Project Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <ctype.h>

// Clear the console screen for Windows
void clearScreen() {
    system("cls");
}

// Helper Functions
char *toLowerCase(const char *str) {
    char *lower = malloc(strlen(str) + 1);  // Dynamically allocate memory for the lowercase
string
    if (lower == NULL) {
        printf("Memory allocation error!\n");
        exit(1);
    }
    int i = 0;
    while (str[i]) {
        lower[i] = tolower(str[i]);
        i++;
    }
    lower[i] = '\0';
    return lower;
}

typedef struct {
    int bookID;
    char title[100];
    char author[100];
```

```
    int isAvailable;
} Book;

typedef struct {
    int userID;
    char name[50];
    char password[20];
} User;

typedef struct {
    int reservationID;
    int userID;
    int bookID;
    char date[20];
    int isApproved;
} Reservation;

typedef struct {
    int transactionID;
    int userID;
    int bookID;
    char date[20];
    int isReturned;
} Transaction;


// Function Prototypes
void menu();
void adminMenu();
void memberMenu(int userID);

void registerUser();
int loginUser();
int admin_login();
void addBook();
void searchBook();
void viewAllBooks();
void borrowBook(int userID);
void returnBook(int userID);
void renewBook(int userID);
void reserveBook(int userID);
void viewBorrowingHistory(int userID);
void approveReservation();
void generateReport();
int generateUniqueBookID();

void pause() {
    printf("\nPress Enter to continue...");
```

```
    while (getchar() != '\n');
    getchar();
}

int main() {
    int choice;
    while (1) {
        clearScreen(); // Clear screen at the start of the loop
        printf("\n--- Library Management System ---\n");
        printf("1. Login as Member\n");
        printf("2. Login as Admin\n");
        printf("3. Register as Member\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1: {
                clearScreen();
                int userID = loginUser();
                if (userID != -1) memberMenu(userID);
                break;
            }
            case 2:
                clearScreen();
                if (admin_login()) {
                    adminMenu();  // Show admin menu only if login is successful
                } else {
                    printf("Admin login failed.\n");
                    pause();
                }
                break;
            case 3:
                clearScreen();
                registerUser();
                pause(); // Allow the user to see the success message
                break;
            case 4:
                exit(0);
            default:
                printf("Invalid choice! Try again.\n");
                pause(); // Allow the user to read the error message
        }
    }
    return 0;
}

// Admin Menu
```

```
void adminMenu() {
    int choice;
    do {
        clearScreen();
        printf("\n--- Admin Menu ---\n");
        printf("1. Add Book\n");
        printf("2. Search Book\n");
        printf("3. View All Books\n");
        printf("4. Approve Reservation\n");
        printf("5. Generate Report\n");
        printf("6. Logout\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                addBook();
                break;
            case 2:
                searchBook();
                break;
            case 3:
                viewAllBooks();
                break;
            case 4:
                approveReservation();
                break;
            case 5:
                generateReport(); // Generate report option
                break;
            case 6:
                return;
            default:
                printf("Invalid choice! Try again.\n");
        }
        pause();
    } while (choice != 6);
}

// Member Menu
void memberMenu(int userID) {
    int choice;
    do {
        clearScreen();
        printf("\n--- Member Menu ---\n");
        printf("1. Search Book\n");
        printf("2. View All Books\n");
        printf("3. Borrow Book\n");
```

```
        printf("4. Reserve Book\n");
        printf("5. Return Book\n");
        printf("6. Renew Book\n");
        printf("7. View Borrowing History\n");
        printf("8. Logout\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                searchBook();
                break;
            case 2:
                viewAllBooks();
                break;
            case 3:
                borrowBook(userID);
                break;
            case 4:
                reserveBook(userID);
                break;
            case 5:
                returnBook(userID);
                break;
            case 6:
                renewBook(userID);
                break;
            case 7:
                viewBorrowingHistory(userID);
                break;
            case 8:
                return;
            default:
                printf("Invalid choice! Try again.\n");
        }
        pause();
    } while (choice != 8);
}

void registerUser() {
    User newUser;
    FILE *userFile = fopen("Users.txt", "a");

    if (!userFile) {
        printf("Error opening user file.\n");
        return;
    }
```

```
    printf("\n--- Registration ---\n");
    printf("Enter your name: ");
    getchar(); // Clear newline character from previous input
    fgets(newUser.name, sizeof(newUser.name), stdin);
    newUser.name[strcspn(newUser.name, "\n")] = '\0'; // Remove newline character

    printf("Enter your password: ");
    fgets(newUser.password, sizeof(newUser.password), stdin);
    newUser.password[strcspn(newUser.password, "\n")] = '\0'; // Remove newline character

    newUser.userID = rand() % 10000; // Random ID for simplicity

    fwrite(&newUser, sizeof(User), 1, userFile);
    printf("Registration successful! Your User ID: %d\n", newUser.userID);

    fclose(userFile);
}

int loginUser() {
    char name[50], password[20];
    FILE *userFile = fopen("Users.txt", "r");

    if (!userFile) {
        printf("Error opening user file.\n");
        return -1;
    }

    printf("\n--- Login ---\n");
    printf("Enter your name: ");
    getchar(); // Clear newline character from previous input
    fgets(name, sizeof(name), stdin);
    name[strcspn(name, "\n")] = '\0'; // Remove newline character

    printf("Enter your password: ");
    fgets(password, sizeof(password), stdin);
    password[strcspn(password, "\n")] = '\0'; // Remove newline character

    User user;
    while (fread(&user, sizeof(User), 1, userFile)) {
        if (strcmp(user.name, name) == 0 && strcmp(user.password, password) == 0) {
            printf("Login successful! Welcome, %s!\n", user.name);
            fclose(userFile);
            return user.userID; // Return the user's ID if login is successful
        }
    }

    printf("Invalid name or password. Please try again.\n");
    fclose(userFile);
```

```
    return -1; // Return -1 if login fails
}

#define ADMIN_USERID "admin"
#define ADMIN_PASSWORD "shakil123"

// Function for admin login
int admin_login() {
    char entered_userid[20];
    char entered_password[20];

    printf("Admin Login\n");
    printf("Enter User ID: ");
    scanf("%19s", entered_userid); // Using %19s to prevent buffer overflow
    printf("Enter Password: ");
    scanf("%19s", entered_password);

    if (strcmp(entered_userid, ADMIN_USERID) == 0 && strcmp(entered_password,
ADMIN_PASSWORD) == 0) {
        printf("Login successful!\n");
        return 1; // Login successful
    } else {
        printf("Invalid User ID or Password. Access Denied.\n");
        return 0; // Login failed
    }
}

void addBook() {
    Book newBook;
    FILE *bookFile = fopen("Books.txt", "a");

    if (!bookFile) {
        printf("Error opening book file.\n");
        return;
    }

    printf("\n--- Add Book ---\n");
    printf("Enter Book Title: ");
    getchar(); // to clear the newline character left by previous input
    fgets(newBook.title, sizeof(newBook.title), stdin);
    newBook.title[strcspn(newBook.title, "\n")] = '\0'; // Remove newline character

    printf("Enter Author Name: ");
    fgets(newBook.author, sizeof(newBook.author), stdin);
    newBook.author[strcspn(newBook.author, "\n")] = '\0'; // Remove newline character

    newBook.bookID = generateUniqueBookID(); // Use the function to generate a unique ID
    newBook.isAvailable = 1; // Initially the book is available
```

```
    fwrite(&newBook, sizeof(Book), 1, bookFile);
    printf("Book added successfully! Book ID: %d\n", newBook.bookID);

    fclose(bookFile);
}


// In the searchBook function
void searchBook() {
    char searchTerm[50];
    printf("\nEnter Book Title or Author to search: ");
    getchar(); // Clear newline from previous input
    fgets(searchTerm, sizeof(searchTerm), stdin);
    searchTerm[strcspn(searchTerm, "\n")] = '\0'; // Remove newline character

    if (strlen(searchTerm) == 0) {
        printf("Search term cannot be empty.\n");
        return;
    }

    char *loweredSearchTerm = toLowerCase(searchTerm);  // Dynamically allocate memory
for the lowercase version of searchTerm
    FILE *bookFile = fopen("Books.txt", "r");
    if (!bookFile) {
        printf("Error opening book file. Make sure the file exists.\n");
        free(loweredSearchTerm);  // Free the allocated memory
        return;
    }

    Book book;
    int found = 0;
    printf("\n--- Search Results ---\n");
    while (fread(&book, sizeof(Book), 1, bookFile)) {
        // Case insensitive comparison for title and author
        if (strstr(toLowerCase(book.title), loweredSearchTerm) ||
            strstr(toLowerCase(book.author), loweredSearchTerm)) {
            printf("\nBook ID: %d\n", book.bookID);
            printf("Title: %s\n", book.title);
            printf("Author: %s\n", book.author);
            printf("Availability: %s\n", book.isAvailable ? "Available" : "Not Available");
            found = 1;
            break; // Stop the loop once a match is found
        }
    }

    if (!found) {
        printf("No books found matching your search term.\n");
```

```
    }

    fclose(bookFile);
    free(loweredSearchTerm);  // Free the dynamically allocated memory
}

void viewAllBooks() {
    FILE *bookFile = fopen("Books.txt", "r");

    if (!bookFile) {
        printf("Error opening book file.\n");
        return;
    }

    Book book;
    printf("\n--- All Books ---\n");
    while (fread(&book, sizeof(Book), 1, bookFile)) {
        printf("\nBook ID: %d\n", book.bookID);
        printf("Title: %s\n", book.title);
        printf("Author: %s\n", book.author);
        printf("Availability: %s\n", book.isAvailable ? "Available" : "Not Available");
    }

    fclose(bookFile);
}


// Borrow Book
// Borrow Book
void borrowBook(int userID) {
    int bookID;
    printf("\nEnter the Book ID to borrow: ");
    scanf("%d", &bookID);
    getchar(); // Clear the newline character

    FILE *bookFile = fopen("Books.txt", "r+");
    if (!bookFile) {
        printf("Error opening book file.\n");
        return;
    }

    Book book;
    int found = 0;
    while (fread(&book, sizeof(Book), 1, bookFile)) {
        if (book.bookID == bookID) {
            found = 1;
            printf("Book Title: %s\n", book.title);  // Display the title of the book
            if (book.isAvailable) {
```

```
            book.isAvailable = 0;
            fseek(bookFile, -sizeof(Book), SEEK_CUR);
            fwrite(&book, sizeof(Book), 1, bookFile);

            // Record the transaction
            FILE *transactionFile = fopen("Transactions.txt", "a");
            if (!transactionFile) {
                printf("Error opening transaction file.\n");
                book.isAvailable = 1; // Rollback
                fseek(bookFile, -sizeof(Book), SEEK_CUR);
                fwrite(&book, sizeof(Book), 1, bookFile);
                fclose(bookFile);
                return;
            }

            Transaction newTransaction = {rand() % 10000, userID, bookID, "", 0};
            time_t t = time(NULL);
            struct tm tm = *localtime(&t);
            strftime(newTransaction.date, sizeof(newTransaction.date), "%Y-%m-%d
%H:%M:%S", &tm);

            fwrite(&newTransaction, sizeof(Transaction), 1, transactionFile);
            fclose(transactionFile);

            printf("Book borrowed successfully!\n");
        } else {
            printf("Book is currently unavailable.\n");
        }
        break;
    }
  }

  if (!found) {
      printf("Book not found.\n");
  }

  fclose(bookFile);
}


// Return Book
void returnBook(int userID) {
  int bookID;
  printf("\nEnter the Book ID to return: ");
  scanf("%d", &bookID);
  getchar(); // Clear the newline character

  FILE *bookFile = fopen("Books.txt", "r+");
```

```
    if (!bookFile) {
        printf("Error opening book file.\n");
        return;
    }

    Book book;
    int found = 0;
    while (fread(&book, sizeof(Book), 1, bookFile)) {
        if (book.bookID == bookID) {
            found = 1;
            if (!book.isAvailable) {
                book.isAvailable = 1;
                fseek(bookFile, -sizeof(Book), SEEK_CUR);
                fwrite(&book, sizeof(Book), 1, bookFile);

                // Update transaction as returned
                FILE *transactionFile = fopen("Transactions.txt", "r+");
                if (!transactionFile) {
                    printf("Error opening transaction file.\n");
                    book.isAvailable = 0; // Rollback
                    fseek(bookFile, -sizeof(Book), SEEK_CUR);
                    fwrite(&book, sizeof(Book), 1, bookFile);
                    fclose(bookFile);
                    return;
                }

                Transaction transaction;
                int transactionFound = 0;
                while (fread(&transaction, sizeof(Transaction), 1, transactionFile)) {
                    if (transaction.userID == userID && transaction.bookID == bookID &&
transaction.isReturned == 0) {
                        transaction.isReturned = 1;
                        fseek(transactionFile, -sizeof(Transaction), SEEK_CUR);
                        fwrite(&transaction, sizeof(Transaction), 1, transactionFile);
                        transactionFound = 1;
                        break;
                    }
                }
                fclose(transactionFile);

                if (!transactionFound) {
                    printf("Error: Matching borrow transaction not found.\n");
                    book.isAvailable = 0; // Rollback
                    fseek(bookFile, -sizeof(Book), SEEK_CUR);
                    fwrite(&book, sizeof(Book), 1, bookFile);
                } else {
                    printf("Book returned successfully!\n");
                }
```

```
            } else {
                printf("This book was not borrowed.\n");
            }
            break;
        }
    }

    if (!found) {
        printf("Book not found.\n");
    }

    fclose(bookFile);
}


// Renew Book Function
void renewBook(int userID) {
    int bookID;
    printf("\nEnter the Book ID to renew: ");
    scanf("%d", &bookID);

    FILE *bookFile = fopen("Books.txt", "r+");
    if (!bookFile) {
        printf("Error opening book file.\n");
        return;
    }

    Book book;
    int found = 0;
    while (fread(&book, sizeof(Book), 1, bookFile)) {
        if (book.bookID == bookID) {
            found = 1;
            if (!book.isAvailable) {
                // You can add logic for renewal time extension here
                printf("Book renewed successfully!\n");
            } else {
                printf("This book is not borrowed.\n");
            }
            break;
        }
    }

    if (!found) {
        printf("Book not found.\n");
    }

    fclose(bookFile);
}
```

```
// Inside your reserveBook function
void reserveBook(int userID) {
    int bookID;
    printf("\nEnter the Book ID to reserve: ");
    scanf("%d", &bookID);

    FILE *bookFile = fopen("Books.txt", "r");
    if (!bookFile) {
        printf("Error opening book file.\n");
        return;
    }

    Book book;
    int found = 0;
    while (fread(&book, sizeof(Book), 1, bookFile)) {
        if (book.bookID == bookID) {
            found = 1;
            if (book.isAvailable) {
                printf("The book is already available. You can borrow it.\n");
            } else {
                // Create a reservation with current date
                FILE *reservationFile = fopen("Reservations.txt", "a");
                if (!reservationFile) {
                    printf("Error opening reservation file.\n");
                    return;
                }

                // Get the current date and time
                time_t t = time(NULL);
                struct tm tm = *localtime(&t);
                char date[20];
                strftime(date, sizeof(date), "%Y-%m-%d %H:%M:%S", &tm); // Format date and
time

                Reservation newReservation = { rand() % 10000, userID, bookID, "", 0 };
                strcpy(newReservation.date, date); // Set the reservation date to the current time

                fwrite(&newReservation, sizeof(Reservation), 1, reservationFile);
                fclose(reservationFile);

                printf("Book reserved successfully! We will notify you when it's available.\n");
            }
            break;
        }
    }
```

```
    if (!found) {
        printf("Book not found.\n");
    }

    fclose(bookFile);
}


void viewBorrowingHistory(int userID) {
    FILE *transactionFile = fopen("Transactions.txt", "r");
    if (!transactionFile) {
        printf("Error opening transaction file.\n");
        return;
    }

    Transaction transaction;
    int found = 0;
    printf("\n--- Borrowing History ---\n");
    while (fread(&transaction, sizeof(Transaction), 1, transactionFile)) {
        if (transaction.userID == userID) {
            printf("\nTransaction ID: %d\n", transaction.transactionID);
            printf("Book ID: %d\n", transaction.bookID);
            printf("Date: %s\n", transaction.date);
            printf("Returned: %s\n", transaction.isReturned ? "Yes" : "No");
            found = 1;
        }
    }

    if (!found) {
        printf("No borrowing history found for User ID: %d\n", userID);
    }

    fclose(transactionFile);
}

void approveReservation() {
    int reservationID;
    printf("\nEnter Reservation ID to approve: ");
    scanf("%d", &reservationID);

    FILE *reservationFile = fopen("Reservations.txt", "r+");
    if (!reservationFile) {
        printf("Error opening reservation file.\n");
        return;
    }

    Reservation reservation;
    int found = 0;
```

```
    while (fread(&reservation, sizeof(Reservation), 1, reservationFile)) {
        if (reservation.reservationID == reservationID && reservation.isApproved == 0) {
            reservation.isApproved = 1;
            fseek(reservationFile, -sizeof(Reservation), SEEK_CUR);
            fwrite(&reservation, sizeof(Reservation), 1, reservationFile);
            printf("Reservation approved successfully!\n");
            found = 1;
            break;
        }
    }

    if (!found) {
        printf("Reservation not found or already approved.\n");
    }

    fclose(reservationFile);
}

void generateReport() {
    printf("\n--- Library Report ---\n");
    FILE *bookFile = fopen("Books.txt", "r");
    if (!bookFile) {
        printf("Error opening book file.\n");
        return;
    }

    Book book;
    int totalBooks = 0, availableBooks = 0;
    while (fread(&book, sizeof(Book), 1, bookFile)) {
        totalBooks++;
        if (book.isAvailable) {
            availableBooks++;
        }
    }

    printf("Total Books: %d\n", totalBooks);
    printf("Available Books: %d\n", availableBooks);
    printf("Borrowed Books: %d\n", totalBooks - availableBooks);

    fclose(bookFile);
}

int generateUniqueBookID() {
    int newID;
    FILE *bookFile = fopen("Books.txt", "r");
    if (!bookFile) {
        return rand() % 10000;  // If file can't be opened, generate a random ID
    }
```

```
    Book book;
    int isUnique;
    do {
        isUnique = 1;
        newID = rand() % 10000;
        fseek(bookFile, 0, SEEK_SET);  // Reset file pointer before checking all records
        // Check if ID already exists
        while (fread(&book, sizeof(Book), 1, bookFile)) {
            if (book.bookID == newID) {
                isUnique = 0;  // ID is not unique
                break;
            }
        }
    } while (!isUnique);

    fclose(bookFile);
    return newID;
}
```