# Jaypee Institute of Information Technology, Sector - 62, Noida

**B.Tech CSE I Semester**


# SDF PBL Report

## Library Management System

**Submitted to**

Dr. Shardha Porwal and Dr. Kavita Pandey

**Submitted by**


| | |
|---|---|
| Mukul Aggarwal | 2401030239 |
| Dhvani Joneja | 2401030240 |
| Ambar Kumar | 2401030223 |

# Letter of Transmittal

**Dr. Shardha Porwal and Dr. Kavita Pandey**

Department of CSE

**Subject:** Submission of Report on "Library Management System"

Dear Dr. Shardha Porwal and Dr. Kavita Pandey,

We are pleased to submit our report titled *"Library Management System"* as part of our coursework. This report explores the technical foundations of memory allocations and memory management in C using ncurses Library. Focusing on touching every concept that has be taught under your supervision.

We have endeavored to cover the practical aspects comprehensively and hope that this report meets your expectations.

Thank you for your guidance and the opportunity to work on this project.

Sincerely,

Mukul Aggarwal (2401030239)
Dhvani Joneja (2401030240)
Ambar Kumar (2401030223)

Date: November 25, 2024

# // CONTENTS

# Introduction:

The **Library Management System (LMS)** is an intuitive, command-line application developed to streamline library operations. Built with **C programming** and the **<ncurses.h>** library for an interactive terminal-based user interface, this system offers an efficient way to manage books, customers, and other essential library functions. The project is the result of the combined efforts of three developers: **Ambar**, **Dhvani**, and **Mukul**, who worked together with enthusiasm and dedication to bring the idea to life.

Throughout the development process, we explored various aspects of software design and functionality while learning the nuances of the <ncurses.h> library to create a user-friendly, text-based interface. Each of us contributed unique skills to make this project both robust and engaging, ensuring that the LMS is easy to use and manage.

**Key Features:**

1. **Add Books to Database**:
    - Easily add new books into the library database with key details like title, author.
2. **Search Book from Database**:
    - Search for books by title, author to find specific book details.
3. **Add Customer to Database**:
    - Add new customers to the system by storing their personal information.
4. **Search Customer from Database**:
    - Search for a customer based on their unique ID or name to retrieve their details.
5. **Delete Customer from Database**:
    - Remove a customer from the database if they are no longer registered or active in the library.
6. **Display Customers from Database**:
    - View a list of all registered customers along with their relevant details.
7. **Set New Password**:
    - Allow administrators to change passwords to secure the system.
8. **Display All Books**:
    - Display a comprehensive list of all available books in the library's database, including their status (borrowed/available).

**Developed by:**

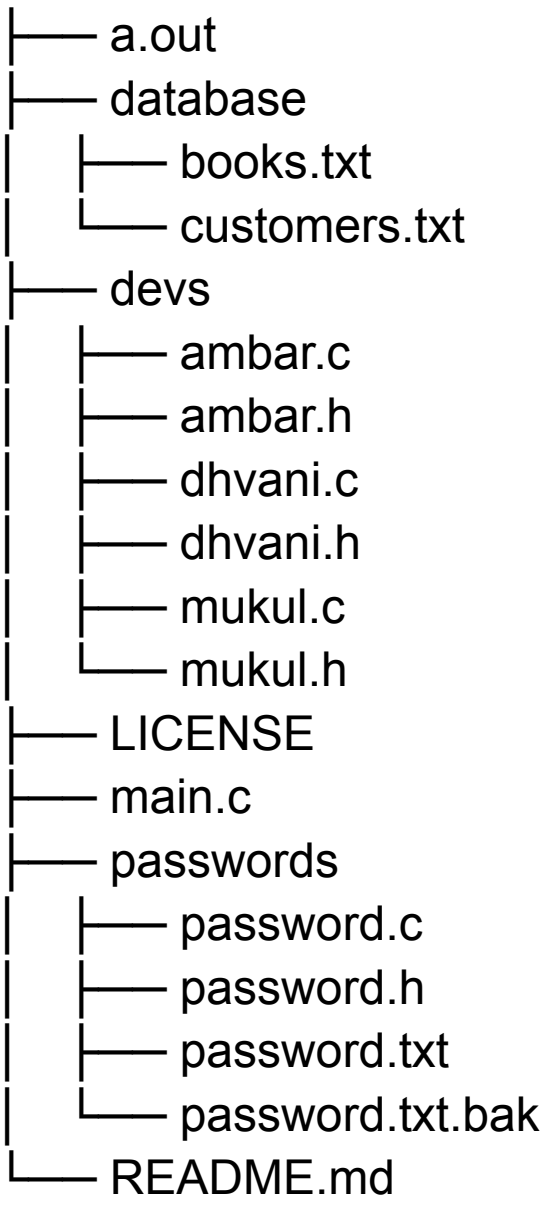
- **Ambar**
- **Mukul**
- **Dhvani**

Together, we enjoyed the journey of designing, coding, and testing each feature of this system, ensuring its functionality and usability. This project not only strengthened our skills in C and UI development but also enhanced our ability to work collaboratively as a team.

**Conclusion:**

The Library Management System is a powerful, simple-to-use solution for managing books and customers in any library. It combines efficient data handling with a user-friendly interface, all while providing essential features that every library needs.

## // FOLDER STRUCTURE



# Project Design

## 1. *Purpose of the Project*

- *Objective:* Create a terminal-based user interface (TUI) for managing a specific task (e.g., a file explorer, process manager, text editor, or a custom dashboard).
- *Key Features:*
    - Multi-panel layout .
    - Keyboard-driven navigation.
    - Dynamic updates to the interface based on user input or external events.

## 2. *Design Components*

- *Modules:*
    - *UI Core:*
        - Handles ncurses initialization and cleanup.
        - Manages screen refreshes and window layouts.
    - *Input Handler:*
        - Captures keyboard events and maps them to actions.
    - *Logic Layer:*
        - Implements application-specific behavior (e.g., browsing files, editing text).
    - *Output Renderer:*
        - Updates windows or panels with content dynamically.
    - *Utilities:*
        - Helper functions for string formatting, color setup, etc.
- *Screen Layout:*
    - *Header:* Displays app name, shortcuts, or system information.
    - *Main Content Area:* Focus area for the primary functionality (e.g., file list, text editor).
    - *Sidebar/Navigation:* Optional, for menus or additional information.
    - *Footer/Status Bar:* Displays current mode, error messages, or help hints.

### 3. *Workflow*

1. *Initialization:*
    - Start ncurses mode.
    - Configure colors, input modes (e.g., raw or cbreak), and layout.
2. *Main Loop:*
    - Wait for user input using getch().
    - Process input and call appropriate handlers.
    - Update the UI accordingly.
3. *Cleanup:*
    - Exit ncurses mode and restore terminal settings.

### 4. *Example ncurses Functionality*

- *Window Management:*
    - Use newwin() to create separate regions for each panel.
    - Refresh individual windows with wrefresh().
- *Color Handling:*
    - Define color pairs with init_pair().
    - Apply colors to text using attron() and attroff().
- *Input Processing:*
    - Map keys (e.g., arrow keys, function keys) to commands like moving a cursor or opening a menu.
- *Dynamic Content:*
    - Update content in response to changes (e.g., scroll a file list).

### 5. *Key Files in Your Project*

- main.c (or main.cpp): Contains the entry point and initializes the UI.
- ui.c / ui.h: Manages ncurses setup, window creation, and layout.
- input.c / input.h: Processes user inputs and routes commands.
- logic.c / logic.h: Implements application-specific operations.
- utils.c / utils.h: Contains helper functions.

### 6. *Enhancements*

- *Configuration:* Load themes or key bindings from a config file.
- *Scrolling:* Implement smooth scrolling for content-heavy sections.
- *File I/O:* If working with files, integrate fopen() and other C I/O functions.
- *Extensibility:* Modularize components to support new features easily.

### Example Use Case: File Explorer Design

- *Header:* Current directory path.
- *Main Area:* File and directory listing.
- *Sidebar:* Optional preview of a selected file.
- *Footer:* Command shortcuts (e.g., q to quit, o to open)

---

# BIBLIOGRAPHY

1) documentation from google
2) documentation from google
3) C structures and file handling from geeksforgeeks.

**The project is on github.. Please star our repo**

https://github.com/me-is-mukul/Library_Management_JIIT