**Software Requirements Specification**

\*\*SRS for\*\*

**Library Management System**

Prepared by

Specialization: Data Science, Computer Science Engineering  
Name:

* Shefali Upadhyay(500120337)
* Ritika Tyagi(500124384)
* Rishika Goyal(500126501)
* Naman Dixit(500125539)
* Kashyapee Anchal(500120297)

School of Computer Science;

UNIVERSITY OF PETROLEUM & ENERGY STUDIES,  
DEHRADUN - 248007, Uttarakhand

**Table of Contents**

1. Revision History
2. Introduction  
   2.1 Purpose of the Project  
   2.2 Target Beneficiaries  
   2.3 Scope of the Project  
   2.4 References
3. Project Description  
   3.1 Reference Technology  
   3.2 Data Structures  
   3.3 SWOT Analysis  
   3.4 Project Features  
   3.5 User Classes and Characteristics  
   3.6 Constraints  
   3.7 Design Diagrams  
   3.8 Assumptions and Dependencies
4. System Requirements  
   4.1 User Interface  
   4.2 Software Interface  
   4.3 Database Interface  
   4.4 Protocols
5. Non-functional Requirements  
   5.1 Performance Requirements  
   5.2 Security Requirements  
   5.3 Software Quality Attributes
6. Other Requirements
7. Appendices A. Glossary  
   B. Analysis Model  
   C. Issues List

**Revision History**

| **Date** | **Change** | **Reason for Changes** | **Mentor Signature** |
| --- | --- | --- | --- |
| 22/04/2025 | Initial draft | First version of LMS SRS |  |

**1. Introduction**

**1.1 Purpose of the Project**

This project aims to design and develop a desktop-based Library Management System using Python's Tkinter GUI toolkit. The system aims to simplify and streamline the book management process for library administrators through CRUD operations and data export features.

**1.2 Target Beneficiaries**

* College and School Libraries
* Personal book collectors
* Librarians and administrative staff
* Developers learning desktop application design

**1.3 Scope of the Project**

This Library Management System will enable the following:

* Add, update, delete, and search for book records
* View records in tabular form
* Export records to CSV
* Toggle between dark/light mode for accessibility

**1.4 References**

* Tkinter Documentation
* SQLite3 Documentation
* IEEE SRS Documentation Standards
* Pandas Documentation

**2. Project Description**

**2.1 Reference Technology**

The system is built using Python 3 with Tkinter for GUI and SQLite3 for the database. Pandas is used for CSV export functionality.

**2.2 Data Structures**

The main data consists of:

* Book ID (int, primary key)
* Title (text)
* Author (text)
* Year (int)
* ISBN (text)

**2.3 SWOT Analysis**

**Strengths**:

* Simple GUI, quick access to records, light-weight, fast local storage

**Weaknesses**:

* No authentication, single-user design

**Opportunities**:

* Extendable to cloud, multi-user access, integration with QR scanners

**Threats**:

* Platform dependence (Python required), limited to basic library functions

**2.4 Project Features**

* GUI with dark/light toggle
* Book record CRUD operations
* Search functionality with multiple filters
* Export records to CSV
* TreeView widget for data display

**2.5 User Classes and Characteristics**

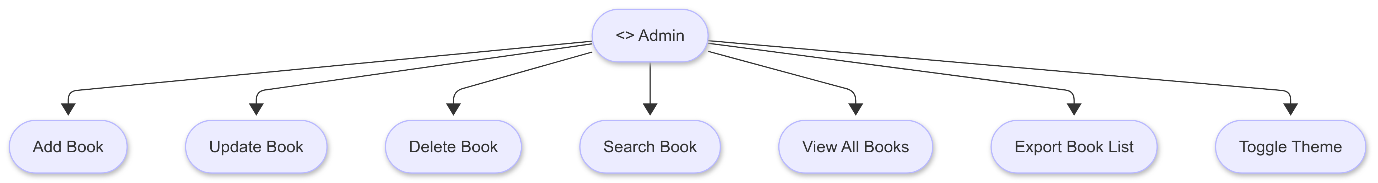
* **Admin/Librarian**: Basic computer knowledge, handles all functionalities
* **Developer (optional)**: May extend functionalities or perform maintenance

**2.6 Design and Implementation Constraints**

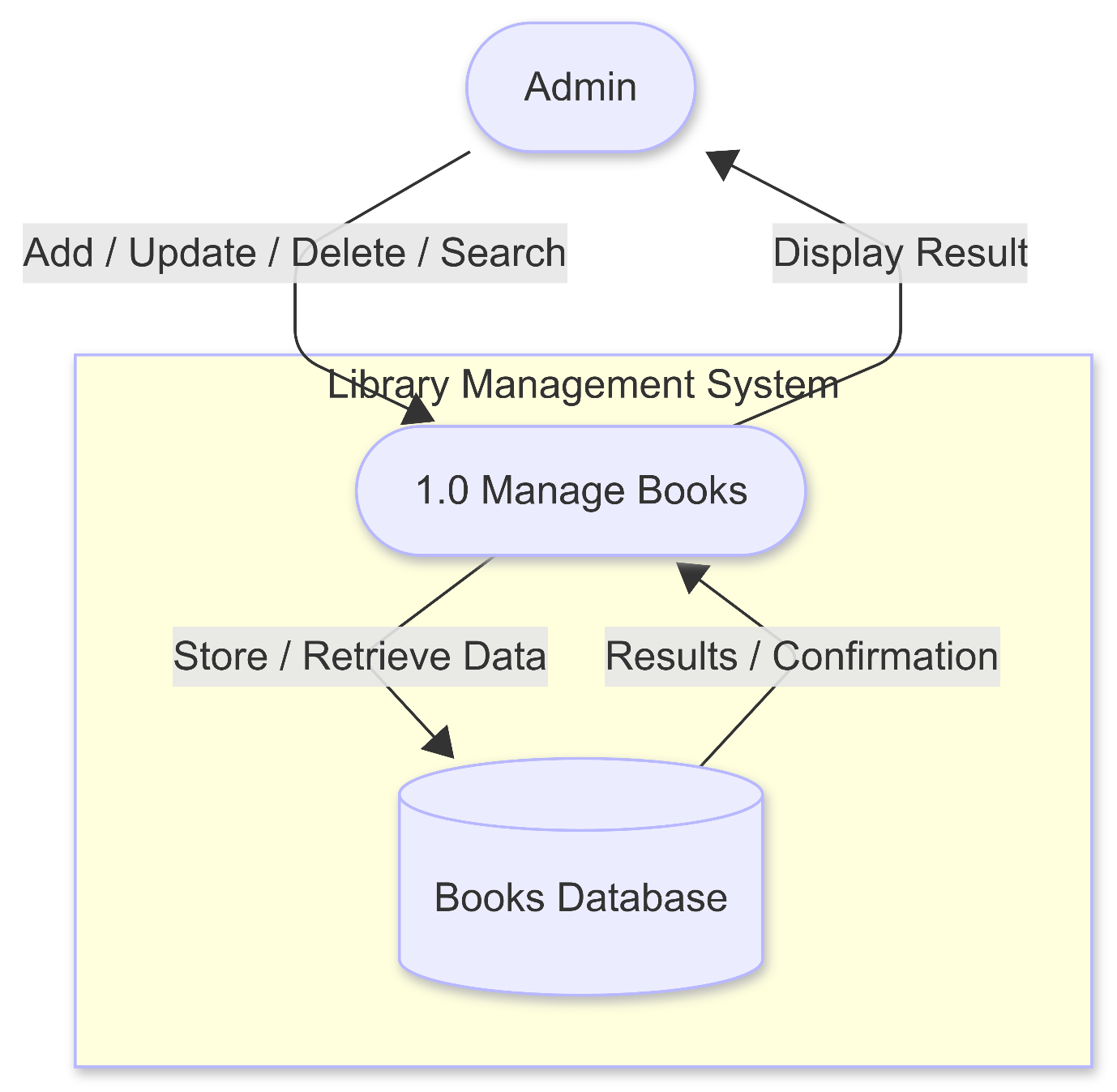
* Python and Tkinter must be pre-installed
* SQLite used for local data storage
* Export requires pandas library
* Interface designed for PC, not optimized for mobile

**2.7 Design Diagrams**

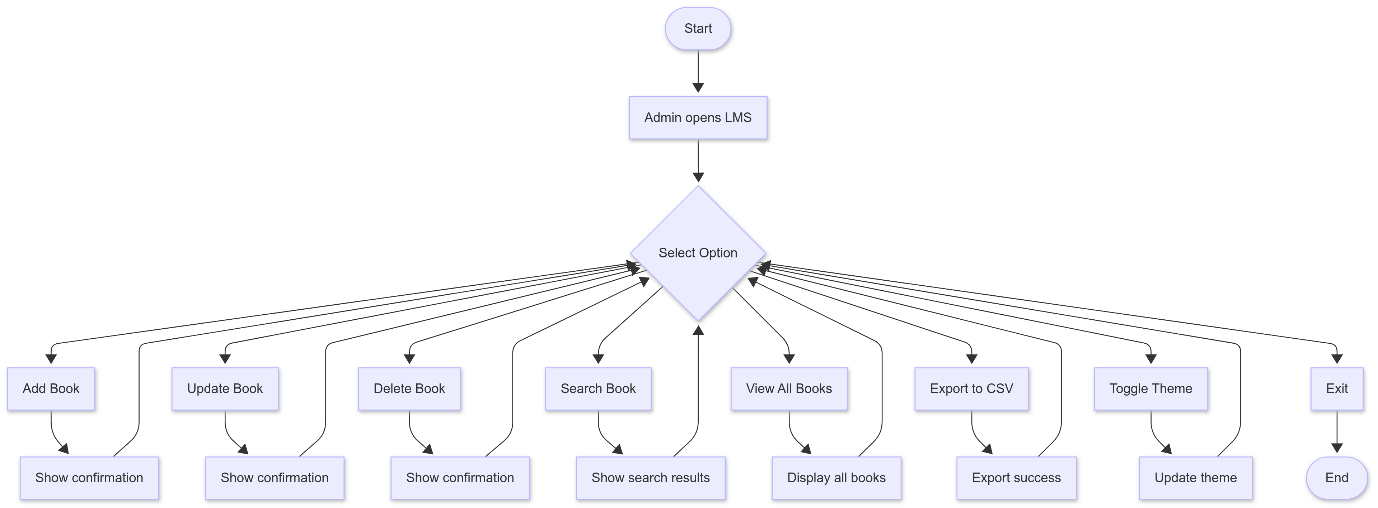
* **Use Case Diagram**: Showcases Add, Update, Delete, Search, Export interactions with Admin

****

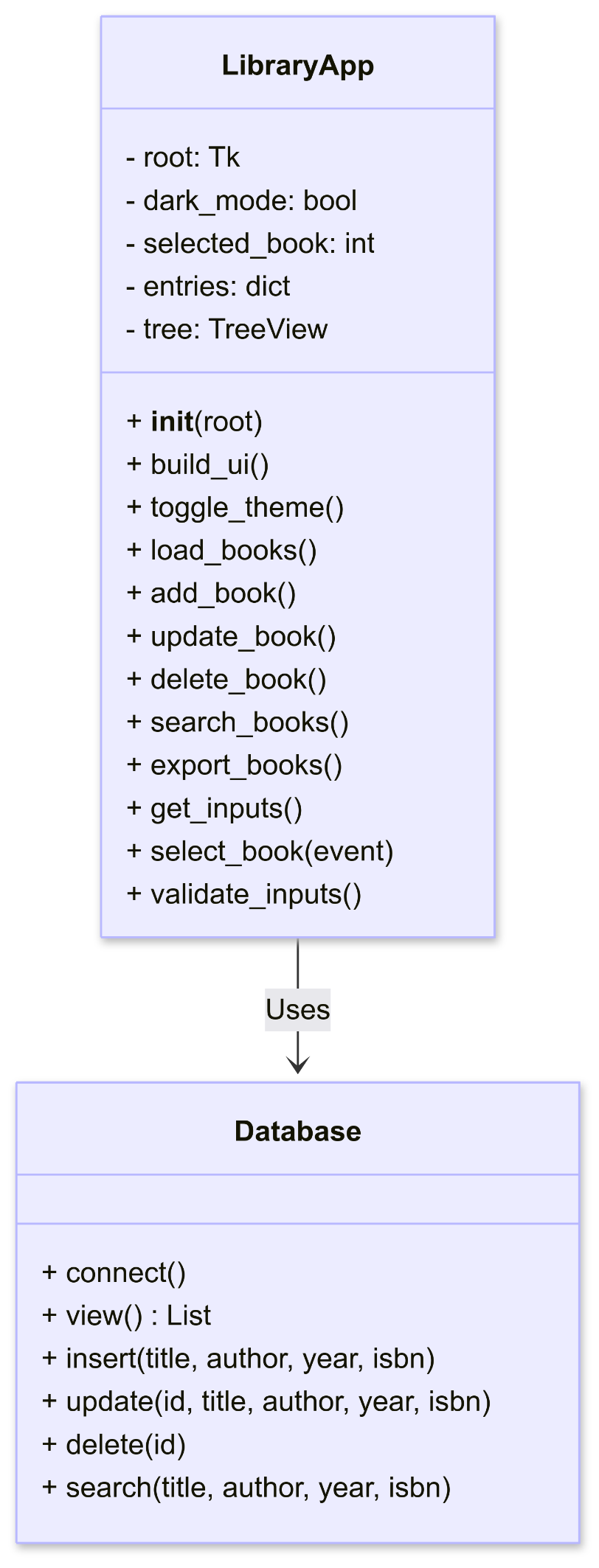
* **DFD Level 0**:
  + User → Interface → Database → Response



* **Activity Diagram**: Illustrates record manipulation flow



* **Class Diagram**: LibraryApp, DatabaseFunctions

** 2.8 Assumptions and Dependencies**

* Admin has basic technical proficiency
* Data entries are accurate
* CSV file required for initial database is present

**2.9 Methodology**

For the development of the Library Management System, the Waterfall Model was adopted due to its simplicity, structured approach, and suitability for small-scale academic projects. The project followed a sequential flow through the following phases:

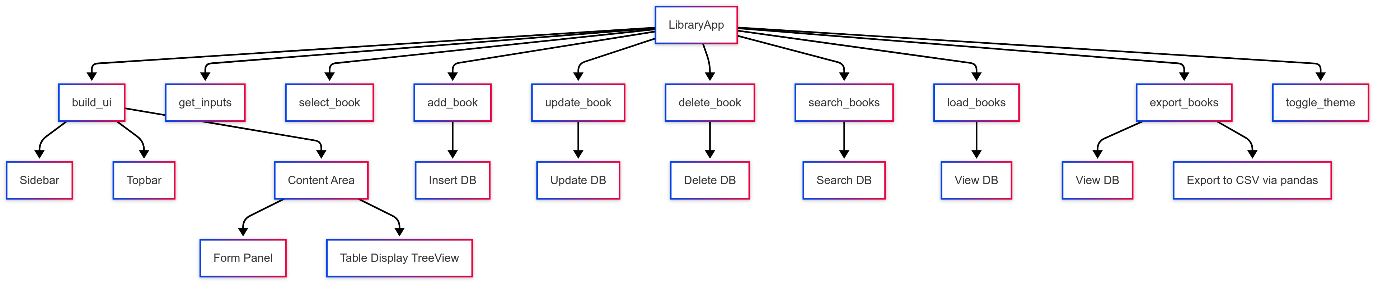
1. Requirements Gathering
   * Identified system objectives and functional needs (e.g., CRUD operations, CSV export, GUI design).
   * Defined user expectations and technical constraints.
2. System Design
   * Created design diagrams including use case, class, and activity diagrams.
   * Defined data structures and designed the UI layout.
3. Implementation
   * Developed the application using Python's Tkinter for GUI and SQLite3 for the backend.
   * Integrated Pandas for export functionality.
4. Testing
   * Conducted unit tests for all CRUD operations.
   * Verified UI responsiveness, database connectivity, and data export reliability.
5. Deployment
   * The application was packaged for local use on Windows and Linux platforms.
6. Documentation
   * Created the Software Requirements Specification (SRS) and technical documentation.

2.10 Structure Chart

The following structure chart represents the hierarchical decomposition of the Library Management System. At the top level, the system is managed by the LibraryApp module, which orchestrates the user interface, event handling, and database interaction logic. The application is divided into key functional components such as build\_ui(), get\_inputs(), select\_book(), and database-driven operations like add\_book(), update\_book(), delete\_book(), and export\_books().

Each functional unit is further broken down into UI components (e.g., Sidebar, Topbar, Form Panel, TreeView) or linked to corresponding database operations (e.g., Insert DB, Update DB, View DB). The structure emphasizes modularity, maintainability, and clear separation of concerns between the interface and data handling layers.

The visual representation of the structure chart illustrates the system’s logical architecture and flow of control from high-level functions to supporting modules.



**3. System Requirements**

**3.1 User Interface**

* User-friendly and responsive
* Toggle for dark and light mode
* Fields for title, author, year, and ISBN

**3.2 Software Interface**

* Python 3.8+
* Tkinter
* SQLite3
* Pandas for export functionality

**3.3 Database Interface**

* SQLite3
* Local .db file
* Table: books (id, title, author, year, isbn)

**3.4 Protocols**

* Internal local function calls, no networking used

**4. Non-functional Requirements**

**4.1 Performance Requirements**

* Database load time < 2 seconds
* Handles 1000+ records smoothly

**4.2 Security Requirements**

* Local file encryption (optional for .db file)
* No external access to database

**4.3 Software Quality Attributes**

* **Reliability**: Operates offline without server dependency
* **Maintainability**: Modular codebase
* **Usability**: Easy-to-use interface
* **Portability**: Runs on Windows/Linux/macOS with Python

**5. Other Requirements**

* Optionally allow import/export to/from Excel
* Can be enhanced to support user authentication
* Future integration with RFID/QR scanning systems

**6. Appendices**

**Appendix A: Glossary**

* **CRUD**: Create, Read, Update, Delete
* **Tkinter**: Python standard GUI library
* **SQLite3**: Lightweight SQL database engine
* **CSV**: Comma-Separated Values

**Appendix B: Analysis Model**

* Sample database screenshot
* Interface layout
* Output CSV snapshot

**Appendix C: Issues List**

* No multi-user access in current version
* No online/cloud support yet
* Input validation improvements under progress

**7. Conclusion**

**The Library Management System was successfully designed and implemented to streamline book management tasks through a user-friendly graphical interface. By utilizing Python’s Tkinter and SQLite3, the system ensures lightweight performance, fast access to records, and efficient handling of CRUD operations.**

**Key features such as search filters, CSV export, and a dark/light mode toggle enhance usability and accessibility for librarians and administrators. Though the system is currently designed for single-user offline use, it serves as a strong foundation for future enhancements like multi-user support, user authentication, and cloud-based integration.**

**Overall, this project demonstrates how simple tools and effective design principles can lead to a practical and scalable solution for library resource management.**

**End of Document**