**MANGALORE INSTITUTE OF TECHNOLOGY & ENGINEERING**

*Accredited by NAAC with A+ Grade, An ISO 9001: 2015 Certified Institution*

*(A Unit of Rajalaxmi Education Trust®, Mangalore - 575001)*

*Affiliated to V.T.U., Belagavi, Approved by AICTE, New Delhi*



**TECHNICAL TRAINING PROJECT**

**Group-CP059**

**TOPIC : RESOURCE MANAGEMENT AND BOOKING**

**GROUP MEMBERS:**

|  |  |
| --- | --- |
| Sai ManoDileep | 4MT21AI041 |
| Sameeksha HS | 4MT21AI043 |
| Savi Sanjiv | 4MT21AI047 |
| Ganesh SP | 4MT21AI019 |
| Varun MC | 4MT21AI062 |

**TABLE OF CONTENTS**

**Problem Statement**

**1.Abstract**

**2.Introduction**

2.1 Background

2.2 Objectives

**3.Technologies Used**

**4.System Architecture**

4.1 Front-End

4.2 Back-End

4.3 Database

**5.Project Modules** :

**6.Design and Implementation**

6.1 Front-End Design

6.2 Back-End Design

6.3 Database Design

**7.Features and Functionality**

**8.Testing**

## 8.1 User Registration Testing:

8.2 Performance Testing

## 8.3 Compatibility Testing:

**9.Challenges Faced**

**10.Future Enhancements**

**11.Conclusion**

**12.References**

**13.Appendices**

**13.1 Screenshots**

# 13.2 Code Snippets

**Problem Statement:**

Design and develop a Resource Booking and Management System (RBMS) that allows organizations to efficiently book, manage, and optimize the utilization of their resources. The RBMS should address the following aspects:

# Introduction:

# In today's dynamic and evolving business landscape, the effective management and allocation of resources are paramount for organizations striving to maintain productivity, reduce costs, and enhance the overall user experience. Resource Booking and Management systems have emerged as essential tools to address these challenges, offering streamlined solutions for scheduling, reserving, and optimizing a wide range of valuable assets.

**2.1 Backround:** The background for the concept of Resource Booking and Management stems from the growing need for organizations to optimize their resources efficiently in today's fast-paced and dynamic business environments. As organizations expand and evolve, the effective allocation and utilization of resources such as meeting spaces, equipment, and facilities have become paramount for productivity, cost management, and providing seamless experiences for employees and stakeholders.

## 2.2 Objectives:

* Efficient Resource Booking:
* Resource Optimization:
* Streamlined Booking Process:
* User Accountability:

## 3. Technologies Used :

* Programming Language: C
* Standard Libraries: `<stdio.h>`, `<stdbool.h>,`<string.h>`

## 4. System Architecture 4.1 Front-End

The front-end of this project is text-based and relies on user input and output through the command line.

**4.2 Back-End**

The back-end manages menu navigation, and element manipulation.

**4.3 Database**

The system does not use a database as it stores user and element data in memory.

## 5. Project Modules:

Resource Management Module:

Booking and Reservation Module:

Notification and Reminder Module:

## 6. Design and Implementation

**6.1 Front-End Design**

* The front-end design is minimalistic and relies on text-based input and output.

**6.2 Back-End Design**

* The back-end handles menu selection, and element manipulation.

**6.3 Database Design**

* The system does not use a database.

## 7. Features and Functionality

**User Registration**:

Allow users to sign up by providing necessary information (name, pupose etc.).

**User Authentication:**

Implement secure login mechanisms with password hashing and salting.

No Support login via social media accounts

**User Roles and Permissions:**

Define roles (e.g., regular user, administrator, resource manager) with specific permissions.

Assign users to roles during registration or by administrators.

## 8. Functional Testing:

## User Registration Testing:

## User Authentication Testing:

## Performance Testing:

## Load Testing:

## booking Testing:

**9. Challenges Faced**

* Challenges included handling user input and implementing random algorithms.
* Performance Optimization
* User Interface Design

## 10. Future Enhancements

- Future enhancements could include data persistence with file storage and a more user-friendly interface.

**RESOURCE MANAGEMENT AND BOOKING:**

**Description:**

Resource Booking and Management is a critical aspect of organizational operations, whether in corporate environments, educational institutions, or other settings. It involves the implementation of a system or process that allows users to book and manage resources according to their needs while ensuring optimal utilization.

**Key Components:**

**Resource Catalog:**

A centralized catalog or database that lists all available resources, including their attributes (e.g., capacity, location, availability).

**Booking System:**

A mechanism that enables users to search for, reserve, and manage resources. This system typically includes features like booking confirmation, modification, and cancellation.

**User Management:**

User registration, authentication, and role-based access control to determine who can book and manage resources.

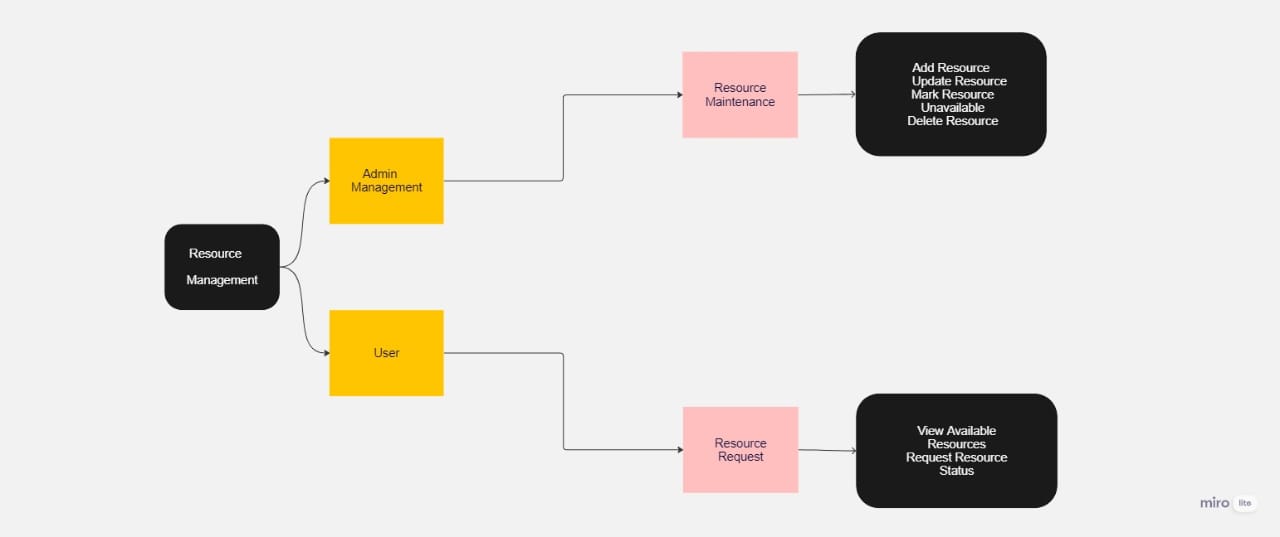
**Notifications:**

Automated notifications and reminders sent to users about their bookings, ensuring they don't miss scheduled appointments.

**Reporting and Analytics:**

Tools and features that provide insights into resource utilization, occupancy rates, and booking patterns, aiding in decision-making.

MIND MAP:



**PROGRAM CODE:**

#include <stdio.h>

#include <stdbool.h>

#include <string.h>

#define MAX\_LABS 5

#define MAX\_ROOMS 10

#define MAX\_BOOKINGS 10

struct Booking {

int RoomNumber;

int labNumber;

char purpose[100];

char userName[50];

};

struct Room {

int roomNumber;

bool isAvailable;

};

struct Lab {

int labNumber;

bool isAvailable;

};

struct Lab labs[MAX\_LABS];

struct Room rooms[MAX\_ROOMS];

struct Booking bookings[MAX\_BOOKINGS];

int totalBookings = 0;

int totalLabs = 0;

int totalRooms = 0;

void initializeLabs() {

for (int i = 0; i < MAX\_LABS; i++) {

labs[i].labNumber = i + 1;

labs[i].isAvailable = true;

totalLabs++;

}

}

void displayAvailableLabs() {

printf("Currently Available Labs:\n");

for (int i = 0; i < totalLabs; i++) {

if (labs[i].isAvailable) {

printf("Lab %d\n", labs[i].labNumber);

}

}

}

void initializeRooms() {

for (int i = 0; i < MAX\_ROOMS; i++) {

rooms[i].roomNumber = i + 1;

rooms[i].isAvailable = true;

totalRooms++;

}

}

void displayAvailableRooms() {

printf("Currently Available Rooms:\n");

for (int i = 0; i < totalRooms; i++) {

if (rooms[i].isAvailable) {

printf("Room %d\n", rooms[i].roomNumber);

}

}

}

void bookLab(int labNumber, const char \*purpose, const char \*userName) {

if (labNumber < 1 || labNumber > totalLabs) {

printf("Invalid lab number is given.\n");

return;

}

if (!labs[labNumber - 1].isAvailable) {

printf("Lab %d is not available.\n", labNumber);

return;

}

labs[labNumber - 1].isAvailable = false;

struct Booking newBooking;

newBooking.labNumber = labNumber;

strcpy(newBooking.purpose, purpose);

strcpy(newBooking.userName, userName);

bookings[totalBookings++] = newBooking;

printf("Lab %d booked for %s by %s.\n", labNumber, purpose, userName);

}

void bookRoom(int roomNumber, const char \*purpose, const char \*userName) {

if (roomNumber < 1 || roomNumber > totalRooms) {

printf("Invalid Room number is given.\n");

return;

}

if (!rooms[roomNumber - 1].isAvailable) {

printf("Room %d is not available.\n", roomNumber);

return;

}

rooms[roomNumber - 1].isAvailable = false;

struct Booking newBooking;

newBooking.RoomNumber = roomNumber;

strcpy(newBooking.purpose, purpose);

strcpy(newBooking.userName, userName);

bookings[totalBookings++] = newBooking;

printf("Room %d booked for %s by %s.\n", roomNumber, purpose, userName);

}

void displayLabBookings() {

printf("Lab Bookings:\n");

for (int i = 0; i < totalBookings; i++) {

if (bookings[i].labNumber > 0) {

printf("Lab %d: %s (Booked by %s)\n", bookings[i].labNumber, bookings[i].purpose, bookings[i].userName);

}

}

}

void displayRoomBookings() {

printf("Room Bookings:\n");

for (int i = 0; i < totalBookings; i++) {

if (bookings[i].RoomNumber > 0) {

printf("Room %d: %s (Booked by %s)\n", bookings[i].RoomNumber, bookings[i].purpose, bookings[i].userName);

}

}

}

void addLab() {

if (totalLabs < MAX\_LABS) {

struct Lab newLab;

newLab.labNumber = totalLabs + 1;

newLab.isAvailable = true;

labs[totalLabs++] = newLab;

printf("Lab was added successfully.\n");

} else {

printf("Maximum labs reached.\n");

}

}

void addRoom() {

if (totalRooms < MAX\_ROOMS) {

struct Room newRoom;

newRoom.roomNumber = totalRooms + 1;

newRoom.isAvailable = true;

rooms[totalRooms++] = newRoom;

printf("Room was added successfully.\n");

} else {

printf("Maximum rooms reached.\n");

}

}

void deleteLab(int labNumberToDelete) {

if (labNumberToDelete < 1 || labNumberToDelete > totalLabs) {

printf("Invalid lab number.\n");

} else {

if (labs[labNumberToDelete - 1].isAvailable) {

printf("Lab %d is already available.\n", labNumberToDelete);

} else {

labs[labNumberToDelete - 1].isAvailable = true;

printf("Lab %d deleted.\n", labNumberToDelete);

}

}

}

void deleteRoom(int roomNumberToDelete) {

if (roomNumberToDelete < 1 || roomNumberToDelete > totalRooms) {

printf("Invalid room number.\n");

} else {

if (rooms[roomNumberToDelete - 1].isAvailable) {

printf("Room %d is already available.\n", roomNumberToDelete);

} else {

rooms[roomNumberToDelete - 1].isAvailable = true;

printf("Room %d deleted.\n", roomNumberToDelete);

}

}

}

int main() {

int role;

printf("Resource Booking System\n");

initializeLabs();

initializeRooms();

while (1) {

printf("Select your role:\n");

printf("1. User\n");

printf("2. Admin\n");

printf("3. Exit\n");

scanf("%d", &role);

if (role == 1) {

while (1) {

printf("\nUser Menu\n");

printf("1. Display Available Labs\n");

printf("2. Book a Lab\n");

printf("3. Display Your Lab Bookings\n");

printf("4. Display Available Rooms\n");

printf("5. Book a Room\n");

printf("6. Display Your Room Bookings\n");

printf("7. Return to Role Selection\n");

int choice;

scanf("%d", &choice);

switch (choice) {

case 1:

displayAvailableLabs();

break;

case 2: {

int labNumber;

char purpose[100];

char userName[50];

printf("Enter a lab number between 1 to %d\n", totalLabs);

printf("Enter Lab number to be booked: ");

scanf("%d", &labNumber);

printf("Enter your purpose: ");

scanf("%s", purpose);

printf("Enter your name: ");

scanf("%s", userName);

bookLab(labNumber, purpose, userName);

break;

}

case 3:

displayLabBookings();

break;

case 4:

displayAvailableRooms();

break;

case 5: {

int roomNumber;

char purpose[100];

char userName[50];

printf("Enter a Room number between 1 to %d\n", totalRooms);

printf("Enter Room number to be booked: ");

scanf("%d", &roomNumber);

printf("Enter your purpose: ");

scanf("%s", purpose);

printf("Enter your Name: ");

scanf("%s", userName);

bookRoom(roomNumber, purpose, userName);

break;

}

case 6:

displayRoomBookings();

break;

case 7:

break;

default:

printf("Invalid choice.\n");

}

if (choice == 7) {

break;

}

}

} else if (role == 2) {

while (1) {

printf("\nAdmin Menu\n");

printf("1. Display Available Labs and Rooms\n");

printf("2. Add Lab\n");

printf("3. Delete booked Lab\n");

printf("4. Add Room\n");

printf("5. Delete Booked Room\n");

printf("6. Return to Role Selection\n");

int adminChoice;

scanf("%d", &adminChoice);

switch (adminChoice) {

case 1:

displayAvailableLabs();

displayAvailableRooms();

break;

case 2:

addLab();

break;

case 3: {

int labNumberToDelete;

printf("Enter Booked lab number to delete: ");

scanf("%d", &labNumberToDelete);

deleteLab(labNumberToDelete);

break;

}

case 4:

addRoom();

break;

case 5: {

int roomNumberToDelete;

printf("Enter Booked room number to delete: ");

scanf("%d", &roomNumberToDelete);

deleteRoom(roomNumberToDelete);

break;

}

case 6:

break;

default:

printf("Invalid choice in Admin Menu.\n");

}

if (adminChoice == 6) {

break;

}

}

} else if (role == 3) {

printf("Exiting the Resource Booking System.\n");

break;

} else {

printf("Invalid role selection.\n");

}

}

return 0;

}

Output:

Resource Booking System

Select your role:

1. User

2. Admin

3. Exit

1

User Menu

1. Display Available Labs

2. Book a Lab

3. Display Your Lab Bookings

4. Display Available Rooms

5. Book a Room

6. Display Your Room Bookings

7. Return to Role Selection

5

Enter a Room number between 1 to 10

Enter Room number to be booked: 8

Enter your purpose: STATYING

Enter your Name: XYZ

Room 8 booked for STATYING by XYZ.

User Menu

1. Display Available Labs

2. Book a Lab

3. Display Your Lab Bookings

4. Display Available Rooms

5. Book a Room

6. Display Your Room Bookings

7. Return to Role Selection

6

Room Bookings:

Room 8: STATYING (Booked by XYZ)

User Menu

1. Display Available Labs

2. Book a Lab

3. Display Your Lab Bookings

4. Display Available Rooms

5. Book a Room

6. Display Your Room Bookings

7. Return to Role Selection

7

Select your role:

1. User

2. Admin

3. Exit

2

Admin Menu

1. Display Available Labs and Rooms

2. Add Lab

3. Delete booked Lab

4. Add Room

5. Delete Booked Room

6. Return to Role Selection

4

Maximum rooms reached.

Admin Menu

1. Display Available Labs and Rooms

2. Add Lab

3. Delete booked Lab

4. Add Room

5. Delete Booked Room

6. Return to Role Selection

3

Enter Booked lab number to delete: 8

Invalid lab number.

Admin Menu

1. Display Available Labs and Rooms

2. Add Lab

3. Delete booked Lab

4. Add Room

5. Delete Booked Room

6. Return to Role Selection

5

Enter Booked room number to delete: 8

Room 8 deleted.

Admin Menu

1. Display Available Labs and Rooms

2. Add Lab

3. Delete booked Lab

4. Add Room

5. Delete Booked Room

6. Return to Role Selection

6

Select your role:

1. User

2. Admin

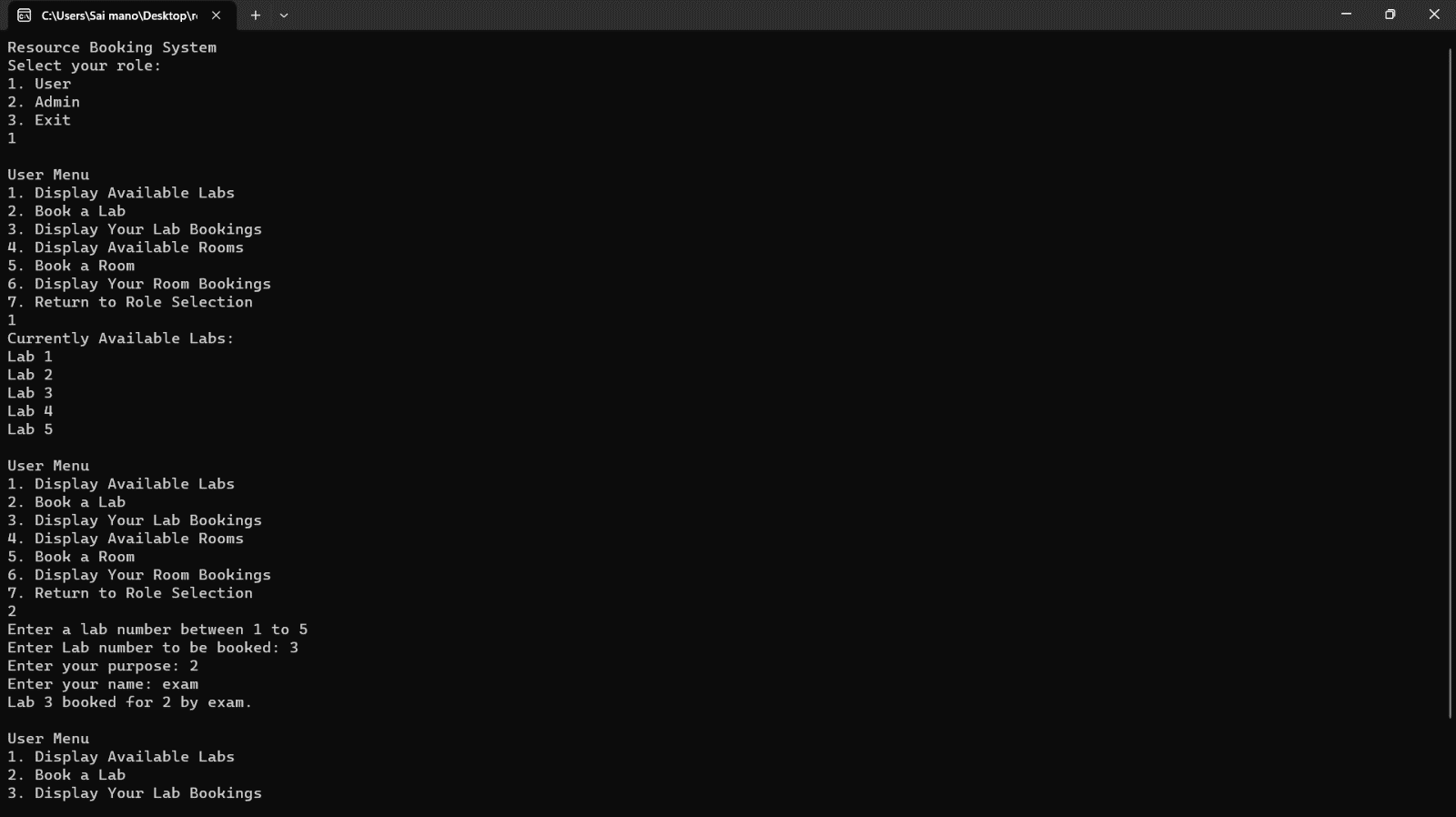
3. Exit

3

Exiting the Resource Booking System.

Process returned 0 (0x0) execution time : 84.631 s

Press any key to continue.



**CONCLUSION:**

Resource Booking and Management plays a pivotal role in modern organizational efficiency and productivity. It encompasses the effective allocation and utilization of various resources, from meeting rooms to equipment, within an organization. The key components of this system include a resource catalog, booking system, user management, notifications, reporting and analytics, integration, and robust security measures.

**REFERENCES:**

[**https://www.saviom.com/blog/resource-booking-and-how-drag-and-drop-make-it-effective/**](https://www.saviom.com/blog/resource-booking-and-how-drag-and-drop-make-it-effective/)

[**https://squareup.com/au/en/appointments/features/resource-booking**](https://squareup.com/au/en/appointments/features/resource-booking)

[**https://github.com/bexis/Module\_ResourceManagement**](https://github.com/bexis/Module_ResourceManagement)

**WORK DISTRIBUTION:**

**> MINDMAP:**

**1.sameeksha**

**2.savi**

**> CODE CREATION:**

**1.sai mano Dileep**

**2.sameeksha**

**3.ganesh**

**4.savi**

**5.varun**

**REPORT:**

**1.varun mc**

**2.Ganesh**