Abbottabad University of Science & Technology

Online Bookstore Search Engine

By

Fatima Bibi 14647

Supervisor

(Sir Jamal Abdul Ahad)

Table of Contents

Introduction:

1.1 **Objective:**

The primary goal of this project is to create a search engine for an online bookstore that efficiently manages and retrieves book data. Hash maps are employed to ensure quick access, and a user-friendly GUI is designed for seamless interactions.

1.2 Key Features**:**

- Fast lookup and insertion using hash maps.
- GUI-based interface for adding and searching books.
- Capability to handle prefix-based searches.

1.3 **Use Case:**

Online bookstores often manage large inventories of books. Providing users with a fast and efficient way to search and manage books can improve user satisfaction and operational efficiency.

2. Methodology

2.1 Technological Stack:

1. **Programming Language:** Python
2. **Data Structure:** Hash Map
3. **GUI Library:** Tkinter

2.2 Workflow:

1. **Book Addition:**
   - Users can add books by specifying the title and author.
   - The system validates inputs and stores the data in a hash map.
2. **Search Functionality:**
   - Users input a search query.
   - The system checks the hash map for exact matches or titles starting with the query.
   - Results are displayed in a list box within the GUI.

2.3 Design Considerations:

- **Efficiency:** Hash maps offer constant time complexity for insertion and lookup operations.
- **User Experience:** The GUI is designed to be intuitive and easy to use, catering to non-technical users.
- **Validation:** Input fields are validated to ensure completeness and prevent duplicate entries.

3. Implementation Details

3.1 Core Functionalities:

- **Book Storage:** Books are stored in a hash map for efficient management. Example:

```
book_map = {
  "The Great Gatsby": {"author": "F. Scott Fitzgerald"},
  "1984": {"author": "George Orwell"},
  # Additional books...
}
```

- **Add Book:** A function to add new books, ensuring no duplicates:

```
def add_book():
  title = title_entry.get().strip()
  author = author_entry.get().strip()
  if not title or not author:
    messagebox.showwarning("Warning", "Both Title and Author are required!")
    return
  if title in book_map:
    messagebox.showinfo("Info", f"'{title}' is already in the collection!")
    return
  book_map[title] = {"author": author}
  title_entry.delete(0, tk.END)
  author_entry.delete(0, tk.END)
  messagebox.showinfo("Success", f"Book '{title}' added!")
```

- **Search Books:** A function to search for books based on titles or prefixes:

```
def search_books():
    query = search_entry.get().strip()
    if not query:
        messagebox.showwarning("Warning", "Please enter a search term!")
        return
    results_list.delete(0, tk.END)
    for title, details in book_map.items():
        if title.lower().startswith(query.lower()):
            results_list.insert(tk.END, f"Title: {title}, Author: {details['author']}")
    if results_list.size() == 0:
        messagebox.showinfo("No Results", "No books found matching your search!")
```

3.2 GUI Design:

- Input fields for adding and searching books.
- Buttons for "Add Book" and "Search" functionalities.
- A list box to display search results.

3.3 Enhancements to GUI:

- Clear feedback messages for successful and failed operations.
- Dynamic resizing of the list box to accommodate search results.

4. Results

Performance:

- The use of hash maps ensures constant time complexity for insertion and lookup operations.
- The GUI provides an intuitive interface, simplifying interactions for users.

Functionality Testing:

1. Added books successfully using the GUI.
2. Performed searches with full titles and prefixes, obtaining accurate results.

3. Handled invalid inputs gracefully with appropriate warnings.

Scalability Testing:

- Tested with a large number of book entries to ensure the system remains responsive.

5. Advantages

1. **Efficiency:** Hash maps provide fast access to book records.
2. **Scalability:** The system can manage a growing collection of books efficiently.
3. **User-Friendliness:** Tkinter-based GUI ensures ease of use.
4. **Flexibility:** Prefix-based search allows users to find books even with partial information.

6. Limitations and Future Enhancements

Limitations:

- Search functionality is limited to titles; authors cannot be queried directly.
- Data is not persistent across sessions, as no database integration is implemented.
- Current design does not support advanced filtering or sorting options.

Future Enhancements:

1. Add the ability to search books by author names.
2. Organize books into categories for better management.
3. Integrate a database to store book data persistently.
4. Implement sorting and filtering features for search results.
5. Extend the GUI to include book descriptions, publication years, and genres.
6. Enable export and import of book data in CSV or JSON format.

7. Conclusion

This project demonstrates the effective use of hash maps for managing and searching book data in an online bookstore. The intuitive GUI further enhances user experience, making the system both efficient and accessible. Future developments can expand its functionality and scalability to better serve the needs of users.

Key Learnings:

- Hash maps are a powerful data structure for applications requiring quick data retrieval.
- A well-designed GUI significantly improves user engagement.
- Modular and extensible codebase allows for easier future enhancements.

---

8. Appendix

Code: The complete Python code for the project is included in the accompanying files.

References:

- Python Official Documentation
- Tkinter GUI Programming Guide
- Hash Map Implementation Concepts