ajmera infotech inc.

Asp.net Core Backend Coding Exercise

The purpose of this exercise is to provide Ajmera InfoTech with a candidate's familiarity with coding a .net core API service. The candidate is welcome to cap their effort to 6 hours and if the exercise is not complete by that point, we will evaluate the partially complete effort. If the candidate wishes to spend more time completing the exercise, that is also OK.

API Service Description

The API service should provide restful endpoints to create and get a book entity.

Post Entity Schema

{

"name": "Rich Dad Poor Dad",

"authorName": "Robert Kiyosaki",

}

Get book from Id

The GET endpoint should allow for retrieval by id. The GET endpoint does not need to allow for retrieval by any other properties. Though we asked for the create endpoint.

{

"id": "54c4e684-0a6a-449d-b445-61ddd12ffd3d",

```
“name”: “Rich Dad Poor Dad”,
“authorName”: “Robert Kiyosaki”
}
```

Get books

The GET endpoint should allow for retrieval of all books.

```
[{
“id”: “54c4e684-0a6a-449d-b445-61ddd12ffd3d”,
“name”: “Rich Dad Poor Dad”,
“authorName”: “Robert Kiyosaki”
}]
```

Also note that I’m assuming the id is generated as part of the POST operation.

Response codes

The candidate should brainstorm and code appropriate HTTP response codes for non-success.

Data Store

The data store should be MSSQL.

Integration Tests

The Visual Studio solution should include an integration tests project with a reasonable number of tests to exercise the endpoints and prove their correctness, including some scenarios where non-success responses are expected. The candidate is welcome to provide a unit test project though the integration tests are what we'll focus on. For the integration tests, please make actual HTTP calls from your tests into your service.

How to submit the code

Please host the code in a free code repo GitHub make it publicly accessible, and provide a link to the resulting repo.

Consider using some of our favorite libraries:

Purpose	**Library**
Dependency Injection Container	Built-in Microsoft DI container
Logging	Candidate's discretion
Object mapping / conversion	AutoMapper
Validation	Candidate's discretion
Endpoint documentation	Swagger
ORM	Entity framework
Design patterns	Service and repository

Bonus Points

The following are not mandatory. Feel free to do these for "bonus points".

- Dockerize the solution
- Health check endpoint that validates SQL connectivity
- Swagger documentation of the endpoints
- Provide an endpoint to mutate/modify the user record