**Efficient Library Management System (ELMS)**

**1. Introduction**

The **Efficient Library Management System (ELMS)** is a C++ project designed to demonstrate the integration of **Object-Oriented Programming (OOP) principles** with multiple **data structures** such as **Stack, Queue, Linked List, Hash Table, and Binary Search Tree (BST)**.  
It manages library operations like **borrowing, returning, employee search, and book search** efficiently.

This project also showcases concepts of **templates, file handling, and exception handling**, making it an ideal academic and practical implementation of advanced C++ programming.

**2. Objectives**

* Apply **OOP principles** (Abstraction, Encapsulation, Composition).
* Demonstrate the use of **multiple data structures** in a real-world problem.
* Enable efficient **book and employee management** using advanced C++ concepts.
* Showcase **file handling and exception handling** for robust software design.
* Provide a **structured, extensible, and maintainable project** suitable for academic or professional use.

**3. Features**

* OOP concepts: **Composition, Encapsulation, Abstraction**
* Book borrowing & returning handled with Queue
* Overdue and undo/redo functionality with Stack
* Fast book searching via Binary Search Tree (BST)
* Employee record management using Hash Table
* Data persistence via file handling (DataFile.txt)
* Exception handling for safe operations
* Use of templates for generic implementations
* User-friendly console interface with structured output

**4. Technologies & Concepts Used**

* **Language:** C++ (C++11/14 standard compatible)
* **OOP Concepts:** Abstraction, Encapsulation, Composition
* **Data Structures:**
  + Stack
  + Queue
  + Linked List
  + Hash Table
  + Binary Search Tree (BST)
* **Other Concepts:** Templates, File Handling, Exception Handling

**5. Program Flow**

1. Load book & employee data from DataFile.txt.
2. Display all loaded information.
3. Perform **book borrowing operations** (queue-based).
4. Perform **book returning operations** (queue + stack for undo).
5. Undo specific operations using **stack**.
6. Perform **book searching** using BST.
7. Perform **employee searching** using Hash Table.
8. Display program completion message.

**6. How to Run**

**6.1 Clone the Repository**

git clone https://github.com/Muneeb-techpro/efficient-library-management-system.git

cd efficient-library-management-system/src

**6.2 Compile the Project**

g++ \*.cpp -o LibraryApp

**6.3 Run the Executable**

* **On Linux / macOS:**

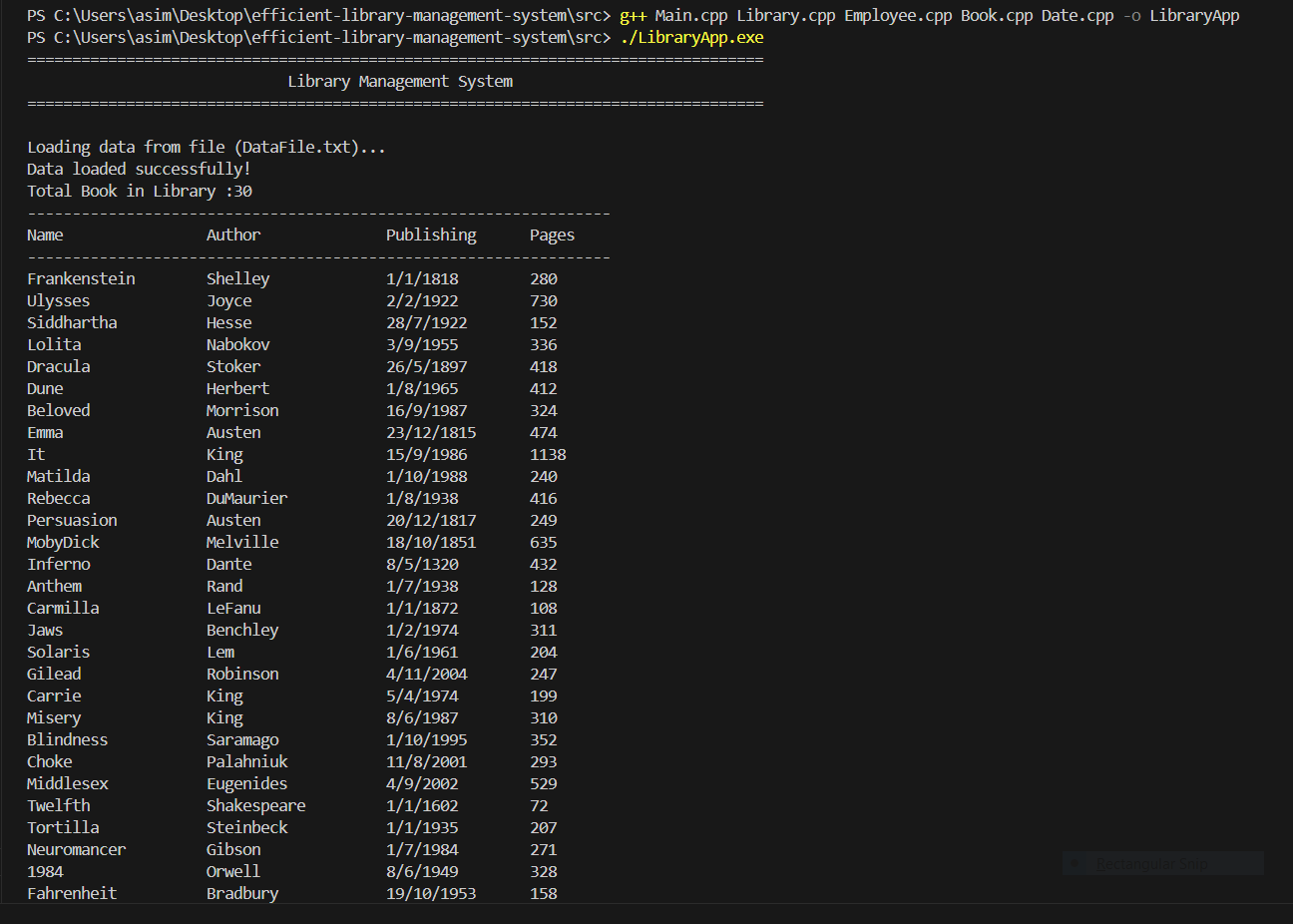
./LibraryApp

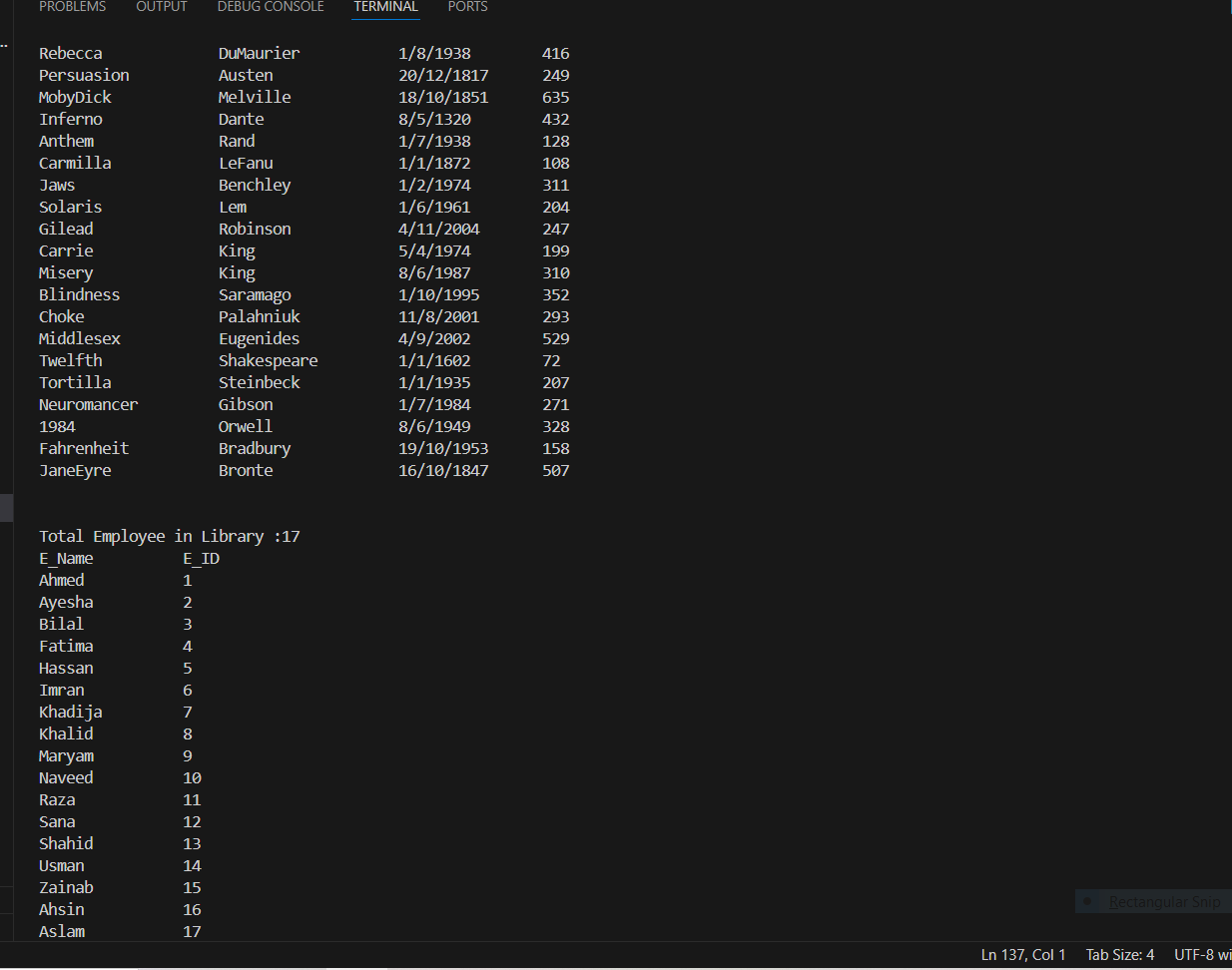
* **On Windows (PowerShell or CMD):**

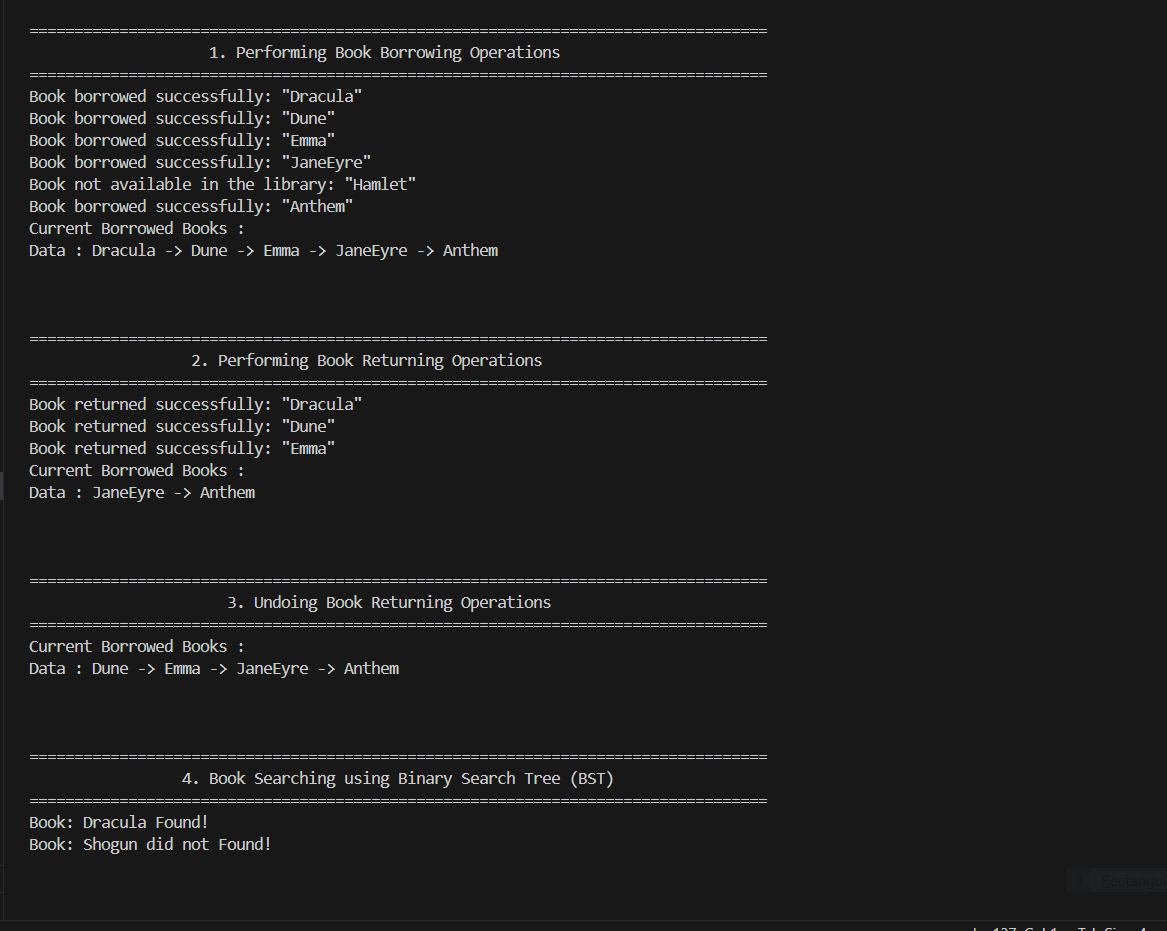
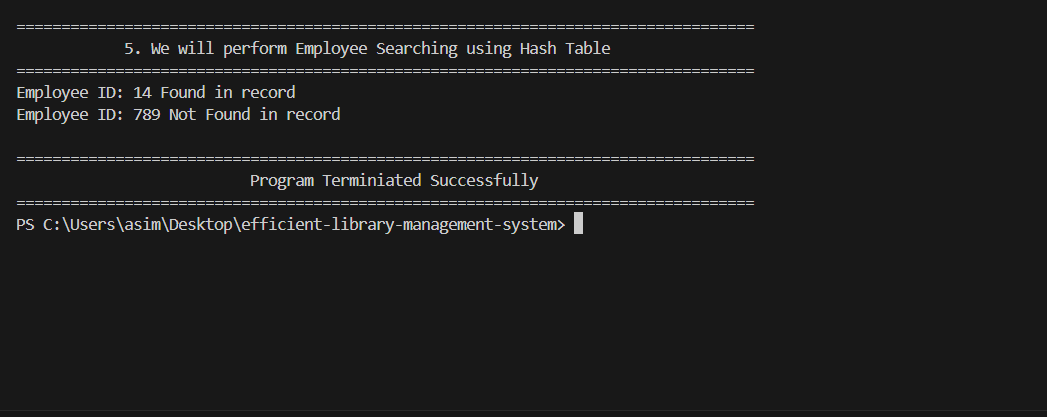
LibraryApp.exe

**7. Output (Screenshots + Logs)**

**📷 Screenshots**

* 01\_output.png – Program start screen
* 02\_output.png – Data successfully loaded

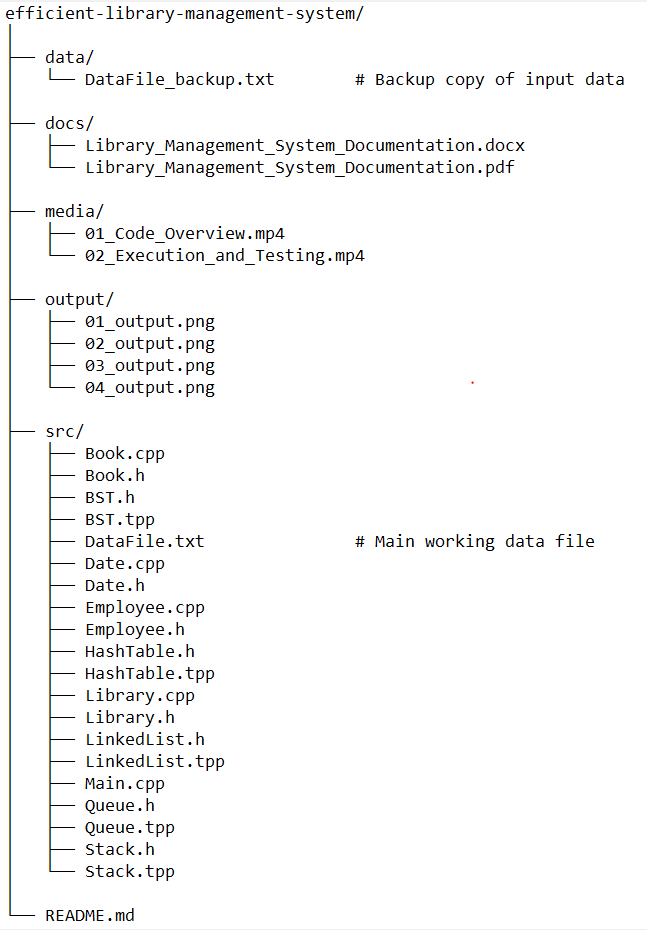


* 03\_output.png – Borrowing operations
* 04\_output.png – Employee search results

**📄 Complete Output Log**

See: program\_output.txt

**8. Folder Structure**



**9. Exception Handling**

The project includes **robust exception handling** for cases such as:

* Missing DataFile.txt
* Invalid book or employee IDs
* Queue underflow/overflow in borrowing/returning operations
* Stack underflow in undo operations

**10. Future Enhancements**

These are some optional future enhancements that may be considered:

* Add a **GUI interface** (Qt/JavaFX/React frontend with backend integration).
* Add **database support** (MySQL/SQLite instead of text file).
* Implement **book recommendation system** using Graph.
* Add **user authentication** and member login system.
* Support for **digital e-books** and issue tracking.

**11. Conclusion**

The **Efficient Library Management System (ELMS)** successfully demonstrates how multiple data structures and OOP principles can be combined to build a robust and efficient C++ application.

It not only serves as a **functional library system** but also as a **learning project** for students to understand **data structures, OOP, templates, and exception handling** in real-world problem solving.