Senior Full-stack Engineering Take-home Assignment

Assignment Details

This assignment will showcase your ability to build a full-stack system capable of handling and sending patient data to various EHR (Electronic Health Record) systems based on which EHR system is selected by the client (e.g. hospitals and clinics). This project will help us evaluate your understanding of transaction consistency, architectural choice, and practical problem-solving abilities. Use Typescript and React (or any React-based framework) for this assignment.

Problem Statement

The patient will be answering questions that the clinical team has set up for their doctor visit, and the EHR will be determined by the client, e.g. the hospital. Each question can have a different mapping in each EHR – e.g. answering a question about symptoms goes into one API endpoint for an EHR and answering a question about family history is submitted to a different API endpoint for the same given EHR. Please refer to the Example of EHR mapping to see an example of what that could look like.

Requirements

API

- Implement a method within the API that can map input data received from users to the appropriate fields in the EHR systems. This method should be flexible to handle different types of input data and different EHR systems.
- Implement a method that ensures transactions captured by your API are written to the correct users in the EHRs. This could involve validation checks, error handling, or other techniques.
- Design the API in a way that allows for the addition of more EHR integrations without significant code changes. This could involve using a modular design or implementing a standard interface for EHR integrations.
- Describe and plan out, but don't implement, an API design that allows for scalability as more users are added. This could involve efficient data structures, load balancing, or other techniques.

- Implement a system for managing the mappings for each EHR integration. This could involve storing the mappings in a database or configuration files, and providing methods for updating and retrieving these mappings.
- Implement performance measures to ensure that the system remains performant as it scales. This could involve caching, efficient data structures, or other techniques.
- Design a testing strategy for the API. This could involve any testing techniques of your choice. **Bonus**: Actually implement it if time permits.
- **Bonus:** the API supports multi-language, e.g. questions and answers can be submitted in Spanish and English

Frontend

Based on the information from the API section above, you will need to implement an internal tool to:

- Allow internal team members to be able to make changes for mappings based on EHRs.
- Implement error handling and exception management based on valid inputs.
- Plan a testing strategy for the frontend application. This could involve any testing techniques of your choice. **Bonus**: Actually implement it if time permits.
- **Bonus**: Support multi-language on the front-end as well.
- **Bonus:** Make bulk changes for patients for a given provider or hospital.

Your work will be assessed on the overall architecture of the system, the design of the API, its security aspects, readability and modularity of code, and importantly the approach taken for ensuring future readiness such as backwards compatibility, performance, and scalability assurances. Please ensure that the end result is runnable on any standard machine without needing to install complex dependencies.

What we will be evaluating and looking for:

- **Scalability:** the system should be able to efficiently manage a load of 10 million concurrent active users.
- **Backward compatibility:** while handling a large number of requests and transactional operations, the schema of the system needs to be robust, adaptive, but opinionated. The system should be designed in such a way that the addition

of new versions of the software does not obstruct the operation of the existing ones.
- **Service Resiliency:** the system needs to present an immaculate service uptime and should be designed to keep the system resilient to failure by implementing necessary fault tolerance, redundancy, and failover mechanisms.
- **Performance:** the system should be able to process a high number of requests per second and manage a large volume of data – e.g. optimizing the API for speed and resource allocation.
- **Security:** the system should have security as a top priority. Use encryption, sanitization techniques to secure patient data during transmission, storage and authorization to access these data should be strictly controlled.

Delivery of Assignment

Please upload the zip project to the Greenhouse link that is shared with you. Include a **README file** with instructions to build/run your solution, and any other deliverables like code documentation, diagrams, assumptions and explanations you think would clarify your approach and the architecture decisions you made.

Timeline

We ask that you complete and submit the assignment within a week from receiving it. We know life happens and if you need extra days to complete it, please do let us know as soon as possible. Additionally, we estimate that this project should not take more than **4 hours of your time** – please try to limit your time to within that time period.

Please feel free to reach out if you have any questions – we look forward to your project.

Example of EHR Mapping

```
{
    "Athena": {
        "patient": {
            "name": "PATIENT_IDENT_NAME",
            "gender": "GENDER_OF_PATIENT",
            "dob": "DATE_OF_BIRTH_PATIENT",
            "address": "PATIENT_LOCATION_ADDRESS",
            "phone": "TELEPHONE_NUMBER_PATIENT",
            "email": "PATIENT_EMAIL_ID",
            "emergencyContact": "EMERGENCY_CONTACT_PATIENT",
            "insuranceProvider": "INSURANCE_PROVIDER_PATIENT",
            "insurancePolicyNumber": "POLICY_NUMBER_INSURANCE_PATIENT",
            "primaryCarePhysician": "PRIMARY_CARE_DOCTOR_PATIENT",
            "allergies": "ALLERGIES_PATIENT",
            "currentMedications": "PATIENT_MEDICATIONS_CURRENT",
            "medicalHistory": "HISTORY_MEDICAL_PATIENT",
            // This is usually previously diagnosed medical issues such as abdominal pain, shortness of breath, chest pain, injuries, past surgeries, etc.
            "socialHistory": "HISTORY_SOCIAL_PATIENT", // These are familial, occupational, and recreational aspects of the patient's life that can have the potential to be clinically significant. Think of alcohol, tobacco, drugs, diet, travel, etc.
            "familyHistory": "HISTORY_FAMILY_PATIENT" // This can be things like history of high blood pressure, cancer, stroke, diabetes, rare conditions, etc, that run in the family
```

```
            }
        } "Allscripts": {
            "patient": { //note that the keys are different
            "p_name": "NAME_OF_PAT",
            "p_gender": "GENDER_PAT",
            "p_dob": "BIRTHDATE_OF_PAT",
            "p_address": "ADDRESS_PAT",
            "p_phone": "PHONE_NUMBER_PAT",
            "p_email": "EMAIL_ID_PAT",
            "p_emergencyContact": "EMERGENCY_CONTACT_PAT",
            "p_insuranceProvider": "PROVIDER_INSURANCE_PAT",
            "p_insurancePolicyNumber": "POLICY_NUM_INSURANCE_PAT",
            "p_primaryCarePhysician": "PRIMARY_CARE_DOC_PAT",
            "p_medicalHistory": "HISTORY_MEDICAL_PAT",
            "p_allergies": "ALLERGIES_PAT",
            "p_currentMedications": "CURRENT_MEDS_PAT",
            "p_socialHistory": "SOCIAL_HISTORY_PAT",
            "p_familyHistory": "FAMILY_HISTORY_PAT"
            }
        }
}
```