JSI Engineering Applicant Test

Introduction

This test is designed to take you through some tasks that would be typical for a backend 4Sight software developer. It should allow you to get a sense of the sort of work that you would be involved with at JSI, and will help us assess your suitability for our team.

You have 3 hours to complete this test. The test results must be emailed to JSI no later than 3 hours after receiving it. Please leave yourself sufficient time to document and package your results. The results package may contain code files and text documents. Please ensure it is clear which items relate to which question.

Deliverable Overview:

1. A system that can process the two API requests via HTTP
2. Appropriate tests (unit, integration, etc)
3. A quick explanation of:
    a. Choice of data format. Include one example of a request and response.
    b. Optional: what authentication mechanism was chosen, and why.

The purpose of this exercise is for you to demonstrate your current skills, not pick up a new library/framework/workflow/IDE. We know good developers can adapt, so for this exercise please use your preferred stack to deliver working and properly-tested code.

Task 1 – Parsing

The first task is to consume and parse the three data files given in the resources folder, and store that data in memory. Each file represents a different data type where the filename is the *type*. New data types may be added in the future. Please model and test accordingly. Using a real database is not required.

Deliverable:

1. Functional and tested code that can parse and store data read from files.
2. The parsing should occur on startup of the Web API so it can be consumed later.

Follow-up question:

Briefly describe how you would approach designing and implementing a database to store your model.

Task 2 – Serving

The second task is to serve your parsed data with two different endpoints:

- ***GetTypes*** endpoint should be a GET request that returns each of the parsed data types from the resource files.
- ***TimeFilter*** endpoint should be a POST request, it should accept a date range filter, and a list of data types. See examples below:

Deliverable:

1. Fully functional Web API.
2. It should integrate the parsing component from task one.

Optional:

Describe an authentication strategy that could be added to the ***TimeFilter*** endpoint

Example Response *GetTypes* endpoint:

```
[
  "ChatMessage",
  "Email",
  "Sms"
]
```

Example Request *TimeFilter* endpoint:

```
{
   "DataTypes":[
      "ChatMessage",
      "Email"
   ],
   "FromTime":"2021-01-01T09:00",
   "ToTime":"2021-01-27T10:00"
}
```

See next page for more examples.

Example Response *TimeFilter* endpoint:


```
[
   {
      "application":"Facebook",
      "from":"john@yahoo.com",
      "to":"Susan Smith",
      "text":"Hi did you call me earlier?\r",
      "communicationType":"ChatMessage",
      "id":0,
      "time":"2021-01-01T09:00:00"
   },
   {
      "application":"WhatsApp",
      "from":"6135557777",
      "to":"6135556666",
      "text":"Let's go see a movie in the weekend?!",
      "communicationType":"ChatMessage",
      "id":1,
      "time":"2021-01-01T09:01:00"
   },
   {
      "from":"alice@email.com",
      "subject":"Hello",
      "to":[
         "bob@email.com",
         "charles@email.com"
      ],
      "cc":[
         ""
      ],
      "bcc":[
         ""
      ],
      "body":"How are you guys?",
      "communicationType":"Email",
      "id":2,
      "time":"2021-01-01T11:00:00"
   },
   {
      "from":"bob@email.com",
      "subject":"Hey",
      "to":[
         "alice@email.com",
         "charles@email.com"
```


Task 3 – Continuous Integration and continuous Delivery/Deployment

Deliverable:

Briefly describe a strategy for creating a CI-CD pipeline for the API created in earlier tasks.