WEEK 11 OVERVIEW

WEEK 11 — FULL DOCUMENTATION
Smart Library Management & Recommendation System (C++)

This document includes:

USE-CASE DESCRIPTIONS

USE-CASE DESCRIPTIONS

Actor: User
1. Search Book — User enters ISBN, system displays match.
2. Borrow Book — System checks availability, updates copies or adds to waitlist.
3. Return Book — System notifies next user in waitlist or increases availableCopies.
4. View History — Retrieves user’s LinkedList of borrowing history.
5. Get Recommendations — (Week 12) System finds connected books in graph.

Actor: Admin
6. Add Book — Admin enters ISBN, title, copies.
7. Update Book — Admin modifies fields.
8. Remove Book — Book deleted or marked unavailable.
9. View Reports — System displays popularity rankings and overdue books.

UML CLASS DIAGRAM (TEXT)

UML CLASS REPRESENTATION (TEXT FORMAT)

```
+---------------------+
|       Book          |
+---------------------+
| isbn : string       |
| title : string      |
| author : string     |
| totalCopies : int   |
| availableCopies: int|
| waitlist : queue<int> |
| popularityCount : int |
+---------------------+

+---------------------+
|       User          |
+---------------------+
| userId : int        |
| name : string       |
| borrowedCount : int |
| history : LinkedList<string> |
+---------------------+

+---------------------+
|    LinkedList<T>    |
+---------------------+
| head : Node<T>*     |
+---------------------+
| insert(T)           |
| print()             |
+---------------------+

+---------------------+
|     HashTable       |
+---------------------+
| table : Book[]      |
| capacity : int      |
| size : int          |
+---------------------+
| hashFunc()          |
| resize()            |
| insert(Book)        |
| search(isbn)        |
+---------------------+

+---------------------------+
|      BorrowEngine         |
+---------------------------+
| borrowBook()              |
| returnBook()              |
+---------------------------+
```

DATA STRUCTURE SPECIFICATIONS

DATA STRUCTURE SPECIFICATIONS

1. Hash Table (Custom)
   - Open addressing
   - Linear probing
   - Dynamic resizing
   - Average O(1) search

2. LinkedList<T>
   - Singly linked list
   - Used for user history

3. Arrays
   - User storage: User users[1000]

4. Queue<int>
   - Waitlist per book

5. Graph (Week 12)
   - Adjacency list: int graph[MAX][MAX]

6. Balanced BST (Week 12)
   - Popularity ranking

FILE FORMAT DESIGNS

FILE FORMAT DESIGN

```
books.txt
------------------------------------
ISBN|Title|Author|TotalCopies|AvailableCopies|Popularity
111|C++ Book|Bjarne|3|1|10

users.txt
------------------------------------
UserID|Name|BorrowedCount
1|Aiman|2

history_userID.txt
------------------------------------
ISBN (latest last)
111
222

waitlists.txt
------------------------------------
ISBN|userId1,userId2,userId3
111|2,5,7
```

SAMPLE DATASETS

SAMPLE DATASETS

sample_books.txt:
```
111|C++ Programming|Stroustrup|3|3|0
222|DSA Essentials|Mark|1|1|0
333|Algorithms|CLRS|2|2|0
```

sample_users.txt:
```
1|Aiman|0
2|Ahmed|0
3|Sara|0
```

CORE IMPLEMENTATION SUMMARY

CORE IMPLEMENTATION SUMMARY

Implemented in Week 11:
- Book struct
- User struct
- LinkedList<T>
- HashTable (insert, search, resize)
- UserManager
- BorrowEngine (borrow + return)
- CLI interface (menu-based)

These form the foundation for Week 12 implementation:
- File I/O
- Recommendation graph
- Balanced BST

UNIT TESTS (HashTable + LinkedList)

```
UNIT TESTS — HASH TABLE & LINKED LIST

---------------------------------
TEST 1: HASH TABLE INSERT + SEARCH
---------------------------------
HashTable ht(10);
Book b("111", "C++ Book", "Bjarne", 3);
ht.insert(b);

assert(ht.search("111") != nullptr);
assert(ht.search("111")->title == "C++ Book");

---------------------------------
TEST 2: HASH TABLE RESIZING
---------------------------------
HashTable ht(2);
ht.insert(Book("111", "A", "X", 1));
ht.insert(Book("222", "B", "Y", 1));  // triggers resize
ht.insert(Book("333", "C", "Z", 1));

assert(ht.search("333") != nullptr);

---------------------------------
TEST 3: LINKED LIST INSERTION
---------------------------------
LinkedList<string> history;
history.insert("111");
history.insert("222");

Manually verify:
222 -> 111 -> NULL

---------------------------------
TEST 4: BORROW ENGINE BASIC FLOW
---------------------------------
HashTable catalog;
UserManager users;
BorrowEngine engine(catalog, users);

catalog.insert(Book("111", "Test", "Author", 1));
users.addUser(1, "Aiman");

assert(engine.borrowBook(1, "111") == true);  // book is available
assert(engine.borrowBook(1, "111") == false); // now waitlist
```

WEEK 11 PROGRESS REPORT

WEEK 11 PROGRESS LOG

Completed:
- HashTable implementation (custom)
- Resizing mechanism + testing
- LinkedList for history
- User and Book models created
- BorrowEngine borrow/return logic implemented
- CLI created with options:
  Add User, Add Book, Search Book, Borrow, Return

Next Steps (Week 12):
- Implement file I/O
- Build adjacency list graph for recommendations
- Implement Balanced BST for popularity
- Expand CLI to full feature set