

Library management System

INCREMENTAL SDLC MODEL



CSC702 SOFTWARE ENGINEERING project

Submitted to: Dr. oishila bandopadhyay

Chandni Jha [CSE21030/690] | Shubham Kumar Jha [CSE/21082/742] | PURNAMRITA BHATTACHARYA [CSE/21064/724]

# FEATURES

The core functionalities of the Library Management System can be divided into **Admin** and **Student** modules. Here’s a breakdown:

**1. Admin Functionality**

* **Add Books**: Admins can add new books to the library, specifying details such as the book ID, title, author, genre, and the number of copies available.
* **View Borrowed Books**: Allows the admin to see all books currently borrowed by students, along with details like the student's name and borrow date.
* **Show All Books**: View a list of all books in the library, including available copies for each.
* **View Flagged Students**: Admins can view a list of students who are flagged due to late returns. These students are restricted from borrowing more books until cleared.

**2. Student Functionality**

* **Borrow Books**:
  + Select books by genre.
  + Borrow up to a maximum of three books at any given time.
  + Borrow a book only if copies are available, and only one copy per book can be borrowed by the same student.
* **Return Books**:
  + Students can return borrowed books.
  + The system checks if the book is returned within a week; if overdue, the student is flagged and restricted from further borrowing until the fine is cleared.
* **Show All Books**: Students can view a list of all books in the library, with details about genres and available copies.

**3. Common Functionalities**

* **User Type Selection**: Ability to select the user type (Admin or Student) to access specific functionalities.
* **Flagging System**: Automatic flagging of students who return books late. Flagged students cannot borrow more books until cleared by the admin.
* **Data Persistence**: The application uses session state to store data about books, borrowed books, and flagged students, ensuring data continuity during the session.

These core functionalities provide a comprehensive experience for both students and admin, allowing smooth operations for book borrowing, returning, and tracking library assets and user compliance.

# CODE [[Github](https://github.com/ChandniJha630/Library-Management-System/blob/main/index.py)]

import streamlit as st

import pandas as pd

from datetime import datetime, timedelta

# Initialize data

if "books" not in st.session\_state:

st.session\_state["books"] = pd.DataFrame(columns=["Book ID", "Title", "Author", "Genre", "Available Copies", "Total Copies"])

if "borrowed\_books" not in st.session\_state:

st.session\_state["borrowed\_books"] = pd.DataFrame(columns=["Book ID", "Title", "Student Name", "Borrow Date", "Returned"])

if "students\_flagged" not in st.session\_state:

st.session\_state["students\_flagged"] = set()

# Admin functionalities

def admin():

st.title("Admin - Library Management System")

choice = st.sidebar.selectbox("Choose Option", ["Add Book", "View Borrowed Books", "Show All Books", "Flagged Students"])

if choice == "Add Book":

with st.form("add\_book\_form"):

book\_id = st.text\_input("Book ID")

title = st.text\_input("Title")

author = st.text\_input("Author")

genre = st.selectbox("Genre", ["Romantic", "Comedy", "Scientific"])

total\_copies = 3

submitted = st.form\_submit\_button("Add Book")

if submitted:

new\_book = {

"Book ID": book\_id,

"Title": title,

"Author": author,

"Genre": genre,

"Available Copies": total\_copies,

"Total Copies": total\_copies

}

st.session\_state.books = st.session\_state.books.append(new\_book, ignore\_index=True)

st.success("Book added successfully!")

elif choice == "View Borrowed Books":

st.write("List of Borrowed Books")

st.write(st.session\_state.borrowed\_books[st.session\_state.borrowed\_books["Returned"] == False])

elif choice == "Show All Books":

st.write("All Books in the Library")

st.write(st.session\_state.books)

elif choice == "Flagged Students":

st.write("Flagged Students")

if st.session\_state.students\_flagged:

st.write(list(st.session\_state.students\_flagged))

else:

st.write("No flagged students currently.")

# Student functionalities

def student():

st.title("Student - Library Management System")

choice = st.sidebar.selectbox("Choose Option", ["Borrow Book", "Return Book", "Show All Books"])

student\_name = st.text\_input("Enter Your Name")

# Check if student is flagged

if student\_name in st.session\_state.students\_flagged:

st.warning("You are flagged and cannot borrow books until your fine is cleared.")

return

borrowed\_books\_count = st.session\_state.borrowed\_books[(st.session\_state.borrowed\_books["Student Name"] == student\_name) & (st.session\_state.borrowed\_books["Returned"] == False)].shape[0]

if choice == "Borrow Book":

if borrowed\_books\_count >= 3:

st.warning("You have already borrowed the maximum number of books (3).")

else:

genre = st.selectbox("Choose Genre", ["Romantic", "Comedy", "Scientific"])

available\_books = st.session\_state.books[(st.session\_state.books["Genre"] == genre) & (st.session\_state.books["Available Copies"] > 0)]

if not available\_books.empty:

book\_id = st.selectbox("Select Book", available\_books["Book ID"].tolist())

if book\_id in st.session\_state.borrowed\_books[(st.session\_state.borrowed\_books["Student Name"] == student\_name) & (st.session\_state.borrowed\_books["Returned"] == False)]["Book ID"].tolist():

st.warning("You already borrowed a copy of this book.")

else:

with st.form("borrow\_book\_form"):

borrow\_confirm = st.form\_submit\_button("Borrow Book")

if borrow\_confirm:

st.session\_state.books.loc[st.session\_state.books["Book ID"] == book\_id, "Available Copies"] -= 1

book\_info = st.session\_state.books[st.session\_state.books["Book ID"] == book\_id].iloc[0]

borrow\_date = datetime.now()

borrowed\_book = {

"Book ID": book\_id,

"Title": book\_info["Title"],

"Student Name": student\_name,

"Borrow Date": borrow\_date,

"Returned": False

}

st.session\_state.borrowed\_books = st.session\_state.borrowed\_books.append(borrowed\_book, ignore\_index=True)

st.success(f"{book\_info['Title']} borrowed successfully!")

else:

st.warning("No books available in this genre.")

elif choice == "Return Book":

student\_borrowed\_books = st.session\_state.borrowed\_books[

(st.session\_state.borrowed\_books["Student Name"] == student\_name) &

(st.session\_state.borrowed\_books["Returned"] == False)

]

if not student\_borrowed\_books.empty:

book\_id = st.selectbox("Select Book", student\_borrowed\_books["Book ID"].tolist())

with st.form("return\_book\_form"):

return\_confirm = st.form\_submit\_button("Return Book")

if return\_confirm:

borrow\_info = student\_borrowed\_books[student\_borrowed\_books["Book ID"] == book\_id].iloc[0]

borrow\_date = borrow\_info["Borrow Date"]

st.session\_state.books.loc[st.session\_state.books["Book ID"] == book\_id, "Available Copies"] += 1

# Check if the book is returned after 7 days

if (datetime.now() - borrow\_date).days > 7:

st.session\_state.students\_flagged.add(student\_name)

st.warning("You are flagged due to late return.")

else:

st.success(f"{borrow\_info['Title']} returned successfully!")

# Update the borrowed books record

st.session\_state.borrowed\_books.loc[

(st.session\_state.borrowed\_books["Book ID"] == book\_id) &

(st.session\_state.borrowed\_books["Student Name"] == student\_name),

"Returned"] = True

else:

st.warning("You have no borrowed books to return.")

elif choice == "Show All Books":

st.write("All Books in the Library")

st.write(st.session\_state.books)

# Main function to control the flow

def main():

st.sidebar.title("Library Management System")

user\_type = st.sidebar.selectbox("Select User Type", ["Admin", "Student"])

if user\_type == "Admin":

admin()

else:

student()

if \_\_name\_\_ == "\_\_main\_\_":

main()

# SDLC Model Overview

The **Incremental SDLC model** is best suited for this Library Management System project. Here’s how it applies:

**1. Incremental Model Overview**

* This model involves breaking down the project into smaller, manageable modules or "increments." Each increment adds specific features or functionality, and each one is tested and validated before moving on to the next.
* In each increment, the code is designed, implemented, and tested, allowing for faster feedback and adjustments based on any user input or issues identified in previous increments.

**2. Application to the Library Management System Project**

* **Increment 1: Basic Book Management**
  + **Feature**: Implement the core functionality for managing books.
  + **Tasks**: Set up Streamlit, initialize book data, and create a user interface for the admin to add books, view book inventory, and display all available books.
* **Increment 2: Borrowing and Returning Books**
  + **Feature**: Add borrowing and returning functionality.
  + **Tasks**: Develop features that allow students to borrow and return books. Implement checks for book availability, enforce the limit on the number of borrowed books, and update the available copies upon return.
* **Increment 3: Late Return Penalty and Flagging**
  + **Feature**: Add late return tracking.
  + **Tasks**: Extend the student module to track late returns, flag students who exceed the borrowing period, and prevent flagged students from borrowing until cleared.
* **Increment 4: User Types and Final Testing**
  + **Feature**: Introduce user roles.
  + **Tasks**: Enable user role selection for students and admin. Conduct end-to-end testing for each module to ensure the system functions as expected.

**3. Benefits of the Incremental Model for This Project**

* **Adaptability**: Allows for changes and additional features, like the login system and flagging feature, to be incorporated gradually.
* **Early Feedback**: Stakeholders can review each increment and suggest changes or improvements early on, preventing larger issues from surfacing later.
* **Reduced Risk**: Each increment is tested independently, ensuring that each functionality is stable before adding new ones.
* **User Satisfaction**: Key features can be delivered and demonstrated sooner, providing visible progress that can be showcased to stakeholders and end users.

Overall, the **Incremental Model** provides a structured approach with flexibility, making it easier to adjust and improve the Library Management System as new requirements emerge.