Table of Contents

Acknowledgment

A project is a golden opportunity for learning and self-development. Many people deserve our cordial thanks for their help in completing this project. First and foremost, we would like to thank our honorable Supervisor **Dr. Mst. Jannatul Ferdous**, Professor, Dept. of CSE, JKKNIU who is so dedicated to helping us in our times of need. We are honored to make this project under her guidance which provides us with her valuable time and knowledge, motivating thought, and encouragement.

We wish to express thanks to **Professor Dr. Md. Sujan Ali**, Head, Department of Computer Science & Engineering, Jatiya Kabi Kazi Nazrul Islam University for providing us with lab opportunities and lab materials related to our project work.

Finally, we express our thanks to the Department of Computer Science and Engineering for allowing us to study here and for supporting us greatly.

Abstract

Today the amount of information on the internet grows very rapidly and people need some instruments to find and access appropriate information. One such tool is called a recommendation system .Recommendation systems help to navigate quickly and receive necessary information. Generally, they are used in Internet shops to increase the profit. This paper proposes a quick and intuitive book recommendation system that helps readers find appropriate books to read next. The overall architecture is presented with its detailed description along with all the technical bits of design and creation of it.

Chapter-1

INTRODUCTION

1.1 Introduction

There are millions of books in the entire world and people need some instructions to find the appropriate book. Making decisions and scrolling for the right book from millions of books can be hard and a complete waste of time. Especially for static websites like gutenberg.org. It is created using simple systems and thus it does not give very well recommendations to its users.

Gutenberg.org has over 70,000+ books in different categories, languages, and types. All of those books are in the public domain. So it doesn't get as much care as most books available today do. Plus adding in the constraint that they do not have any account system that tracks and collects data about the users' preference of the books. They are crippled with the current system that can only suggest some very well-known classics that are constantly downloaded regularly by people. Although this system is good for new users. It is a hindrance to the awesome books that are not as well-known as these classics. They are not read as much as by the new users so these books are always at the bottom of the choice list in the popular section. You can of course search for the book but searching only gets you so far as the website cannot show you a lot of the search results at once. This makes it harder for a bookworm to get to know any book that is already available here but they cannot even read the name of it. Our project is a small attempt to solve this problem.

1.2 Objectives

The major objectives of this game project are:

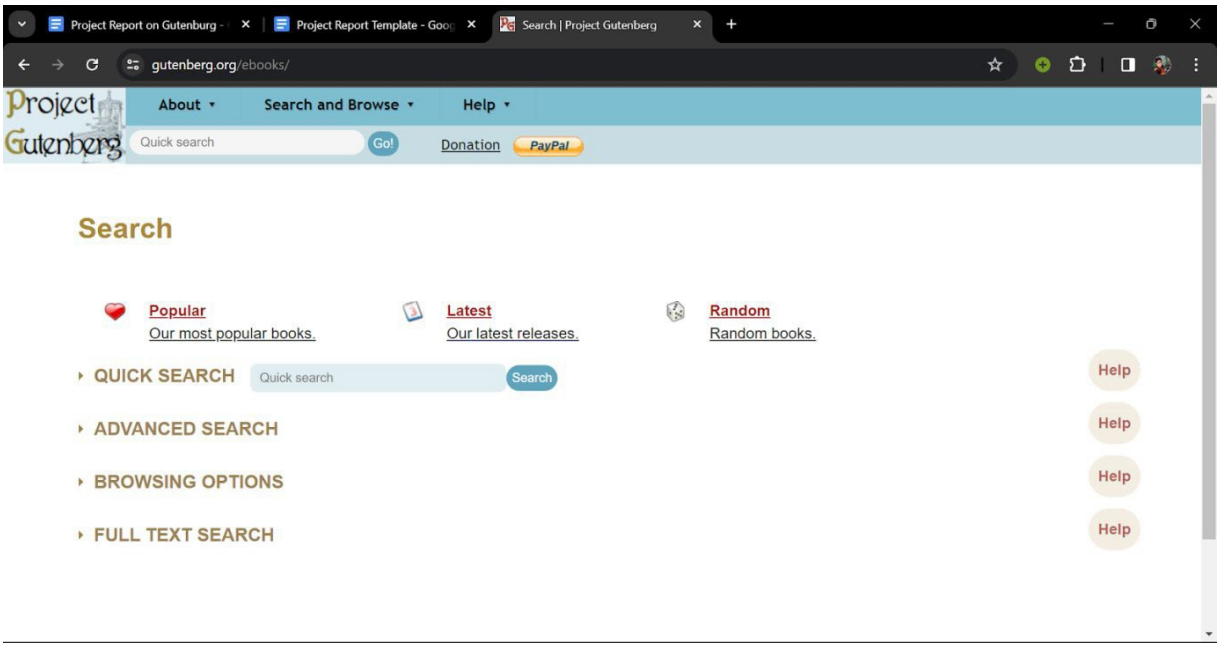
1. Develop a Python-based Book Recommendation System to elevate user interaction within the Gutenberg Library.
2. To create an intuitive and accessible interface for users to seamlessly explore recommended books.

Chapter-2

BACKGROUND STUDY

2.1 Current State

Currently, the Gutenberg Library has a small way of suggesting to users some books by popularity, searching, and for everyone who has read a lot more books, there is the latest book section for books that are recently added to the public domain and added to the Gutenberg library.



Unfortunately, when we want to read a book the genre of the book is dependent on our mood and current interest. There are a lot of awesome books in the Gutenberg library and it is a true shame that it cannot have a proper recommendation system such as Google Books or Amazon's Kindle. The reason is that Project Gutenberg is a nonprofit organization that is held by donations and volunteers. It is impossible to assume that a volunteer will know how to work with a difficult tech stack and even know how to work with basic HTML and book formatting knowledge. So they are stagnant in their development and they cannot give any suggestion system targeted to a reader.

2.2 Our Solution

By using their system we have found that most books that are both old enough and being preserved are really good and almost any book that is in our favorite genre is really good. So, it doesn't need a very precise suggestion system. It only needs a system that is recommended by the user's choice. As long as the suggested book has already not read by the reader then the recommended book will have a higher chance of being a good recommendation enough that the reader may want to read it.

Thankfully Gutenberg.org provides its total book catalog in a comma-separated value(CSV) file in unicode encoding. The CSV file contains the text(A type of book ID in their collection), the title of the book, when it was published, what genre of the book, what other format the book is the public domain material as Gutenberg also hosts other public domain materials such as pictures, music, movies, etc.

Thus we can easily filter the books as the type of that id is set to "Text" and considering Gutenberg has mostly English books we can filter the books and get a catalog of books. Then, we take users' suggestions and filter the catalog with the users' desired catalog and suggest some books of that certain genre and the user can have good suggestions as long as we can make sure the catalog doesn't contain any of the books the user has already read.

2.3 Functional Requirement

The functional requirement for making a book recommendation system such as "**Book Recommendation System for Public Domain Books in Gutenberg Library**" requires the use of many software technologies. Which includes a programming language, in this case, Python (Python is a great choice for software development due to its simple syntax, powerful libraries, and ease of integration with other languages and tools. Python is also highly extensible, meaning that it can be used to write custom scripts and modules for software. Additionally, Python has a large and active community of developers, which makes it easy to find help and resources when developing software. Finally, Python is open source, meaning that it is free to use and modify), a code editor (e.g. Vs Code).

Chapter-3

Methodology

3.1 Application Flow

The diagram outlines a process for recommending books to users. It starts with the user searching for a book by title, author, or category. The system then recommends random books based on the search criteria. The user can then add these books to their wishlist or readlist, open a link to read the book or ignore the recommendation.

Overall, the methodology is relatively simple and straightforward. It focuses on providing users with a variety of book options based on their initial search query. Users can then choose which books they want to learn more about or add to their reading lists.

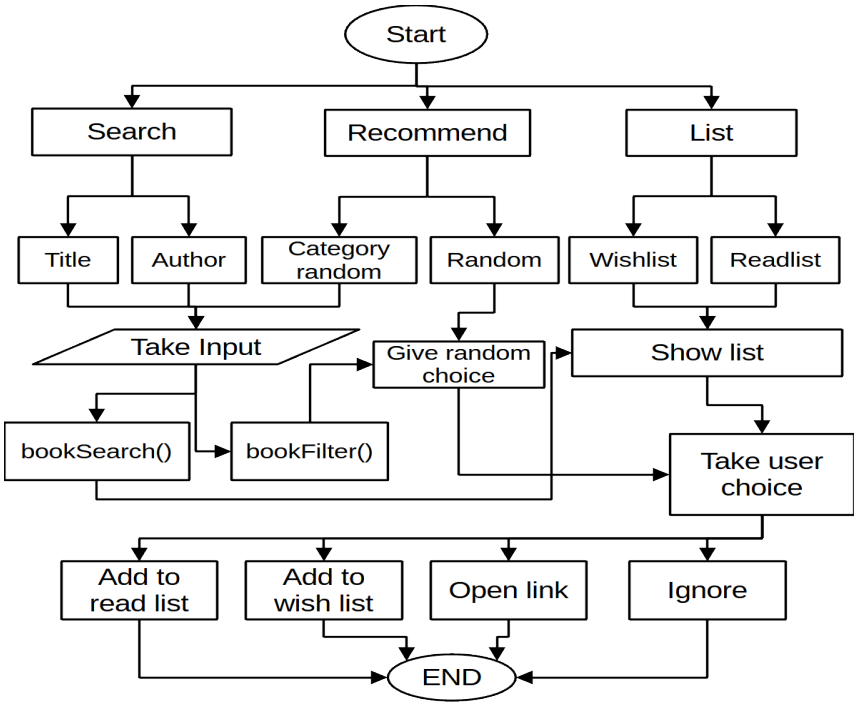


Fig : Application Diagram

3.2 Functions and Modules

i. Preprocessing

The program runs on the basis that the files that store the Gutenberg book catalog and users' wishlist and readlist are already read and stored on the memory as variables so we need to preprocess the data so that those data are in their corresponding variables. So, we open the read books of the user, and using the **DictReader()** method in the CSV module in Python we read all the lines and append them to an empty list.
Following that we do the same for the wishlist as well. But for the project Gutenberg catalog, we cannot just read it. We both read and filtered the books when reading the catalog. We include all the books that are of type as text and we exclude all the books that are already read. So at the end of the preprocessing, we get the book catalog that is not read by the user, the book catalog that is read by the user, and the book catalog which are not read by the user but the user wants to read them.

ii. User-defined functions

- ✓ **bookfilter()** function takes two parameter inputs. A book list and a string in the variable category. It first creates an empty list and then completes a search in the book list to check if the category is a substring of the string that contains the category and the subcategory as a string in the book list. In the case of the Gutenberg catalog when read using the DictReader() method if the book data is considered as i then the category is i["Subjects"]. We temporarily lowercase both the strings so there is no way a book is not considered due to the difference in capitalization.

- ✓ **bookSearch()** function takes two parameter inputs. One is what to search and another is where to search. The book search function searches through the filtered Gutenberg books catalog. The first parameter contains a string that contains what to search in these books and the second contains one of two strings. The default is to search through the book title. And the other is to search through the author. Although it sounds the same as the bookfilter the the difference is that it crates a priority list of the books of how many substring matches there are. The function first takes the text and splits the string into a list of smaller strings by words. If blank space creates a substring and thus makes a list of substrings. Then it declares a dictionary and puts the key of the dictionary as the number from 0 up to the number of the size of that said list of the string. Then we search through the book catalog and set up a counter for each book where we count up if the substring exists in the title of the book or the author name of the book depending on the list second parameter input of the user and this each book get a value from 0 to that number of the size of the substring list created by the split(). We then negate that value from the maximum possible of the number and using the number we have

got we append the book on the list corresponding to the list in the dictionary value. After searching through the entire book catalog we can have a priority dictionary where we have the highest substring match book on the 0 assigned list in the dictionary and then we convert that dictionary to a list. Of course, we can't just give all the value to the user as this makes no sense for the user searching for a book with 10 words in the title to get a list with 1000 books so we make a list that includes the 100 books of the highest priority that contain all the substring in them 100/2 for that contains one less 100/3 that contains 2 less than all the substring and so on and so forth. The design decision of the function came from the fact that a user may make a mistake while searching for the name of the book he is looking for but it is highly unlikely for the user to make a mistake in all of the words.

- ✓ **addToWishlist()** function adds a book to the wishlist of the user. In this case, the function first checks if the book is already in the wishlist or not as the wishlist books are still recommended and show up in the search list. Then first the function appends the book to the wishlist cache variable. Then the function opens the wishlist file using with open method and writes details of the book using the DictWriter() method from the CSV module.

- ✓ **deleteFromWishList()** function deletes a book from the wishlist. It does nothing if the book is not on the wishlist. But if it is in the wishlist then first it takes the book and removes it from the wishlist and then opens the wishlist file and rewrites all the books that were in the list except the one that needs to be deleted.

- ✓ **addToReadlist()** function takes one book data dictionary as input and it first adds the book to the read list data deletes the book from the book catalog cache and then deletes the book from the wishlist by calling deleteFromWishlist() and then opens up the my_book file on append mode and writes the book data at the end of the file.

iii. Modules

There are multiple modules used in this project. They are-

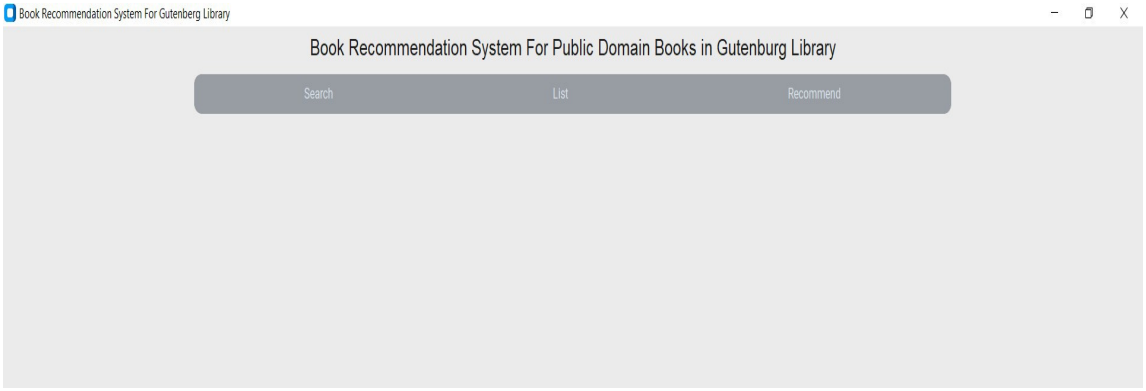
1. **CSV module**: This module makes it easier to parse CSV files. This is a preinstalled module that comes with the Python install.
2. **Random module**: This is also an internal module of Python. This makes it easier to randomize something.
3. **Tkinter module**: This is a GUI framework that is preinstalled with Python. However, it requires a very high learning curve and an extremely high amount of knowledge to make a good user interface. So it is only used as a supplement to the custom tkinter library in this project.
4. **Webbrowser module**: This is used to open a link in a tab in the default browser of the computer. This is also an internal module pre-installed with Python.
5. **Custom Tkinter module**: This is the only module that is used in this project that is not preinstalled with Python install. This uses all the tkinter modules and uses its own styling to make the widgets better looking without needing to write very long and very specific commands that may break with the smallest change.

Chapter-4

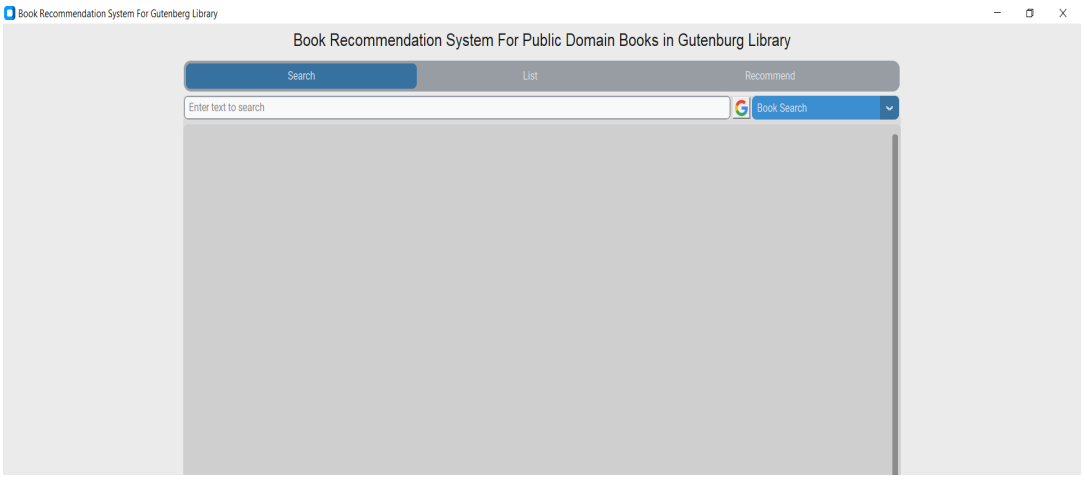
Design and Implementation

4.1 User Interface

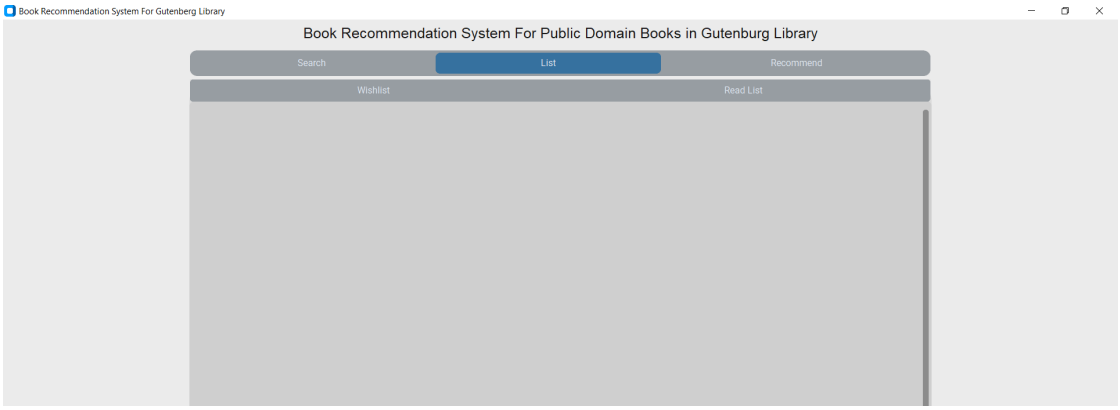
Category Selection Menu: The project starts with an opening menu where the user can select 1 out of 3 categories to solve his problem.



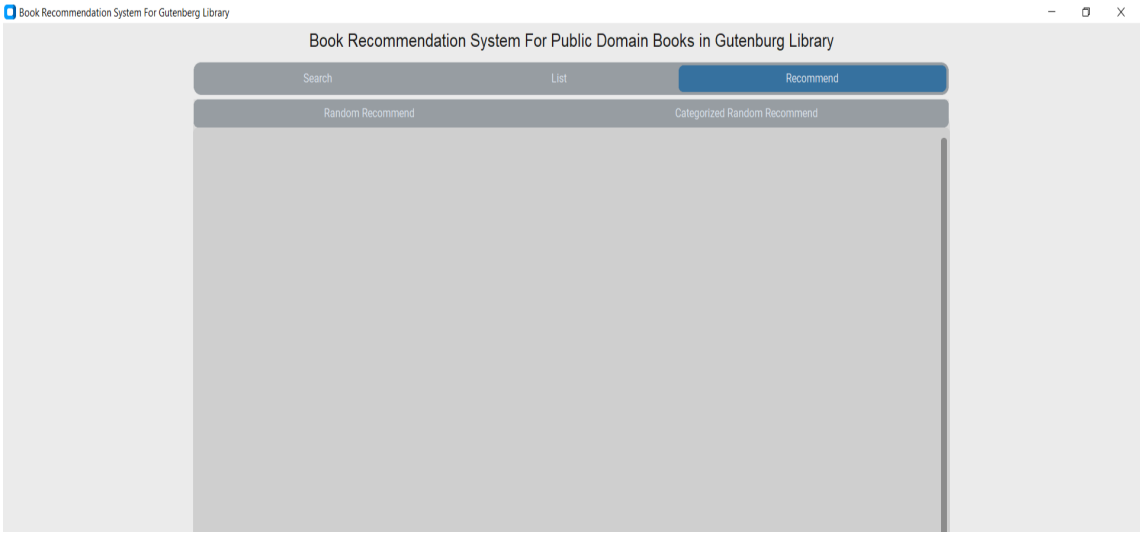
Search Category: Here user can search his desired book using the book name or author name. If the book is not listed in the CSV file then he can search it on Google using the google button beside the search bar.



List Category: The list category is divided into two subcategories. One is a Wishlist where the user stores the books he wants to read. And the other is Read List where already read books are stored.



Recommend Category: The recommend category is divided into two subcategories. One is Random Recommend and the other is Categorized Recommend. If we click Random Recommend then a list of different types of books shows. If we click Categorized Random Recommend then the user can access his preferable genre book using genre name in search bar.



4.2 Development Tools and Environments

Vs Code: IDE for code writing.

Chapter-5

Conclusion

5.1 Limitations of this Project

- There are a lot of concurrent limitations that are plaguing the app of its value
- The project Gutenberg only has a limited amount of books
- The code uses a linear complete search to look for and filter through books causing the application to be slow
- Customtkinter library when showing too many widgets requires more memorythan desired
- We do not have many color schemes due to the appearance theme limitation
- It shows an error when we try to use category recommend using text is not a type of book category

5.2 Future Scope

- We would like to add an automatic update system so the user can automatically update the library catalog from gutenberg.org
- We would like to fix the category recommend error
- We would like to increase the quality of the complete search that is used here
- We would like to optimize the application so that more devices with different operating systems and weaker configurations can use this application
- We would like to add some sort of statistical system so that the random suggestion is similar to the genre of the user's previously read books

5.3 Conclusion

In conclusion, the implemented Book Recommendation System project provides a comprehensive and user-friendly platform for navigating the extensive catalog of public domain books within Project Gutenberg Library. The tkinter-based graphical user interface offers intuitive functionalities, allowing users to seamlessly search for books, manage reading lists, and receive personalized recommendations. The modular design, global variables, and external module integration contribute to the project's organizational structure and aesthetic appeal. While the code achieves its primary objectives, future iterations could benefit from enhanced error handling and more thorough documentation. The project successfully addresses the fundamental goal of creating an engaging and efficient system for users to explore, organize, and discover books within the Project Gutenberg Library, making it a valuable. contribution to the realm of digital library applications

Reference

- **Python Documentation**: https://docs.python.org/3.11/
- **Tkinter Documentation**: https://docs.python.org/3/library/tkinter.html
- **Customtkinter Documentation**: https://customtkinter.tomschimansky.com/documentation/