Project 1: Patient Questionnaire App

JAPL: Jefferson, Alan, Peter, Lazar

Problem Statement

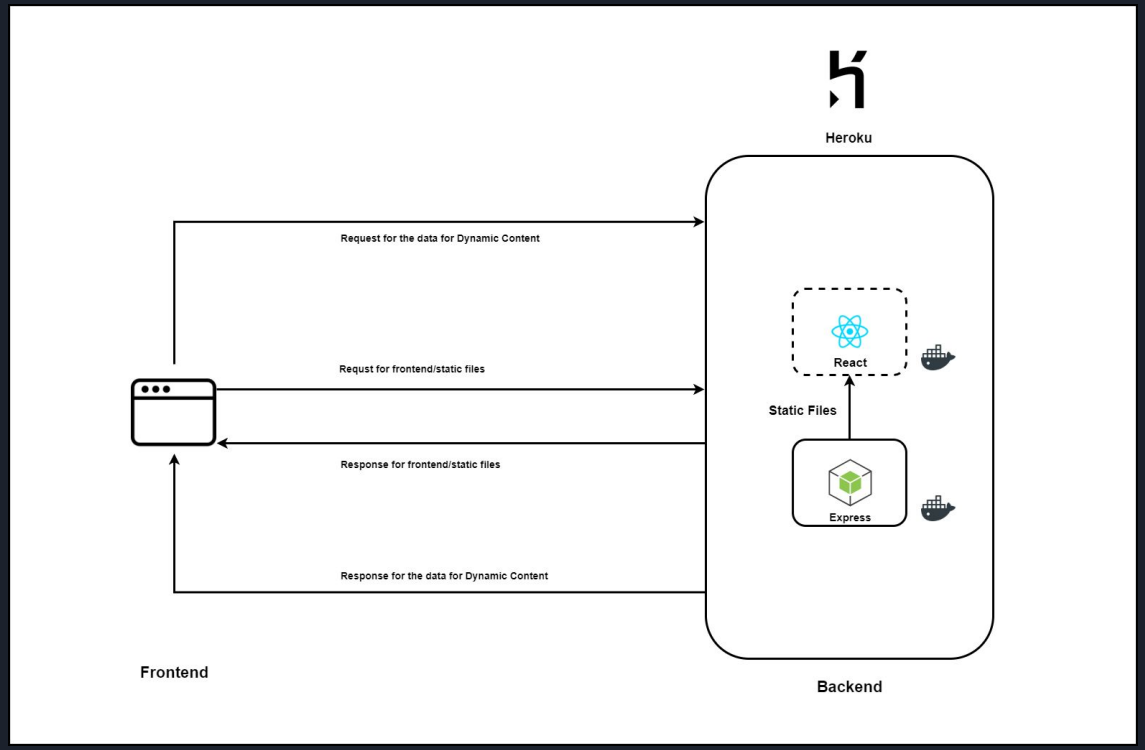
- Currently transfer of information in the medical industry is done using Fax
- FHIR is a complete contract to speed up process of exchanging information
- For example, it allows submission of questionnaires to be faster

Overview

- Build a web app capable of rendering provided XML/JSON FHIR Questionnaires
- Submit them using a standardized FHIR API to HAPI FHIR (Open Source FHIR EndPoint)

The Work

Architecture



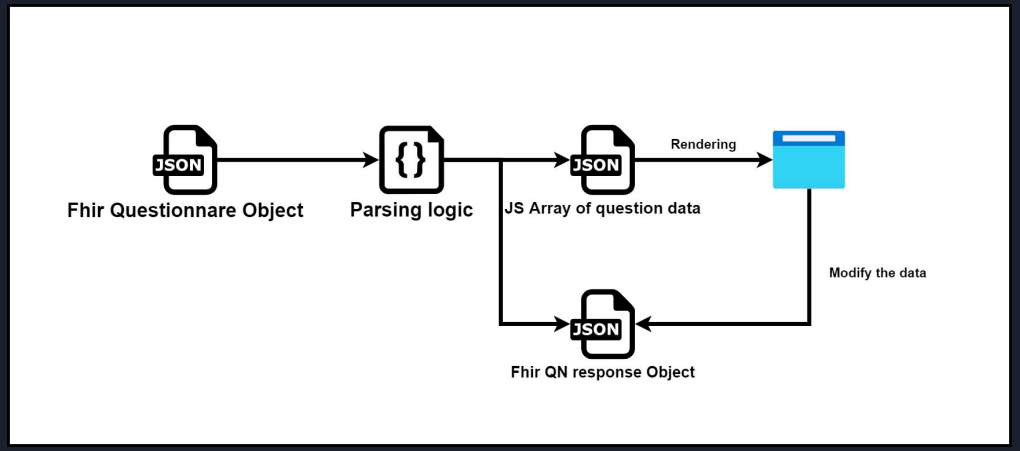
Frontend Work

Demo

Frontend Work - Tech

- Tech Stack (UI)
  - React (SPA)
  - Ant Design
  - ~~Parsing libraries (fhirformjs, questionnaire-to-survey)~~

Frontend Work - Parsing



Challenges:

- Fhir Docs (ambiguities, flexibility, LOINC)
- Lack of examples

Algorithms & Logics:

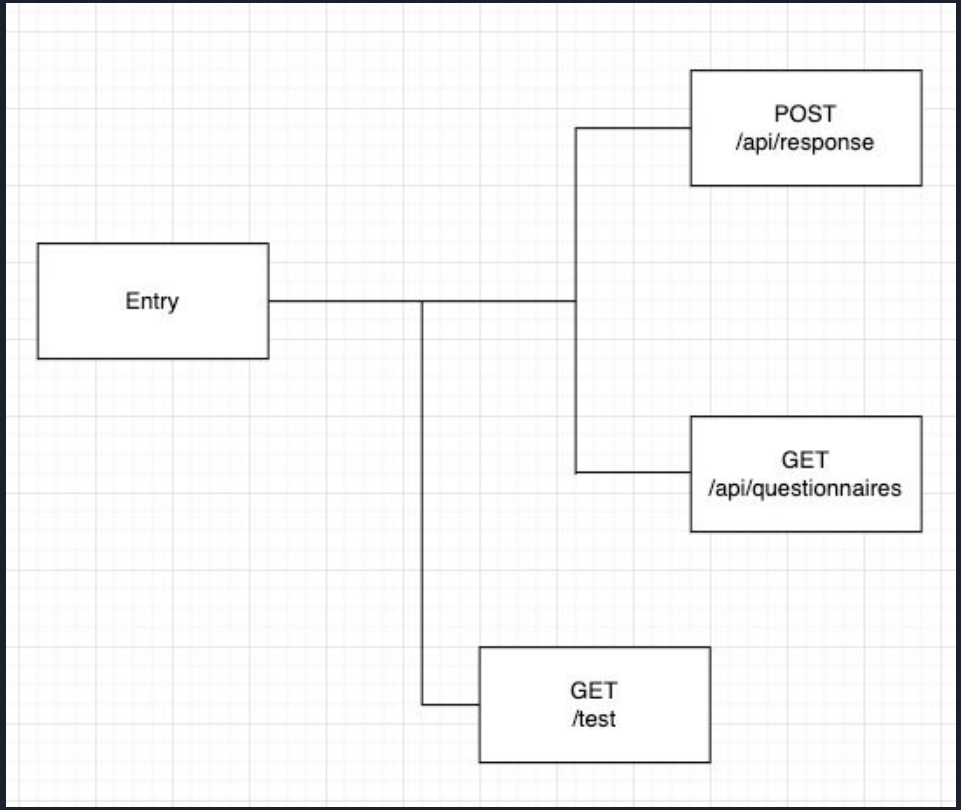
- Interpret Questionnaire
- Generate empty response JSON/Object
- Update response JSON/Object

Backend - Tech Stack

node js™ express

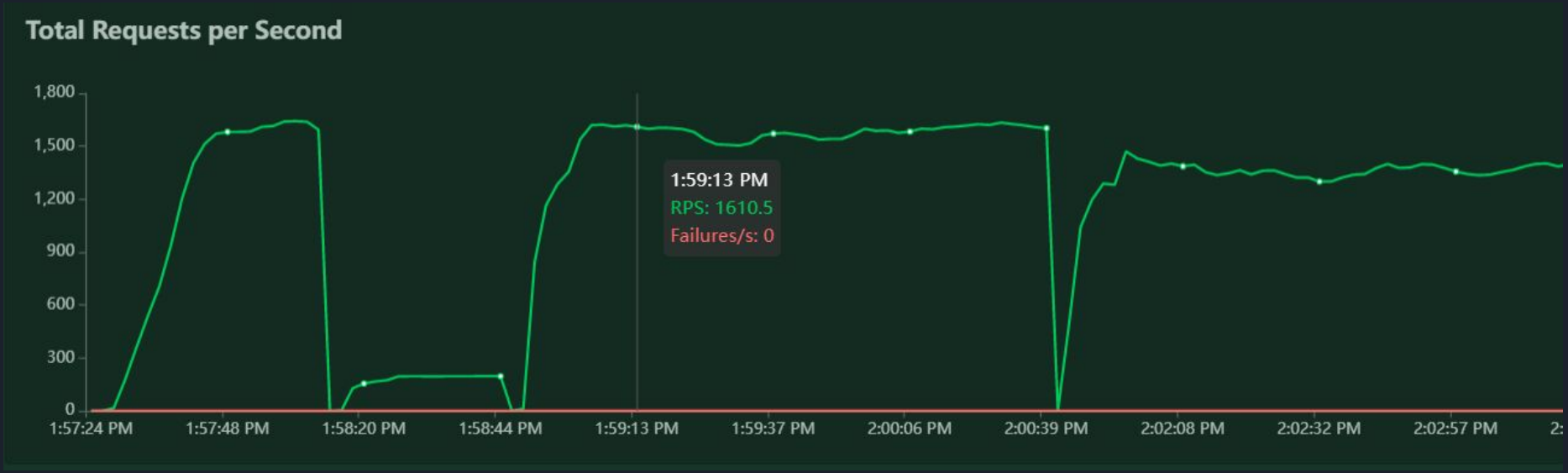
JEST

Backend - Overview



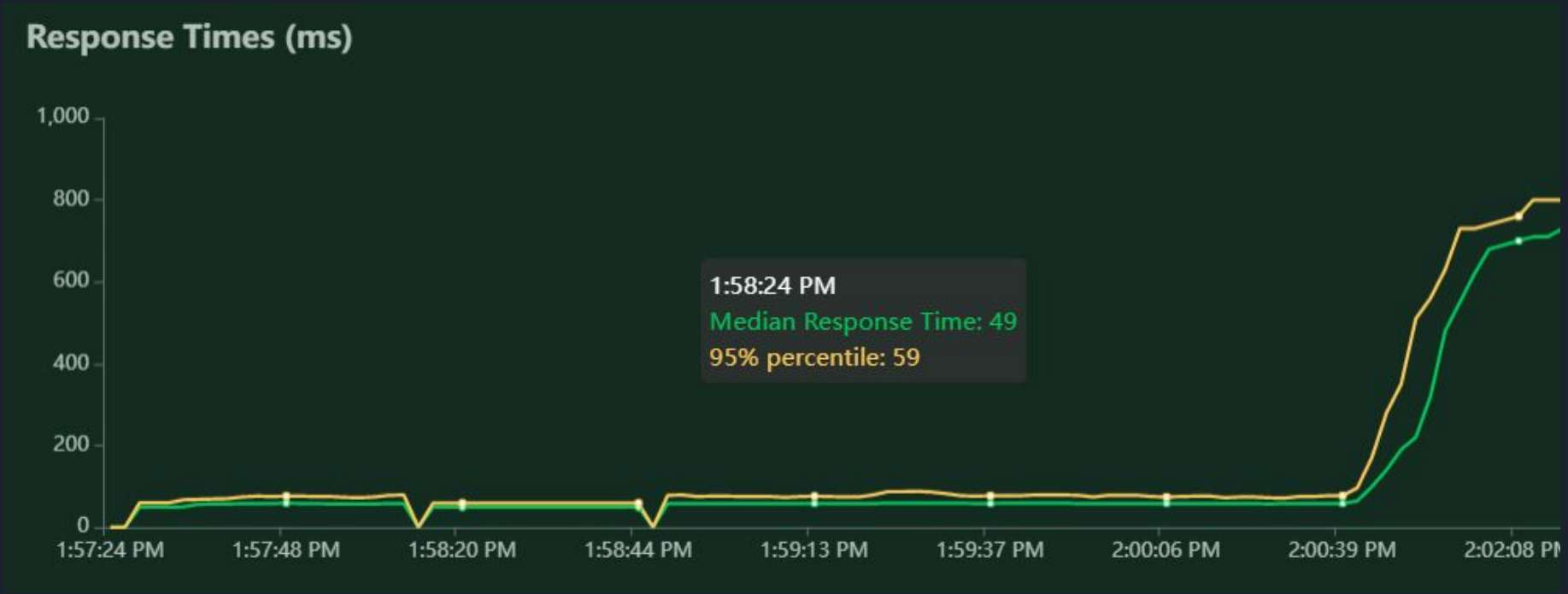
Performance - RPS
- 6 cores & 16g of memory

Throttles around 1600 RPS



Performance - Latency

P95 - 59ms P50 - 49ms



Performance - Improvements

- Serverless
  - Put backend on FaaS platform e.g. AWS Lambda
  - App becomes static and can easily scale horizontally

Decisions

Decision Making

- Team experience
  - Tech Stack
  - Task breakdown
- Industry partner
  - Acceptance and verification criteria
  - Priority
- Retrospectives
  - Adjust scopes

Adjustments

- A1 Retro:
  - Limited to specific number of questionnaires
  - Switched UI library from Material UI to Ant Design
- A2 Retro:
  - Changed approach for rendering questionnaire
  - Left out question types that aren’t used in any existing questionnaires
  - Dropped some optional features

Testing and Validation Criteria

Feature 1

Backend endpoint for fetching a JSON questionnaire

- The backend should send a proper JSON questionnaire when a request is made
- Validated by tests in the backend test suite

Feature 2

Rendering the questionnaire on the frontend

- The user will be presented with a questionnaire in the UI
- The questionnaire will have all the questions and fields present
- Fields in the questionnaire will be able to be populated and the fields will accept the correct formats and data types

- Validated by tests in the frontend test suite as well as manual processes

Feature 3

Handling and packaging the user response

- The user will be able to submit the questionnaire after completing it
- The user will see a message if not all required fields have been filled
- The backend will receive the data object

- Validated by tests in the frontend test suite as well as manual processes

Feature 4

Backend endpoint for receiving the questionnaire response and sending it to the FHIR endpoint

- The backend should receive the questionnaire response from the frontend
- The backend should send the received questionnaire response to the open source FHIR endpoint
- A successful response code should be received from the FHIR endpoint

- Validated by tests in the backend test suite