Backend (Python):
      i. Handle form submissions
      ii. Integrate with regression model container from Task 1
      iii. Implement data validation and error handling

3- App 2: Property Market Analysis (Java Backend)
   a. Frontend:
      i. Create an interactive dashboard with property market visualisations
      ii. Implement filters for analysing different property segments
      iii. Build a "what-if" analysis tool using the model
      iv. Provide data export options (CSV, PDF)
      v. Create responsive data tables with sorting/filtering
   b. Backend (Java):
      i. Create REST API endpoints for market analysis
      ii. Generate aggregate statistics from the housing dataset
      iii. Integrate with the ML model container from Task 1
      iv. Implement caching for performance optimisation

Technical Requirements:

1- Next.js Implementation:
   a. Use App Router
   b. Implement server components and client components appropriately
   c. Use React Server Components for initial data loading
   d. Implement proper data fetching strategies
   e. Create custom hooks for shared functionality

2- UI/UX Requirements:
   a. Create responsive layouts using Tailwind CSS
   b. Implement accessible UI components following WCAG guidelines
   c. Use appropriate loading states and error boundaries
   d. Create smooth transitions between pages and application states
   e. Design a cohesive UI that follows modern design principles

3- State Management and Data Flow:
   a. Implement appropriate client-side state management
   b. Handle form state effectively with validation
   c. Create efficient data fetching patterns
   d. Properly manage API communication and error states
4- Code Quality and Organisation:
   a. Structure the codebase following Next.js best practices

Technical Constraints:
- Python 3.12+
- FastAPI
- Java 21
- Spring Boot 3.4+

Deliverables:
1- Source code in github
2- Ability to show this live during interview

<u>Task 3</u>: System Architecture Design (Advanced Task)

Objective:

This is a non-coding task. Your goal is to design a scalable, resilient, and cost-effective cloud architecture for the entire "Property Portal" system. We want to understand your thought process for building production-ready systems.

Requirements

Based on the applications you built in the core tasks, create a simple architecture diagram and a brief written explanation covering the following:

1- Cloud Infrastructure
2- Scaling and Data
3- CI/CD and Monitoring

Deliverables
1- An architecture diagram (can be a simple drawing)
2- A brief document (less than one page) explaining your architectural decisions, which you will discuss during the interview.