CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF COMPUTING

MINI PROJECT REPORT

Program Name: BCA

Subject Name/Code:

Computing Aptitude/23CAP 308

Submitted by:

Name: Manjeet Singh / Abhishek Anshu

UID: 23BCA10309 / 23BCA10687

Section: 23BCA-8'B'

Submitted to:

Name: Shilpa Mahajan

Designation: Asst Prof.

Library Recommendation System

Project Description

The Library Recommendation System is a console-based C++ application that simulates an intelligent book suggestion engine for a library setting. The primary objective of this project is to recommend new books to a user in a personalized manner, based on their reading preferences and history. The system utilizes structured data storage, user-driven input collection, and algorithmic filtering to provide tailored book recommendations—mirroring real-world digital library experiences.

Objective

- Create a simple digital recommendation tool for libraries, allowing users to:
    - Specify genres they prefer (e.g., Fantasy, Sci-Fi, Thriller).
    - Indicate books they have already read.
    - Receive curated book suggestions based on both their genre interests and previous selections.

Functional Overview

- **Data Organization:**
  A built-in database contains sample books, each with a unique ID, title, author, and genre.
- **User Data:**
  The user is prompted to enter their name, select genres of interest, and identify previously read titles by book ID.
- **Recommendation Logic:**
  The algorithm recommends books:
  - Only from genres the user likes.
  - Excluding those already read (as indicated by their book ID).
- **Result Output:**
  The program displays a personalized list of book suggestions or guides the user to broaden their interests if no matches are found.

Code Explanation

- **Structures Used:**

    - struct Book — to encapsulate book details.
    - struct User — to store username, preferred genres (as a set), and reading history (by book IDs).
- **Key Functions:**
    - loadBookDatabase() — Initializes the in-memory book catalog.
    - getUserPreferences() — Collects and stores user's genre preferences.
    - getUserHistory() — Flags which books the user has already read.
    - getRecommendations() — Filters and creates a recommendation list based on preferences and history.
    - printRecommendations() — Cleanly displays recommendations or prompts user for more input if necessary.
- **User Flow:**

1. Welcome + name entry.
2. Genre preferences collection.

3. Full library catalog display.
4. Reading history collection.
5. Generation and display of recommendations.

Code:

```
#include <iostream>
#include <vector>
#include <string>
#include <map>
#include <set>
#include <sstream>
#include <limits>
struct Book {
    int id;
    std::string title;
    std::string author;
    std::string genre;
};
struct User {
```

```
    std::string username;
    std::set<std::string> preferredGenres; // Genres the user likes
    std::set<int> previousSelections;      // Book IDs the user has already selected
};
std::map<int, Book> loadBookDatabase() {
    std::map<int, Book> database;
    database[101] = {101, "Dune", "Frank Herbert", "Sci-Fi"};
    database[102] = {102, "The Hobbit", "J.R.R. Tolkien", "Fantasy"};
    database[103] = {103, "1984", "George Orwell", "Dystopian"};
    database[104] = {104, "Foundation", "Isaac Asimov", "Sci-Fi"};
    database[105] = {105, "The Name of the Wind", "Patrick Rothfuss", "Fantasy"};
    database[106] = {106, "Brave New World", "Aldous Huxley", "Dystopian"};
```

```
    database[107] = {107, "A Game of Thrones", "George R.R. Martin", "Fantasy"};
    database[108] = {108, "Ender's Game", "Orson Scott Card", "Sci-Fi"};
    database[109] = {109, "The Da Vinci Code", "Dan Brown", "Thriller"};
    database[110] = {110, "Gone Girl", "Gillian Flynn", "Thriller"};
    return database;
}
void clearInputBuffer() {

std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
}
void printAllBooks(const std::map<int, Book>& allBooks) {
    std::cout << "\n--- Full Library Catalog ---" << std::endl;
    for (const auto& pair : allBooks) {
```

```
        const Book& book = pair.second;
        std::cout << "  ID: " << book.id << " | "
                  << book.title << " (" << book.genre <<
")" << std::endl;
    }
    std::cout << "---------------------------" <<
std::endl;
}
void getUserPreferences(User& user) {
    std::string genre;
    std::cout << "\n--- Genre Preferences ---" <<
std::endl;
    std::cout << "Enter your preferred genres, one
at a time (e.g., Sci-Fi, Fantasy, Thriller)." <<
std::endl;
    std::cout << "Type 'done' when you are
finished." << std::endl;

    while (true) {
        std::cout << "Add genre: ";
```

```
        std::getline(std::cin, genre);
          if (genre == "done") {
            break;
        }
        if (!genre.empty()) {
            // Simple normalization: capitalize first
letter
            genre[0] = toupper(genre[0]);
            user.preferredGenres.insert(genre);
            std::cout << "  Added '" << genre << "'." <<
std::endl;
        }
    }
}
void getUserHistory(User& user, const
std::map<int, Book>& allBooks) {
    int id;
    std::string input;
    std::cout << "\n--- Reading History ---" <<
std::endl;
```

```
    std::cout << "Enter the ID of any books you have already read." << std::endl;
    std::cout << "Type 'done' when you are finished." << std::endl;
    while (true) {
        std::cout << "Enter Book ID (or 'done'): ";
        std::getline(std::cin, input);
        if (input == "done") {
            break;
        }
        std::stringstream ss(input);
        if (ss >> id) {
            if (allBooks.count(id)) {
                user.previousSelections.insert(id);
                std::cout << "  Added '" << allBooks.at(id).title << "' to your history." << std::endl;
            } else {
                std::cout << "  Error: No book found with ID " << id << "." << std::endl;
```

```
            }
        } else if (!input.empty()) {
            std::cout << "  Error: Please enter a valid number or 'done'." << std::endl;
        }
    }
}
std::vector<Book> getRecommendations(const User& user, const std::map<int, Book>& allBooks) {
    std::vector<Book> recommendations;
    for (const auto& pair : allBooks) {
        const Book& book = pair.second;
        // 1. Check if the book's genre is one of the user's preferred genres.
        bool genreMatch = (user.preferredGenres.find(book.genre) != user.preferredGenres.end());
        // 2. Check if the user has NOT selected this book before.
```

```
        bool notSelected =
(user.previousSelections.find(book.id) ==
user.previousSelections.end());
        // 3. If it's a genre they like AND they haven't
selected it, recommend it!
        if (genreMatch && notSelected) {
            recommendations.push_back(book);
        }
    }
    return recommendations;
}
void printRecommendations(const
std::vector<Book>& recommendations) {
    std::cout <<
"\n==================================
====" << std::endl;
    std::cout << "   Your Personalized
Recommendations" << std::endl;
    std::cout <<
"==================================
==" << std::endl;
```

```
    if (recommendations.empty()) {
        std::cout << "No new recommendations found based on your preferences and history." << std::endl;
        std::cout << "Try adding more genres!" << std::endl;
        return;
    }
    for (const auto& book : recommendations) {
        std::cout << "  Title: " << book.title << std::endl;
        std::cout << "  Author: " << book.author << std::endl;
        std::cout << "  Genre: " << book.genre << std::endl;
        std::cout << "---------------------------------------" << std::endl;
    }
}
int main() {
```

```
    std::map<int, Book> allBooks =
loadBookDatabase();
    User currentUser;
    std::cout << "Welcome to the Library
Recommendation System!" << std::endl;
    std::cout << "Please enter your name: ";
    std::getline(std::cin, currentUser.username);
    getUserPreferences(currentUser);
    printAllBooks(allBooks);
    getUserHistory(currentUser, allBooks);
    std::vector<Book> recommendations =
getRecommendations(currentUser, allBooks);
    printRecommendations(recommendations);
    return 0;
}
```

Output:

```
Welcome to the Library Recommendation System!
Please enter your name: Aryan

--- Genre Preferences ---
Enter your preferred genres, one at a time (e.g., Sci-Fi, Fantasy, Thriller).
Type 'done' when you are finished.
Add genre: Sci-Fi
  Added 'Sci-Fi'.
Add genre: Thriller
  Added 'Thriller'.
Add genre: done

--- Full Library Catalog ---
  ID: 101 | Dune (Sci-Fi)
  ID: 102 | The Hobbit (Fantasy)
  ID: 103 | 1984 (Dystopian)
  ID: 104 | Foundation (Sci-Fi)
  ID: 105 | The Name of the Wind (Fantasy)
  ID: 106 | Brave New World (Dystopian)
  ID: 107 | A Game of Thrones (Fantasy)
  ID: 108 | Ender's Game (Sci-Fi)
  ID: 109 | The Da Vinci Code (Thriller)
  ID: 110 | Gone Girl (Thriller)
----------------------------

--- Reading History ---
Enter the ID of any books you have already read.
Type 'done' when you are finished.
Enter Book ID (or 'done'): 101
  Added 'Dune' to your history.
Enter Book ID (or 'done'): 104
  Added 'Foundation' to your history.
Enter Book ID (or 'done'): done

========================================
   Your Personalized Recommendations
========================================
  Title: Ender's Game
  Author: Orson Scott Card
  Genre: Sci-Fi
----------------------------------------
  Title: The Da Vinci Code
```