AUC CONNECT PRO

Your guide to a flawless and smooth booking process

Presentated by

Hamdy Osama

Malk Metwally

Radwa Ayman

Sedra Elkhayat

WHY DO YOU NEED AUC CONNECT PRO?

University libraries handle hundreds of reservations every day, study rooms, books, and laptops, all competing for limited time and resources.
But with traditional systems, users can run into problems like:

- Two students booking the same room at the same time
- Borrowing a book or laptop that's already checked out
- A user reserves two rooms at the same time.

HOW OUR PROGRAM STANDS OUT:

Our system uses advanced data-structured conflict detection to ensure no overlaps, no double bookings, and no wasted time.

Powered by an Interval Tree built on a Red-Black Tree, AUC connect pro instantly identifies conflicts, updates schedules efficiently, and scales effortlessly, even with large data.

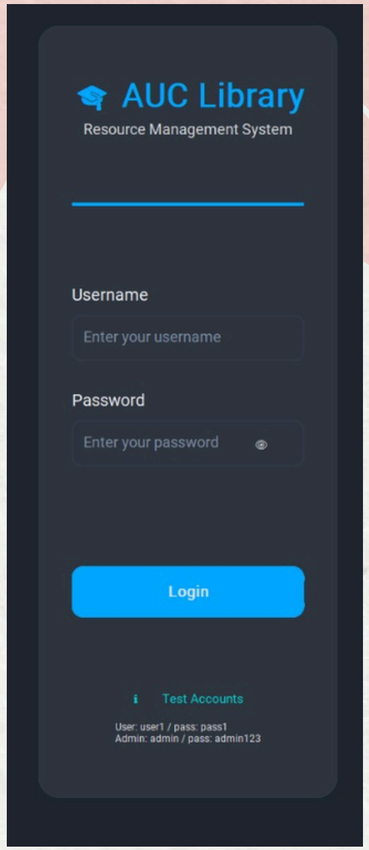
Inspired by the AUC Connect system, AUC connect pro goes beyond it by adding new features like laptop borrowing, book searching, an admin dashboard, and an optional CLI mode.

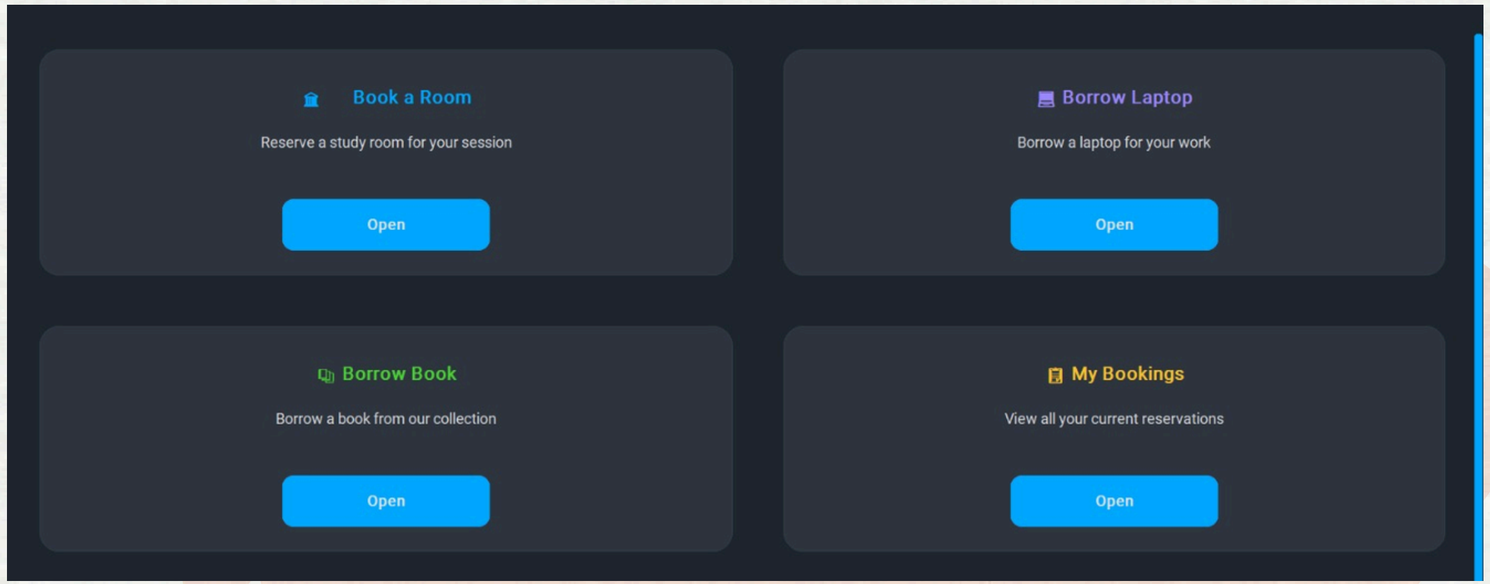
THE FEATURES AVAILABLE

- Login page
- Book a Study Room
- Borrow a book
- Borrow a Laptop
- Admin dashboard
- CLI Mode
- Full GUI Implementation

FEATURES: LOGIN PAGE

Snippet from the GUI:



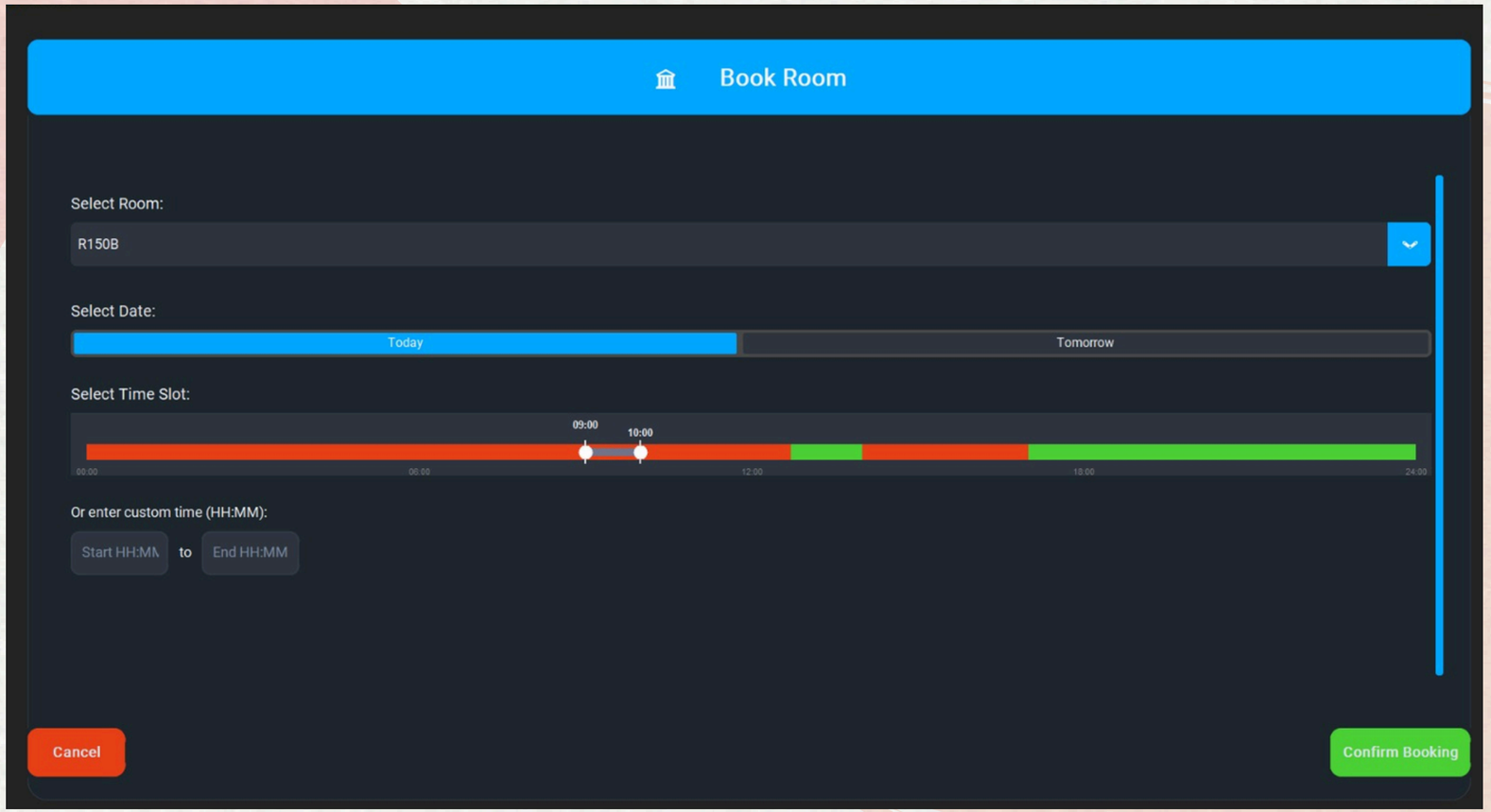


Purpose:

- A clean and secure login system where users enter their username and password to access personalized library services.
- After successful login, the user will have the following options offered by our program.

FEATURES: BOOK A ROOM

- **Snippet from the GUI:**

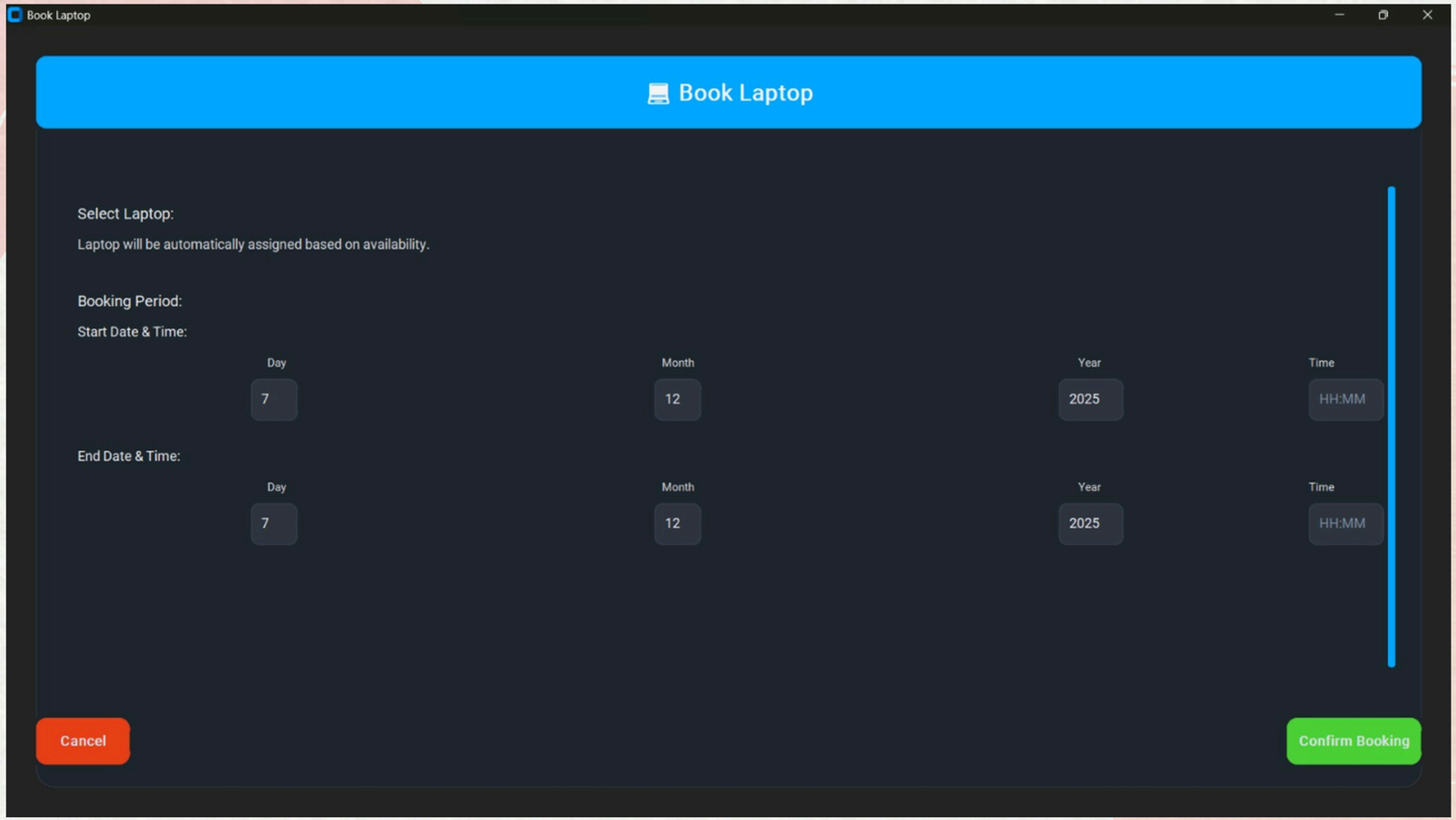


Purpose:

- Students can instantly check room availability and reserve study spaces with conflict-free scheduling.

FEATURES: BORROW A LAPTOP

- Snippet from the GUI:

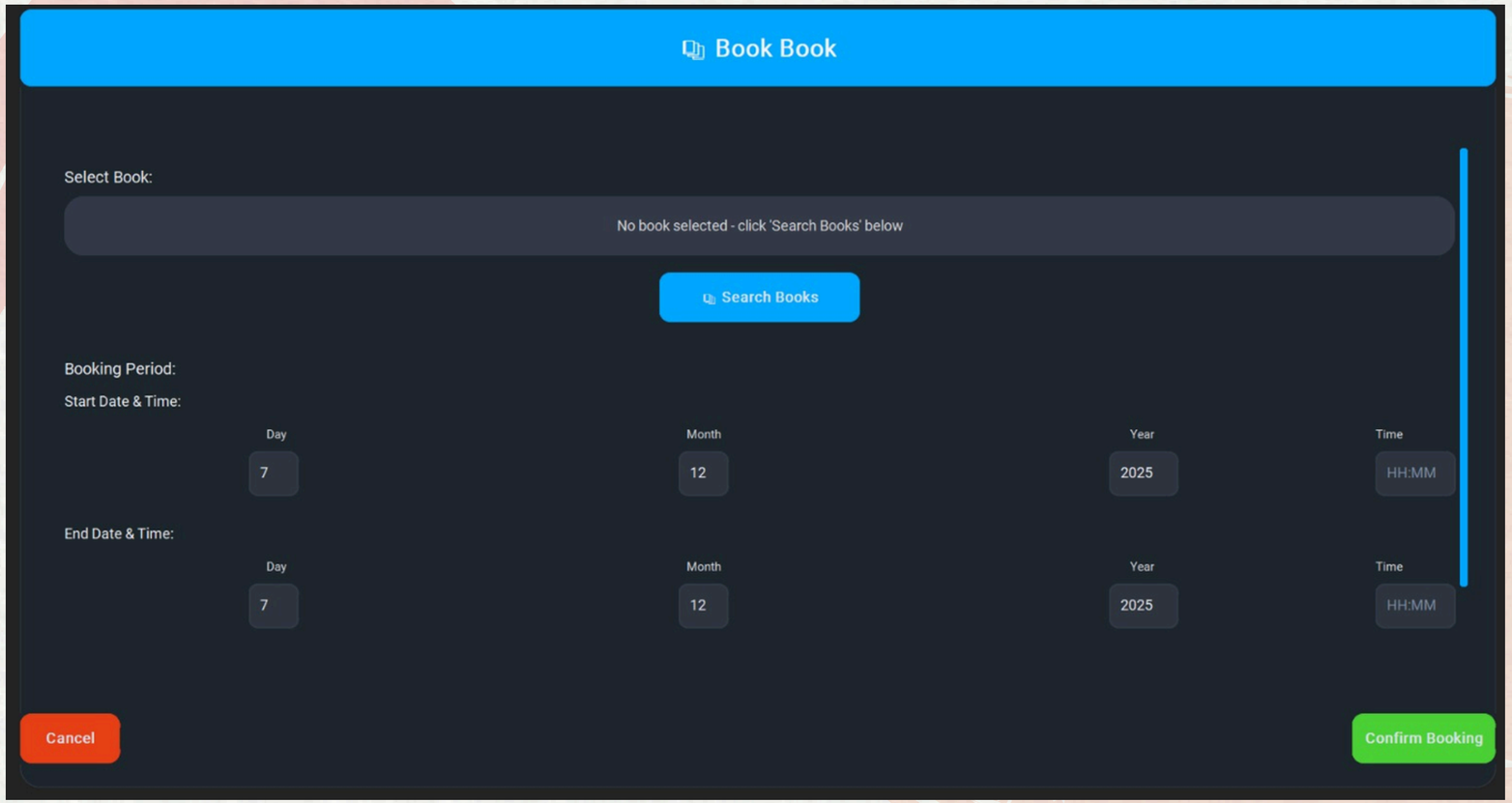


Purpose:

- A brand-new feature not available on AUC Connect, students can borrow laptops based on availability, with real-time conflict checking

FEATURES: BORROW A BOOK

- Snippet from the GUI:

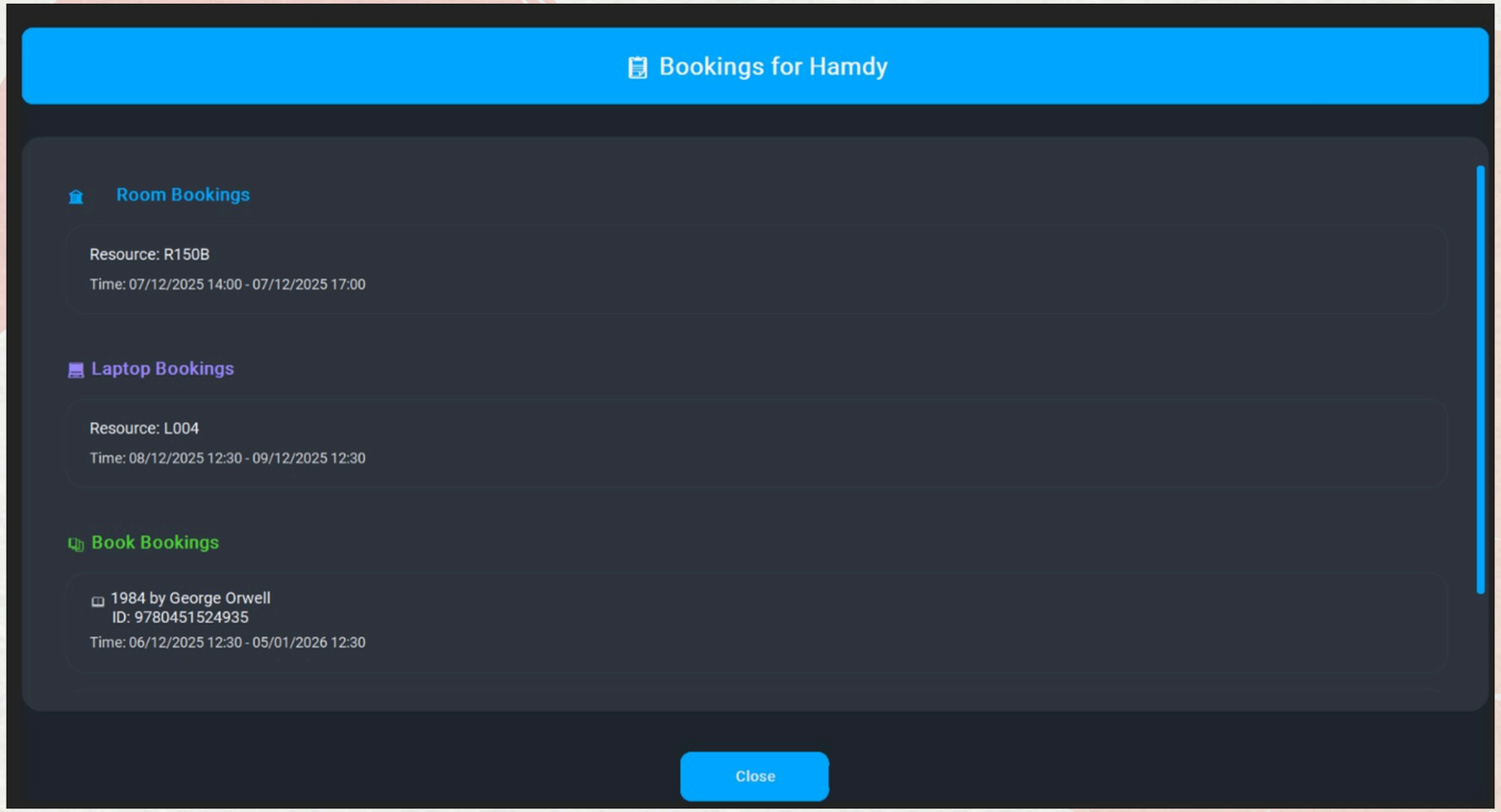


Purpose:

- Search for any book by title or author, check its availability, and borrow it instantly.
- No double borrowing, the system ensures each book is only assigned once.

FEATURES: THE USERS' BOOKINGS

- Snippet from the GUI:

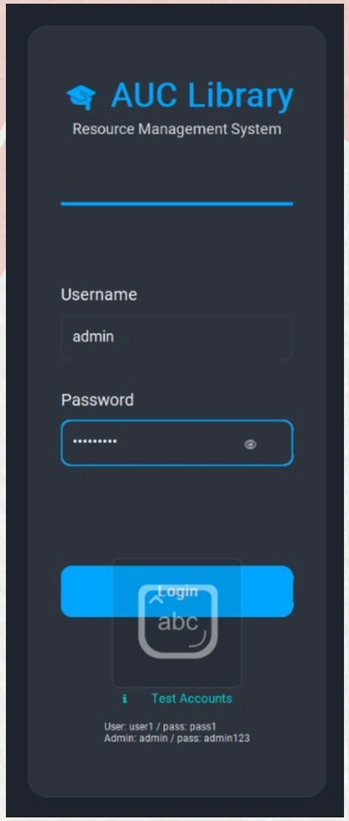


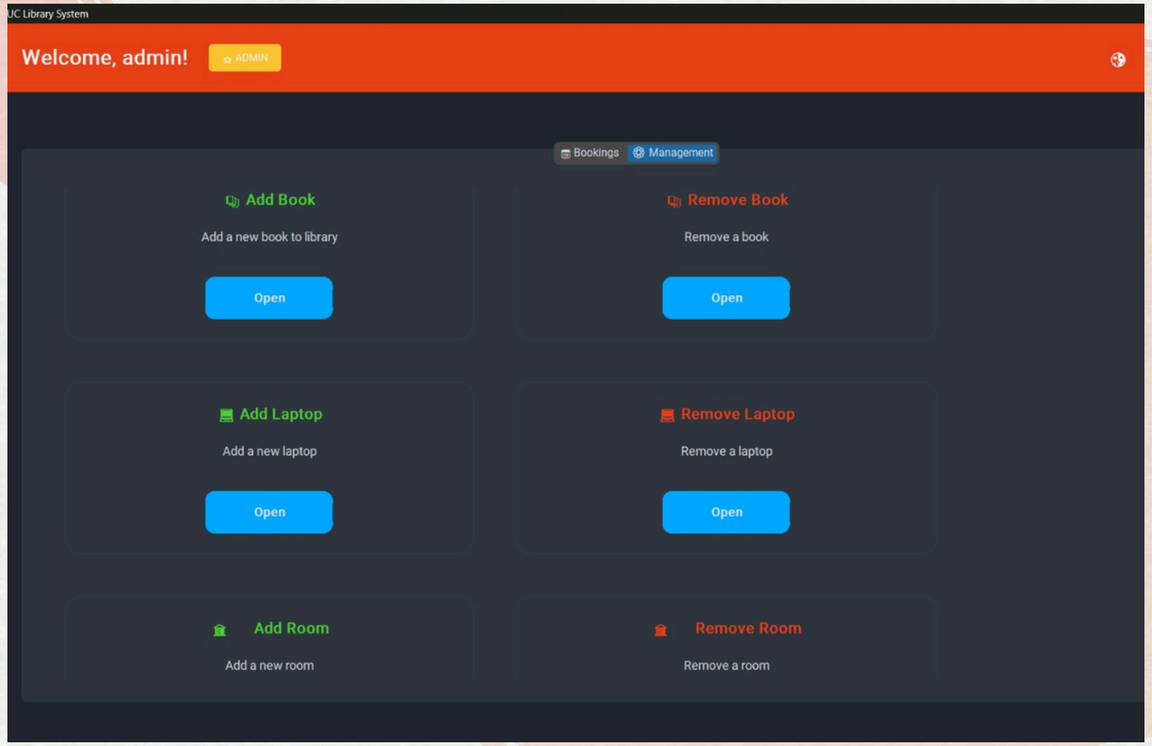
Purpose:

- It shows all the bookings that each user has.

FEATURES: ADMIN DASHBOARD

- **Snippet from the GUI:**





Purpose:

- Any admin can simply interact with the resources in our database by adding or removing
- Books
- Rooms
- Laptops

DATA STRUCTURES USED: HASHMAP

- A HashMap is used to store our data.
- We use separate chaining and dynamic sizing when the load exceeds 75% of the possible capacity
- A Key can be used as a primary key, and shared with another HashMap as a secondary key (for associations)

DATA STRUCTURES USED: INTERVAL TREE

What is a Red-Black Interval Tree?

- A self-balancing Red-Black Tree where each node stores an interval [start, end]
- Each node also stores maxEnd = the largest end time in its entire subtree
- This lets the tree stay balanced (fast insert/delete) and detect overlapping time intervals efficiently

Why We Use It in Our System

- Each resource (room, book, laptop) has its own tree - prevents double-booking
- Each user has their own tree - prevents a user from booking two of the same resource at once

FULL DEMO OF THE APPLICATION

CHALLENGES IN IMPLEMENTATION:

- Improving the time complexity of some of the parts of the code to guarantee a better user experience
- Example
- Changing searching for books from a HashMap with IDs as keys to a HashMap with BookTitle as key
- For search by author:
- Make a hashmap of linked lists with each author linked to its books.

Key (Author)	Value (Linked List*)
"rowling"	→ ["Harry Potter 1"] → ["Harry Potter 2"] ["Fantastic Beasts"]
"asimov"	→ ["Foundation"] → ["I, Robot"]
"orwell"	→ ["1984"]

CHALLENGES IN IMPLEMENTATION:

- Manual testing: Checking for invalid user input data.
- Test cases: Checking that the valid user inputs are handled correctly.

```
ZBook 15 Power G7@Radwa MINGW64 /e/AUC/Fall2025/AppliedDataStructure/ADSProject/
ADS-Project/MalkADS/build (Hamdy-Branch)
$ ctest --output-on-failure
Test project E:/AUC/Fall2025/AppliedDataStructure/ADSProject/ADS-Project/MalkADS
/build
      Start  1: UsersManager login and mock users
 1/11 Test  #1: UsersManager login and mock users ...................   Passed
  0.11 sec
      Start  2: UsersManager admin and normal user
 2/11 Test  #2: UsersManager admin and normal user ..................   Passed
  0.01 sec
      Start  3: LaptopsManager add and remove laptop
 3/11 Test  #3: LaptopsManager add and remove laptop ................   Passed
  0.01 sec
      Start  4: LaptopsManager borrowLaptopDirect
 4/11 Test  #4: LaptopsManager borrowLaptopDirect ...................   Passed
  0.01 sec
      Start  5: LaptopsManager overlapping bookings for same user
 5/11 Test  #5: LaptopsManager overlapping bookings for same user ...   Passed
  0.01 sec
      Start  6: RoomsManager add and book room with mock user
 6/11 Test  #6: RoomsManager add and book room with mock user .......   Passed
  0.01 sec
      Start  7: RoomsManager remove room
 7/11 Test  #7: RoomsManager remove room ............................   Passed
  0.01 sec
      Start  8: Booking conflict for mock user
 8/11 Test  #8: Booking conflict for mock user ......................   Passed
  0.01 sec
      Start  9: Add and get book
 9/11 Test  #9: Add and get book ....................................   Passed
  0.01 sec
      Start 10: Remove book
10/11 Test #10: Remove book .........................................   Passed
  0.01 sec
      Start 11: Borrow book non-interactive
11/11 Test #11: Borrow book non-interactive .........................   Passed
  0.01 sec

100% tests passed, 0 tests failed out of 11

Total Test time (real) =   0.41 sec
```

LIMITATIONS AND FUTURE WORKS:

Limitation - Priority Queue
The laptop priority feature could not be fully implemented because laptops are stored only by ID, not as full objects with usage data.

Future Improvement – Priority Queue
Implement a min-heap–based priority queue to always give users the newest or least-used laptops.

Limitation – Queue
Using a queue to process multiple user commands was removed because it created confusing error handling and a poor user experience.

Future Improvement – Queue
A queue could later be used for user notifications/history, with more efficient resizing (e.g., ×1.1 instead of ×2).

LIMITATIONS AND FUTURE WORKS:

Future Improvement – HashMap
For large datasets, the system should migrate from HashMaps to SQL/NoSQL databases for scalability.

Future Improvement – Interval Tree
Store interval data in a database and expand program to support multi-user access

Future Improvement – Cancellations
Add a cancellation feature so users can free booked laptops and improve resource utilization.

Future Improvement – Input Validation
Improve CLI and GUI input validation to prevent invalid dates, incorrect time frames, or accidental booking errors.

THANK YOU!