Faculty: Sana Shaikh **Lab Manual - BDA** **2023-2024**

Student Name: Alston Fernandes **Experiment No: 4** **Batch: A**
Roll No.: 19

Topic:	Implement Bloom Filter using any programming language.
Prerequisite:	- Familiarity with the programming language - Basic concept of Bloom Filter
Mapping With COs:	CSL702.5
Objective:	Implement and apply Bloom filters for any one appropriate real world application.
Outcomes:	Students will be able to understand the concept of Bloom filter and its usage and also be able to implement it for any real-world problem.
Instructions:	This experiment is a compulsory experiment. All the students are required to perform this experiment in a group. [same group as for Mini Project]
Deliverables:	**Submission on Moodle:** - **Explain one Real World Filter.** A real-world example of a filter-like library where you can add and search for books is a digital library catalogue or a library management system. These systems are commonly used by libraries and educational institutions to keep track of their book collections and provide users with the ability to find and access books. Here's an explanation of how such a system works: **Digital Library Catalogue or Library Management System** 1. **Book Database:** The system maintains a database that stores information about the books in the library's collection. Each book record typically includes details such as the book's title, author, publication date, ISBN, genre, availability status, and more. 2. **Add Books:** Librarians or authorized personnel can add new books to the catalogue. They input the book's information into the system, and the data is stored in the database. Some systems might also support the automatic addition of books using barcode scanning or ISBN lookup. 3. **Search Functionality:** Users, such as library patrons or staff, can search for books in the catalogue. They can search by various criteria, including title, author,

keyword, ISBN, genre, and more. The system uses search algorithms to quickly locate relevant book records based on the user's query.

4. **Filters and Sorting:** Users can often apply filters and sorting options to refine their search results. For example, they can filter books by genre, availability, or publication date. Sorting options allow users to order the results alphabetically, by relevance, or by other criteria.

5. **Availability Status:** The system keeps track of the availability status of each book. It indicates whether a book is available for borrowing, checked out, on hold, or missing. Users can see this status when searching for books.

6. **User Accounts:** Many library systems require users to create accounts. Registered users can have additional features such as reserving books, viewing their borrowing history, and extending loan periods.

7**. Reservations and Checkouts:** Users can reserve books that are currently checked out by others. When a reserved book becomes available, the system notifies the user, and they can check it out. Users can also check out books directly from the catalogue.

A digital library catalogue or library management system is an example of a real-world filter-like library where users can add and search for books. It serves as a centralized repository of information about a library's book collection and provides features for both library staff and patrons to efficiently manage and access books.

- **INPUT:**

- Librarians can add new books to the catalogue. They input the book's Name as information.
- Librarians can search for books in the catalogue.

OUTPUT:

- Status of Book Whether it's present or not.

- **Bloom Filter CODE:** (language: Python)

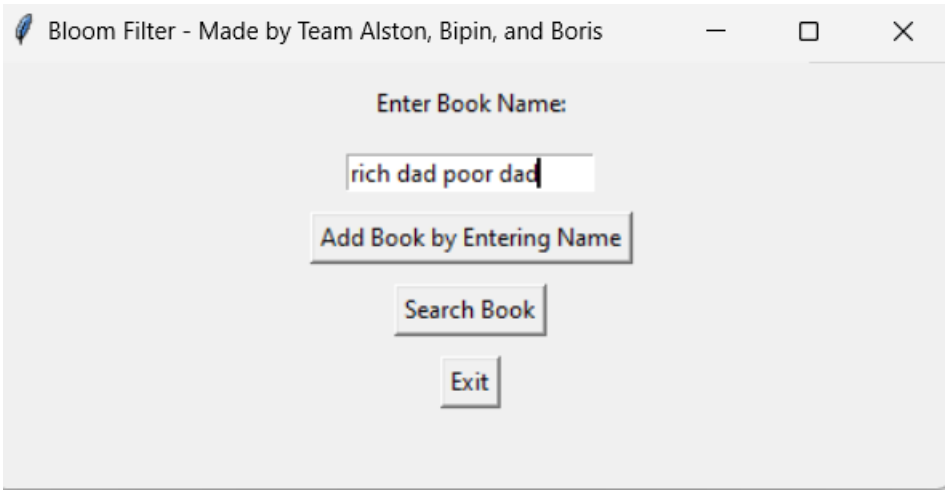
```
import hashlib
import tkinter as tk
from tkinter import messagebox

class BloomFilter:
    def __init__(self, size, hash_functions):
        self.size = size
        self.bit_array = [0] * size
        self.hash_functions = hash_functions
```

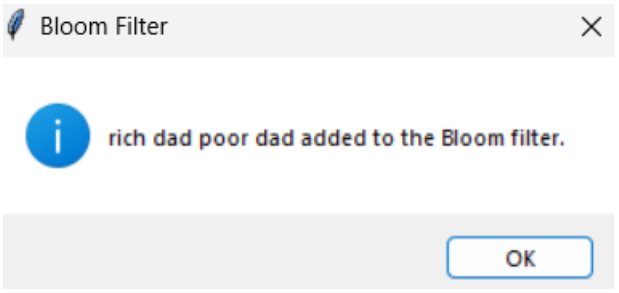
```
    def add(self, element):
        for hash_func in self.hash_functions:
            index = hash_func(element) % self.size
            self.bit_array[index] = 1

    def contains(self, element):
        for hash_func in self.hash_functions:
            index = hash_func(element) % self.size
            if self.bit_array[index] == 0:
                return False
        return True

def add_element():
    element = entry.get()
    if element:
        bloom_filter.add(element)
        entry.delete(0, tk.END)
        messagebox.showinfo("Bloom Filter", f"{element}
added to the Bloom filter.")

def search_element():
    element = entry.get()
    if element:
        if bloom_filter.contains(element):
            messagebox.showinfo("Bloom Filter", f"{element}
may be in the set.")
        else:
            messagebox.showinfo("Bloom Filter", f"{element}
is definitely not in the set.")
        entry.delete(0, tk.END)

def exit_program():
    window.destroy()

size = 20
num_hash_functions = 3

hash_functions = [
    lambda x: int(hashlib.md5(x.encode()).hexdigest(), 16),
    lambda x: int(hashlib.sha1(x.encode()).hexdigest(), 16),
    lambda x: int(hashlib.sha256(x.encode()).hexdigest(),
16)
]

bloom_filter = BloomFilter(size, hash_functions)

window = tk.Tk()
```

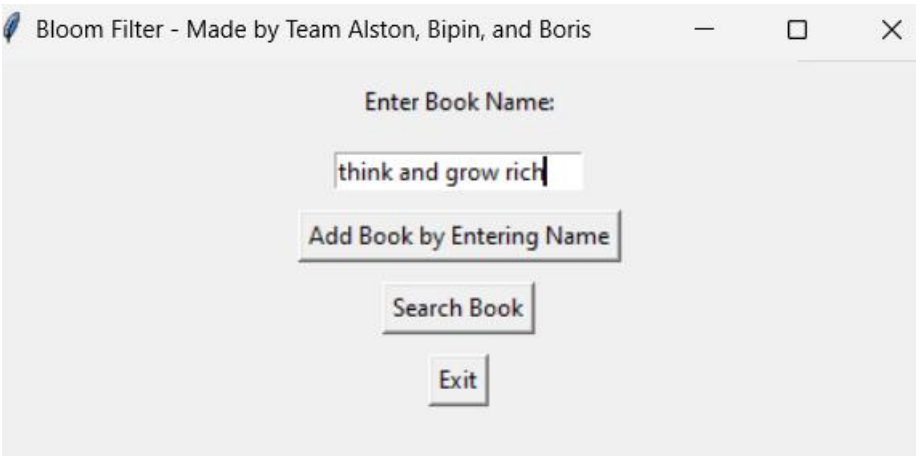
```
window.title("Bloom Filter - Made by Team Alston, Bipin, and
Boris")
window.geometry("400x300")
window.resizable(True, True)

label = tk.Label(window, text="Enter Book Name:")
entry = tk.Entry(window)
add_button = tk.Button(window, text="Add Book by Entering
Name", command=add_element)
search_button = tk.Button(window, text="Search Book",
command=search_element)
exit_button = tk.Button(window, text="Exit",
command=exit_program)

label.pack(pady=10)
entry.pack(pady=5)
add_button.pack(pady=5)
search_button.pack(pady=5)
exit_button.pack(pady=5)

window.mainloop()
```

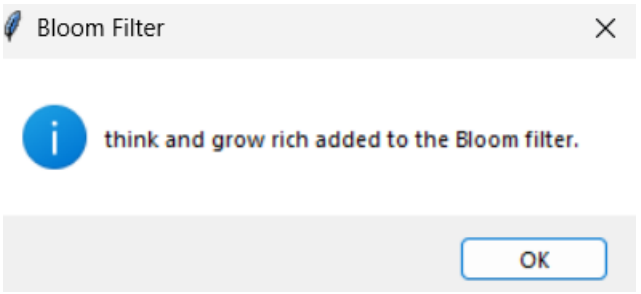
- Results: Input and Output

Adding book

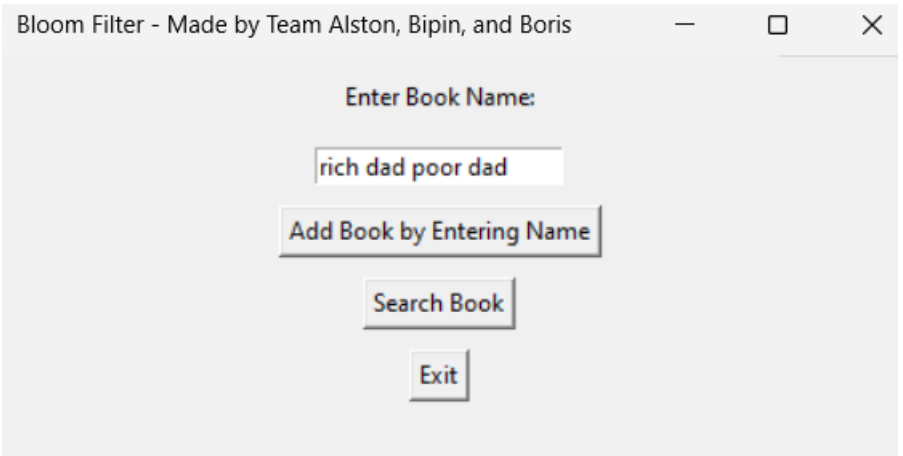


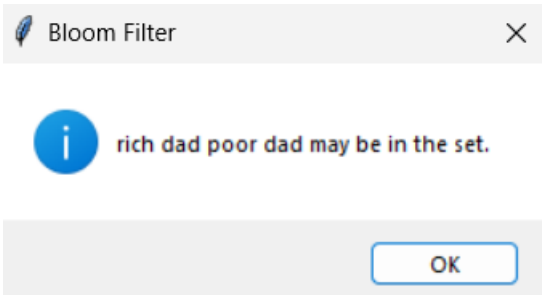




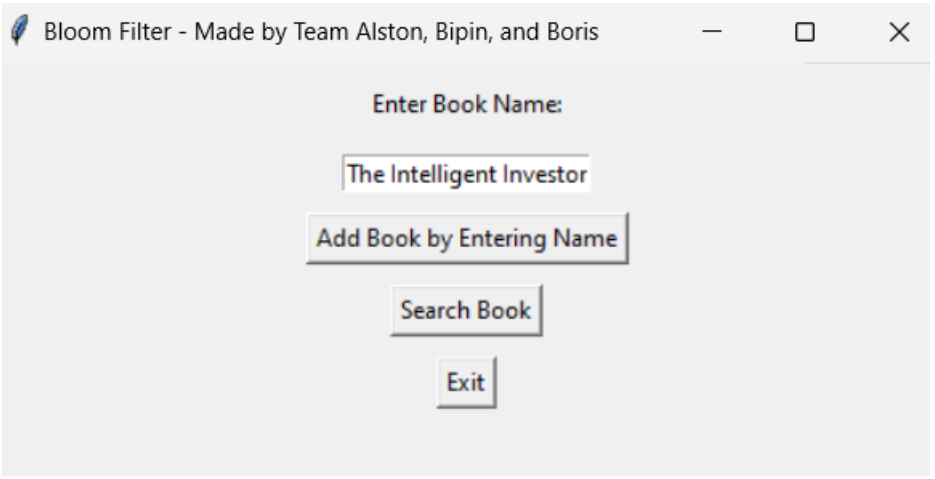


Searching Book:





Searching book not present in library:



Conclusion:	Able to implement Bloom filters for real world problems.
References:	Elearn Notes: https://elearn.dbit.in/course/view.php?id=258 Python Documentation: https://docs.python.org/3/

Don Bosco Institute of Technology
Department of Computer Engineering

Academic year – 2023-2024

Big Data Analytics

Assessment Rubric for Experiment No. : 04

Performance Date :
Submission Date :

Title of Experiment : Implement Bloom Filter using any programming language

Year and Semester : IV^{rh} Year and VIIth Semester

Batch : B

Name of Student: Bipin Dinesh Giri

Roll No. : 24

Performance	**Poor**	**Satisfactory**	**Good**	**Excellent**
	1 point	2 points	3 points	4 points
Results and Documentations	**Poor**	**Satisfactory**	**Good**	**Excellent**
	1 point	2 points	3 points	4 points
Viva	**Poor**	**Satisfactory**	**Good**	**Excellent**
	1 point	2 points	3 points	4 points
Timely Submission	**Submission beyond 7 days of the deadline**	**Late submission till 7 days**	**Submission on time**	
	1 points	2 points	3 points	