GREEN UNIVERSITY OF BANGLADESH

Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Fall, Year: 2025), B.Sc. in CSE (Day)

Smart Library System

Course Title: Artificial Intelligence Lab
Course Code: CSE 316
Section: 223 D1

<u>Students Details</u>

Name	**ID**
Saifulla Tanim	222002014
Mim Akter	222002104

Submission Date: 28.12.2025
Course Teacher's Name: Ms. Abida Sultana

[For teachers use only: Don't write anything inside this box]

Lab Project Status	
Marks:	**Signature:**
Comments:	**Date:**

Contents

Chapter 1

Introduction

1.1 Overview

The **Smart Library System** is an intelligent, interactive, and user-centric platform designed to modernize traditional library management. It offers book searching, graphical shelf visualization, automated capacity monitoring, and AI-based shortest path guidance for locating resources efficiently. [1]. By integrating a dynamic UI and algorithm-based decision support, the system enhances accessibility, usability, and overall learning experience.

1.2 Motivation

Libraries often face challenges such as inefficient searching, difficulty in tracking book locations, and manual monitoring of shelf capacity. Students and librarians lose valuable time while locating books physically. This project is motivated by the need to eliminate these inefficiencies through automation, visual guidance, and intelligent searching technology that makes access faster, smarter, and more engaging. [2].

1.3 Problem Definition

Despite digital advancements, ordinary library systems lack guided navigation, real-time visualization, and automated book management capabilities. Users have no system-driven support to identify shelf load, find shortest walking routes, or interact with library resources meaningfully. Therefore, an enhanced smart system is needed to simplify search operations and improve navigation experience.

1.3.1 Problem Statement

To design and develop a system that helps users search, locate, and visualize books while integrating AI-based route guidance and smart inventory tracking for efficient navigation inside a library.

1.3.2 Complex Engineering Problem

The engineering complexity lies in merging multiple subsystems — book management, interface design, route optimization, data visualization, and user interaction — into a seamless, intelligent solution. The system deals with dynamic data, real-time display updates, decision automation, and algorithmic reasoning to achieve high usability. [3].

Table 1.1: Attributes addressed by the Smart Library System

Project Attribute	**How It Is Addressed in Smart Library System**
P1: Depth of knowledge required	Involves UI design, data structures, algorithm reasoning (A*), CRUD operations, visualization, and system development.
P2: Conflicting requirements	Balances usability vs complexity, capacity limits vs book addition, and performance vs visual animation.
P3: Depth of analysis required	Requires analysis of routing algorithms, path cost prediction, usability evaluation, and functional integration.
P4: Familiarity of issues	Addresses problems like inefficient searching, poor navigation, shelf overloading, and lack of automation.
P5: Extent of applicable standards	Applies UI principles, data validation, search logic, inventory constraints, and software engineering models.
P6: Stakeholder involvement	Users (students), administrators (librarians), and developers interact through decision control and operational updates.
P7: Interdependence	Searching module depends on database, routing depends on map data, CRUD depends on constraints and UI state.

1.4 Design Goals/Objectives

The major objectives of the Smart Library System are:

- To provide an intelligent and user-friendly library interface.
- To enable fast and efficient book searching and filtering.
- To visualize shelf occupancy and resource distribution graphically.
- To integrate AI-based A* search for shortest route guidance.
- To support book management through CRUD operations.
- To automate decision support and improve accessibility.

1.5 Application

The system can be applied in:

- University and college libraries
- Public libraries and reading zones
- Academic institutions and resource centers
- Book retail stores with shelf navigation support

It helps students, librarians, and visitors by making book access easier, improving operational visibility, and enabling smart navigation inside the facility.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

The Smart Library System is designed to modernize traditional library operations by integrating artificial intelligence techniques with an interactive graphical user interface. The system focuses on efficient book management, intelligent searching, and automated shortest path discovery inside the library environment. By utilizing the A* search algorithm, the system ensures optimal navigation from the library entrance to the desired bookshelf.

2.2 Project Details

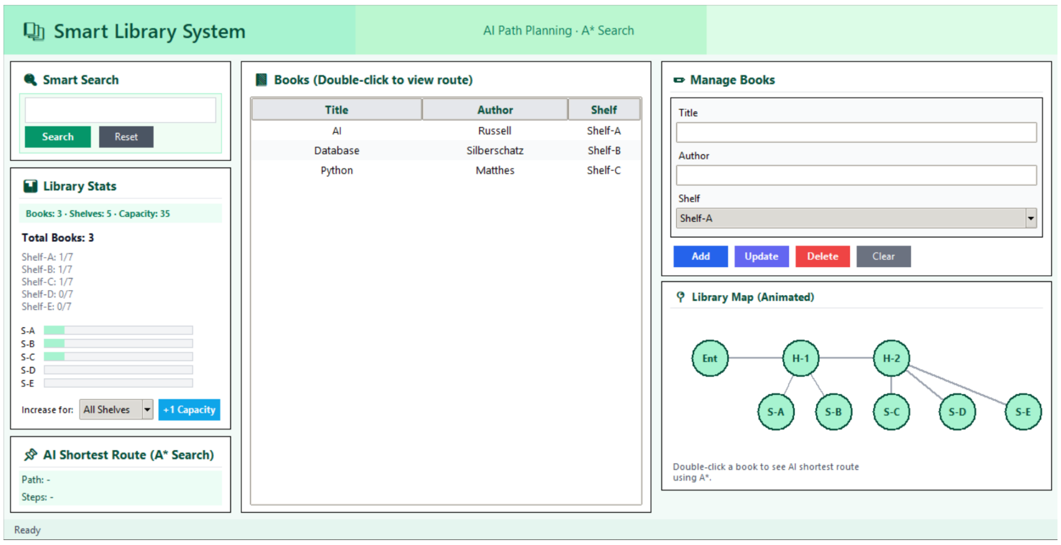
The project consists of three core modules: Smart Search, Book Management, and AI-based Path Planning. These modules work together to provide a user-friendly and intelligent library management experience. The system architecture is designed to be modular, scalable, and easy to maintain.

Graphical User Interface Design

The graphical user interface (GUI) of the Smart Library System is developed using Python Tkinter with an emphasis on clarity, usability, and modern design principles. [4]. The interface adopts a card-based layout and a soft color theme to provide a visually appealing and user-friendly experience. Proper spacing, alignment, and consistent typography are maintained to ensure readability and ease of navigation.

The GUI is logically divided into three primary panels. The left panel contains the smart search functionality and real-time library statistics, allowing users to quickly search for books and monitor shelf usage and capacity. The central panel displays a structured table of available books, including title, author, and shelf information, enabling efficient browsing and selection. The right panel is dedicated to book management operations and AI-based visualization, where users can add, update, or delete books and observe the shortest navigation path within the library.

Additionally, the interface supports interactive features such as double-click selection, dynamic status updates, and animated path visualization. These features improve user engagement and reduce manual effort. Overall, the GUI design ensures smooth interaction between users and system functionalities while maintaining a professional and intuitive appearance suitable for a smart library environment.


Smart Library System

AI Path Planning · A* Search

Smart Search

Library Stats

Books: 3 · Shelves: 5 · Capacity: 35

Total Books: 3

Shelf-A: 1/7
Shelf-B: 1/7
Shelf-C: 1/7
Shelf-D: 0/7
Shelf-E: 0/7

Increase for: All Shelves

AI Shortest Route (A* Search)

Path: -
Steps: -

Books (Double-click to view route)

Title	Author	Shelf
AI	Russell	Shelf-A
Database	Silberschatz	Shelf-B
Python	Matthes	Shelf-C

Manage Books

Title:
Author:
Shelf: Shelf-A

Library Map (Animated)

Double-click a book to see AI shortest route using A*.

Ready

Figure 2.1: Graphical User Interface of the Smart Library System

2.3 Implementation

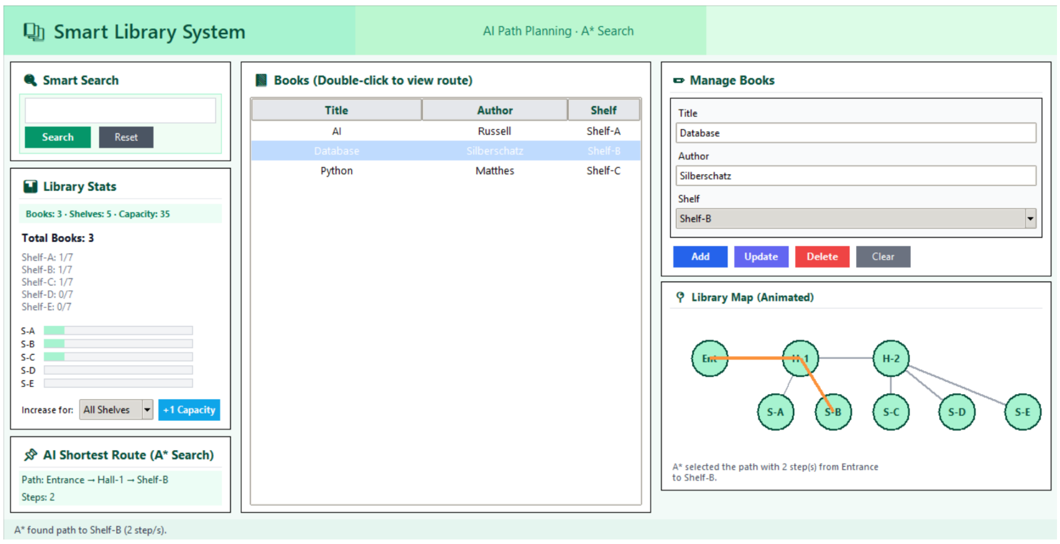
The system is implemented using Python, where Tkinter is used for GUI development and core logic is written following modular programming principles. [5]. Data related to books, shelves, and capacities are maintained dynamically in structured data formats. Event-driven programming is applied to handle user interactions such as searching, adding, updating, and deleting books.

AI-Based Path Visualization

The Smart Library System visually demonstrates the shortest route from the entrance to a selected shelf using an animated map. Nodes represent halls and shelves, while edges represent possible paths. When a user selects a book, the A* search algorithm computes the optimal route and highlights it dynamically on the map, improving understandability and user experience.

The Workflow

The workflow of the Smart Library System follows a structured and user-centric process. Initially, the system loads book data, shelf capacity, and the library layout represented as a graph. Users can search books by title, author, or shelf, and the system dy-



Books: 3 · Shelves: 5 · Capacity: 35

Total Books: 3

Shelf-A: 1/7
Shelf-B: 1/7
Shelf-C: 1/7
Shelf-D: 0/7
Shelf-E: 0/7

Title	Author	Shelf
AI	Russell	Shelf-A
Database	Silberschatz	Shelf-B
Python	Matthes	Shelf-C

Title: Database
Author: Silberschatz
Shelf: Shelf-B

Path: Entrance → Hall-1 → Shelf-B
Steps: 2

A* selected the path with 2 step(s) from Entrance to Shelf-B.

A* found path to Shelf-B (2 step/s).

Figure 2.2: AI-Based Library Map with A* Shortest Path Visualization

namically displays matching results. Upon selecting a book, the A* algorithm computes the shortest path from the entrance to the target shelf, which is shown both textually and through animated visualization. Book management actions automatically update shelf statistics, ensuring real-time accuracy and smooth operation.

Tools and Libraries

The Smart Library System is developed using Python as the primary programming language. Tkinter is used for designing the graphical user interface, while the `ttk` module enhances widget styling. The `heapq` library supports efficient priority queue implementation for the A* algorithm, and the `time` module enables animated path visualization. These tools ensure efficient development, maintainability, and smooth system performance. [2,6].

Implementation Details (with Screenshots and Programming Codes)

The system is implemented in a modular manner, separating data management, user interface handling, and AI-based path planning. Tkinter widgets and frames are used to build interactive panels, while event-driven programming handles user actions. The A* search algorithm calculates the shortest route using a heuristic-guided approach, and the resulting path is animated on the library map. Screenshots and relevant code segments are included to demonstrate system functionality and implementation logic.

2.4 Algorithms

This section presents the main algorithm used in the Smart Library System. The algorithm is written based on the actual implementation logic of the project using Python and object-oriented programming principles. It combines book searching, management, and AI-based shortest path planning in a single workflow.

Algorithm 1: Combined Algorithm of the Smart Library System

```
Input: User action (search / select book / manage book)
Output: Search result, updated records, shortest path
1  Load book list, shelf capacity, and library graph
2  if user performs search then
3      Read search keyword
4      Filter books by title, author, or shelf
5      Display matching book list
6  else
7      if user selects a book then
8          Identify target shelf
9          Apply A* algorithm to find shortest path
10         Display path and number of steps
11         Visualize path on library map
12     else
13         Read book details and selected operation
14         if operation is Add / Update / Delete then
15             Modify book records
16             Update shelf statistics
17 Update system status and interface
```

Chapter 3

Performance Evaluation

3.0.1 Simulation Environment / Simulation Procedure

The system is tested on a standard desktop environment using Python. Various scenarios were simulated by selecting different books located on different shelves to evaluate the accuracy and responsiveness of the path planning algorithm.

Simulation Setup

The library layout is modeled as a graph where nodes represent halls and shelves, and edges represent connections between them. Each simulation starts from the entrance node and ends at the target shelf node.

Test Scenarios

Multiple test cases were conducted by selecting books from different shelves to verify path correctness, step count accuracy, and animation consistency.

3.0.2 Results Analysis / Testing

The results demonstrate that the A* algorithm consistently finds the shortest route with minimum steps. The system responds efficiently to user interactions without noticeable delay. [3].

Result Portion 1

The system successfully calculated optimal paths for all tested shelves with accurate step counts.

Result Portion 2

Animated visualization helped users clearly understand navigation paths within the library.

Result Portion 3

Dynamic updates in book records and shelf capacity were handled smoothly without affecting performance.

3.0.3 Results Overall Discussion

Overall, the Smart Library System exhibits stable and efficient performance throughout the evaluation process. The integration of intelligent searching, dynamic data management, and AI-based path planning ensures accurate results and smooth system behavior. Visual representation of navigation paths enhances user understanding, while real-time updates to book records and shelf statistics contribute to system reliability. These results indicate that the system effectively fulfills its design objectives and provides a practical solution for smart library management.

Complex Engineering Problem Discussion

The project addresses a complex engineering problem by combining artificial intelligence techniques with real-time user interaction and visualization. Determining the optimal navigation path within a constrained library layout requires efficient handling of graph structures and heuristic-based decision making. This challenge is successfully resolved using the A* search algorithm, which balances path cost and heuristic estimation to produce optimal solutions. The seamless integration of algorithmic logic with an interactive graphical interface demonstrates the system's capability to solve real-world engineering problems in an efficient and user-friendly manner.

Chapter 4

Conclusion

4.1 Discussion

This project successfully presents the design and implementation of a Smart Library System that integrates artificial intelligence with a modern graphical user interface. The system demonstrates how intelligent search, automated book management, and AI-based shortest path planning can improve traditional library operations. By applying the A* search algorithm, the system efficiently guides users to desired book locations while providing clear visual feedback. Overall, the project achieves its intended objectives and highlights the practical application of AI techniques in a real-world library environment.

4.2 Limitations

Despite its effectiveness, the current system has certain limitations. The library layout is predefined and static, which means changes in physical arrangement are not handled dynamically. The system also assumes an obstacle-free environment and does not consider real-time factors such as user congestion or temporary blockages. Additionally, data is stored in memory during runtime, limiting long-term persistence and scalability.

4.3 Scope of Future Work

There is significant scope for enhancing the Smart Library System in future. The system can be extended by integrating a database for persistent data storage and multi-user access. Real-time obstacle detection and dynamic path recalculation may be added to improve navigation accuracy. Furthermore, mobile or web-based interfaces, RFID-based book tracking, and advanced analytics can be incorporated to transform the system into a fully automated and intelligent library management platform.

References

[1] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition, 2010.

[2] Python Software Foundation. Tkinter — python interface to tcl/tk. `https://docs.python.org/3/library/tkinter.html`, 2024. Accessed: 2025-01-05.

[3] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[4] John E. Grayson. *Python and Tkinter Programming*. Manning Publications, 2015.

[5] Mark Lutz. *Learning Python*. O'Reilly Media, 5th edition, 2013.

[6] Python Software Foundation. heapq — heap queue algorithm. `https://docs.python.org/3/library/heapq.html`, 2024. Accessed: 2025-01-05.