Programming exercises

Thank you for participating in this exercise. This programming exercise is just a way that we evaluate candidates and we ask all our candidates to go through some form of this. Do ask us for clarifications, bounce ideas off us (we'll help where we can) and feel free to utilize all the resources we have made available. We hope you have fun doing these problems.

Instructions

- Your code should use good programming practices and conventions including consistent indentation, consistent use of variable/function/class/method naming conventions and comments.
- You are free to use any internet resources that you like. If you copy code from the internet, we will require that you can explain the code completely and also the license under which it was copied.

Question 1

The Gutendex API is a hosted instance of the open source project, which in itself is an extension of data available from gutenberg.org, a repository of freely available books. It contains a single endpoint, with a set of query parameters that allow for filtering the books in several manners.

The hosted instance is available on http://skunkworks.ignitesol.com:8000/. The API specification is provided on the last page of this document.

Description

The app has 2 distinct pages, the designs for which are provided here.

The first page will have a title of the app and a list of buttons labelled by the category/genre of books available.

Clicking on the buttons should transition the user to the next page, which would display the books matching the selected category. The books SHOULD be displayed in an infinite scrolled list (fetching more books as the user scrolls through the list). If the user enters any text in the search bar, the API MUST be invoked again to search for **all the books** whose **`title`** or **`author`** match the given input text, while maintaining the currently selected category/genre filter.

The API response contains links for books in different formats. If the user taps on any book the app **must** open the **web browser** pointing to the book in one of the following formats, the format higher in the list must be preferred, falling back to the next available format (if given format is not available)

- HTML
- PDF
- TXT

If none of the above **formats** are available, the app **must** display an alert box with an error message “No viewable version available”.

Caveats

1. Zip files are NOT viewable files.
   (**Hint:** This issue can be easily worked around, bonus points for figuring it out)
2. API response should only contain the books with covers.
   (Hint: ONLY query for books that have a **mime_type** with images)

Example

If the user taps **Fiction** on the first page, the next page will display a scrollable list of all the books of category/genre **Fiction**.

Thereafter, if the user searches "**Vampire**" in the search box, the list MUST be filtered via the API with ALL books that are **Fiction** (**Subjects and Bookshelves**) AS WELL AS **Vampire** (**Title and Authors**) based books.

Upon tapping on the book, the **web browser** opens with either (in order of priority) the HTML version or the PDF version or the TXT version of the book (or gives an error alert).

We expect this exercise to completed within 3 hours

Below is the API as taken from the Github link. It is a working and tested setup.

Any provided designs MUST be followed.

API

Base API URL: http://skunkworks.ignitesol.com:8000/

Lists of Books

Lists of book information in the database are queried using the API at /books (e.g. http://skunkworks.ignitesol.com:8000/books). Book data will be returned in the JSON format

```
{
  "count": <number>,
  "next": <string or null>,
  "previous": <string or null>,
  "results": <array of Books>
}
```

where `results` is an array of 0-32 book objects, `next` and `previous` are URLs to the next and previous pages of results, and `count` in the total number of books for the query on all pages combined.

Books are ordered by popularity, determined by their numbers of downloads from Project Gutenberg. The following query parameters can be passed for filtering the books

`ids`

Use this to list books with Project Gutenberg ID numbers in a given list of numbers. They must be comma-separated positive integers. For example, /books?ids=11,12,13 gives books with ID numbers 11, 12, and 13.

`languages`

Use this to find books in any of a list of languages. They must be comma-separated, two-character language codes. For example, /books?languages=en gives books in English, and /books?languages=fr,fi gives books in either French or Finnish or both.

`mime_type`

Use this to find books with a given MIME type. Gutendex gives every book with a MIME type *starting with* the value. For example, /books?mime_type=text%2F gives books with types `text/html`, `text/plain; charset=us-ascii`, etc.; and/books?mime_type=text%2Fhtml gives books with types `text/html`, `text/html; charset=utf-8`, etc.

`search`

Use this to search author names and book titles with given words. They must be separated by a space (i.e. %20 in URL-encoded format) and are case-insensitive. For example, /books?search=dickens%20great includes *Great Expectations* by Charles Dickens.

`topic`

Use this to search for a case-insensitive key-phrase in books' bookshelves or subjects. For example, `/books?topic=children` gives books on the "Children's Literature" bookshelf, with the subject "Sick children -- Fiction", and so on.