Technical Assessment

Library API

In this test, you are expected to write a small web application to manage a list of Books. Each book has a name, ISBN, and an author. The test consists of two parts, a RESTful API as the backend and the Javascript based frontend application.

Backend

Use the following structure to model the data.

```
class Author(Model):
    first_name = models.TextField()
    last_name = models.TextField()


class Book(Model):
    name = models.TextField()
    isbn = models.TextField()
    author = models.ForeignKey(Author)
```

Implement the following API endpoints:

- GET /books
  - Returns a list of books in the database in JSON format.
  - Should be able to paginate the response.
- GET /book/{{id}}/
  - Returns a detailed view of the specified book id.
  - Nest author details in JSON format.
- GET /authors
  - Returns a list of authors in the database in JSON format.
- GET /author/{{id}}
  - Returns a detailed view of the specified author id.
- POST /author
  - Creates a new author with the specified details.
  - Expects a JSON body.
- POST /book
  - Creates a new book with the specified details.
  - Expects a JSON body.

Optional: You can go a step further by implementing API endpoints to update existing records if you like.

eg:

- PUT /author/{{id}}
    - Updates an existing author.
    - Expects a JSON body.
- PUT /book/{{id}}
    - Updates an existing book.
    - Expects a JSON body.

You are recommended to use Node.js along with Express to implement your backend and API layer.

Frontend

Implement a small frontend application to consume the API you developed above.

- The frontend should be able to show a list of names of the books available in the database.
- Upon clicking the name of a book on the list, the user should be navigated to a more detailed view of the selected book, where they are presented with the ISBN and the author details.
- You should also implement two forms where the user is able to create/update authors and books (using the POST and PUT endpoints)
- You are recommended to use Angular to create the frontend, but you are free to use a different Javascript framework.

Tech Stack:

- Angular
- Any CSS framework if required
- RxJS or Redux

Tips

- Come up with your own UI design, but let's make sure it is simple and user friendly.
- The project structure is up to you to decide.
- You are recommended to use git commits in a logical manner to demonstrate the development progress. We prefer the semantic commit message format.
- Writing tests and adhering to development standards/conventions will let you gain extra points :)

- When you are developing the application, focus on the following points:
    - Simplicity
    - Maintainability
    - Testability
    - Performance

This assessment is targeted to be similar to a day's work at YTA. Once completed, share your work in a public git repository.

Happy Coding!