Course Registration Portal (Academia)

Course: EGC 301P Operating Systems Lab
Institute: IIIT Bangalore
Student Name: Akshat Batra
Roll Number: IMT2023025
Programming Language: C
Concepts Used: Socket Programming, File Handling, Pthreads, File Locking, System Calls

1. Problem Statement

Academic institutions require reliable and secure platforms to manage the course registration process. These systems must handle multiple roles—such as administrators, faculty, and students—each with their own set of operations. Students should be able to register for courses, faculty should manage courses, and administrators must control user accounts.

This project simulates such a system using a **terminal-based client-server application in C**, leveraging **TCP socket programming** for real-time communication, **POSIX threads** for concurrent client handling, and **binary files with file locks** for data integrity. The system supports multiple simultaneous users and enforces strict access control through a login system.

The goal is to implement an extensible, role-based course registration portal that enforces security, concurrency, and consistency—while remaining lightweight and system-level in design.

2. Implementation Details

The portal is structured using a **client-server architecture** implemented in C. The **server** manages persistent data and business logic, while **clients** connect via TCP sockets and interact with the system using a text-based interface.

Socket Communication

The system uses **raw TCP sockets** (socket(), bind(), listen(), accept(), connect(), read(), write()) to establish real-time communication between client and server.

- The **server** runs on a fixed port (8080), listens for connections, and spawns a new thread for every client.
- The **client** connects using connect() and interacts using prompts and input/output exchanged over sockets.

All communication is manual — no protocol middleware is used. The server sends prompts like "Enter username:", the client responds, and the session proceeds accordingly.

Multi-Threading and Concurrency

Each client connection is handled in a separate thread using pthread_create(). Threads run concurrently, ensuring multiple users can operate simultaneously—students enrolling in courses, admins modifying accounts, and faculty adding courses.

To prevent race conditions, shared resources (like courses.dat, users.dat) are protected using **file-level locks** implemented via fcntl():

- **Read locks** for viewing data
- **Write locks** for modifying data

This ensures no two threads corrupt the same file during simultaneous access.

File Handling and Persistence

All data is stored persistently in binary files:

- users.dat — stores all user accounts (ID, username, password, role, active)
- courses.dat — stores course data (ID, name, faculty, seats)
- enrollments.dat — maps students to courses

Files are accessed using fopen(), fread(), fwrite(), and fseek() for efficient low-level(granular) control.

Authentication & Access Control

Every client must log in by providing a username, password, and selecting their role. The server verifies these credentials using the authenticate() function. Students must also be marked “active” to log in.

After successful login, the user is redirected to a **role-specific menu**:

- Admin: manages users
- Faculty: manages courses
- Student: enrolls/unenrolls from courses

All role logic is encapsulated and strictly enforced.

3. Source Code for Main Files with Brief Explanation

server.c

Purpose: Core logic — manages socket setup, threads, login, and all backend operations.

Key Functions:

main() – Server Startup

```
int main() {
   bind(...);
   listen(...);
   while (1) {
      client_socket = accept(...);
      pthread_create(...);
   }
}
```

Starts server, listens for incoming connections, and handles each client in a separate thread.

handle_client() – Client Thread Logic

```
if (authenticate(...)) {
    if (strcmp(role, "admin")) handle_admin(...);
}
```

Manages login, verifies credentials, and routes the client to their respective menu.

Role-Specific Handlers:

- handle_admin() — add/view/block users
- handle_student() — enroll/unenroll/view courses
- handle_faculty() — add/remove/update courses

Each uses read(), write(), and file operations with locking.

client.c

Purpose: Lightweight client that connects to the server and processes server prompts.

Key Loop:

```
while (1) {
    read(sock, buffer, ...);
    if (strstr(buffer, ":")) {
        fgets(...);
        write(sock, ...);
    }
}
```

Continuously listens to the server, takes input from the user, and sends it back.

common.h

Purpose: Contains shared structures and function declarations.

Structures:

```
struct User {
    int id;
    char username[50], password[50], role[10];
    int active;
};
```

Used throughout the project for storing and transferring user, course, and enrollment data.

Functions:

- authenticate(...)
- lock_file(...)
- unlock_file(...)

init_data.c

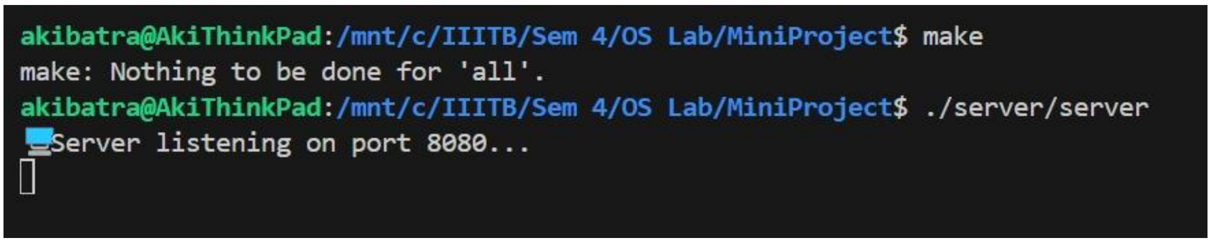
Purpose: Sets up sample data for testing.

Example Usage:

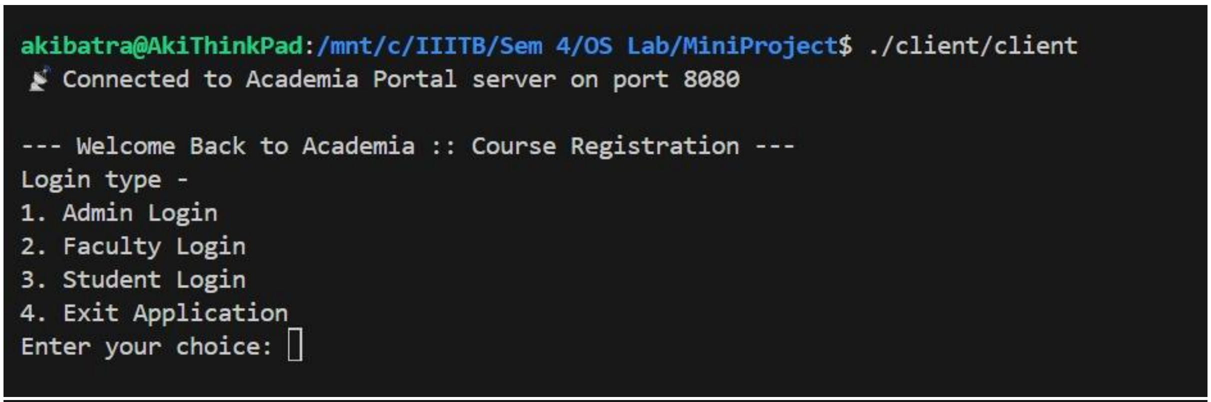
```
add_user(1, "admin1", "admin1pwd", "admin", 1);
add_course(101, "OperatingSystems", 3, 3);
```

Clears previous data and inserts users and courses for demo/test use.

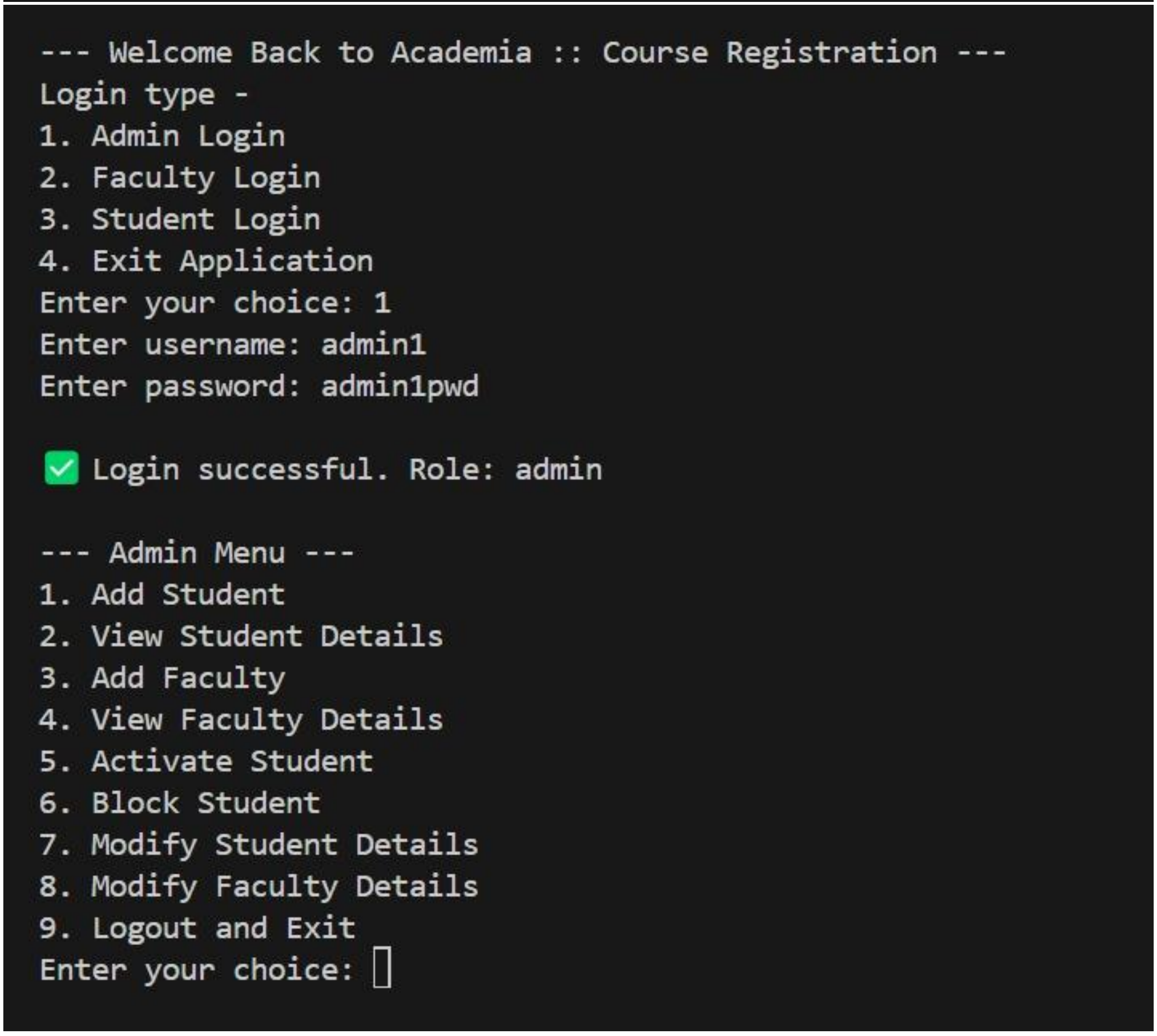
4. Output Screenshots



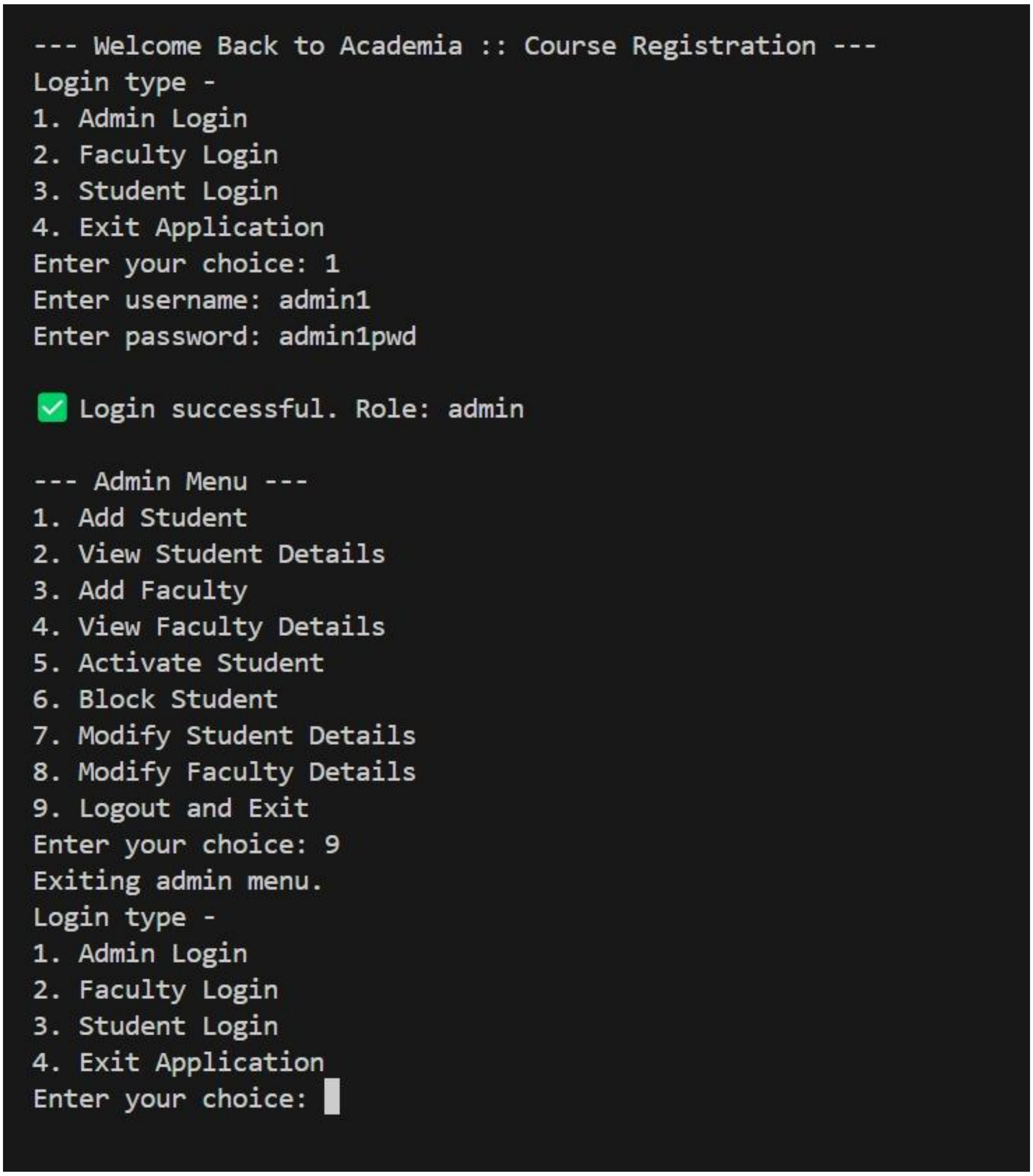
```
akibatra@AkiThinkPad:/mnt/c/IIITB/Sem 4/OS Lab/MiniProject$ make
make: Nothing to be done for 'all'.
akibatra@AkiThinkPad:/mnt/c/IIITB/Sem 4/OS Lab/MiniProject$ ./server/server
💻Server listening on port 8080...
```



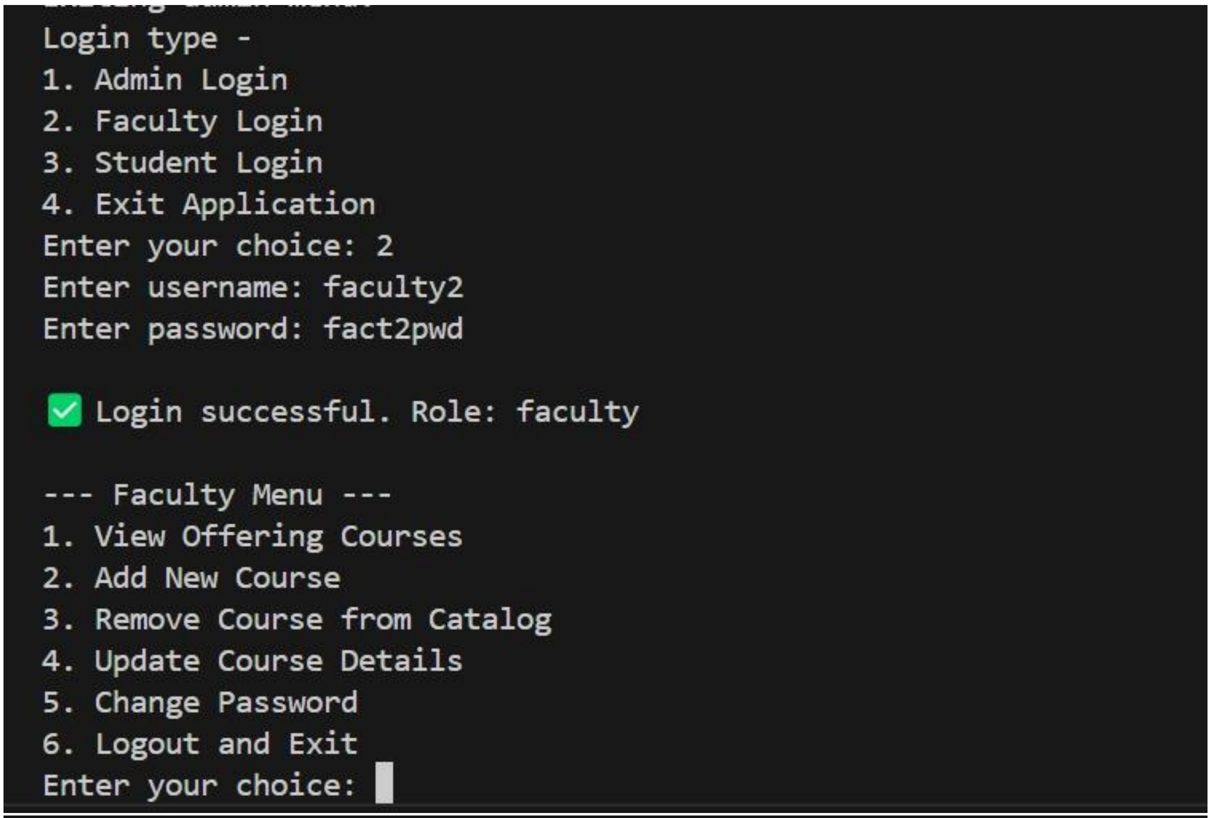
```
akibatra@AkiThinkPad:/mnt/c/IIITB/Sem 4/OS Lab/MiniProject$ ./client/client
 Connected to Academia Portal server on port 8080

--- Welcome Back to Academia :: Course Registration ---
Login type -
1. Admin Login
2. Faculty Login
3. Student Login
4. Exit Application
Enter your choice:
```



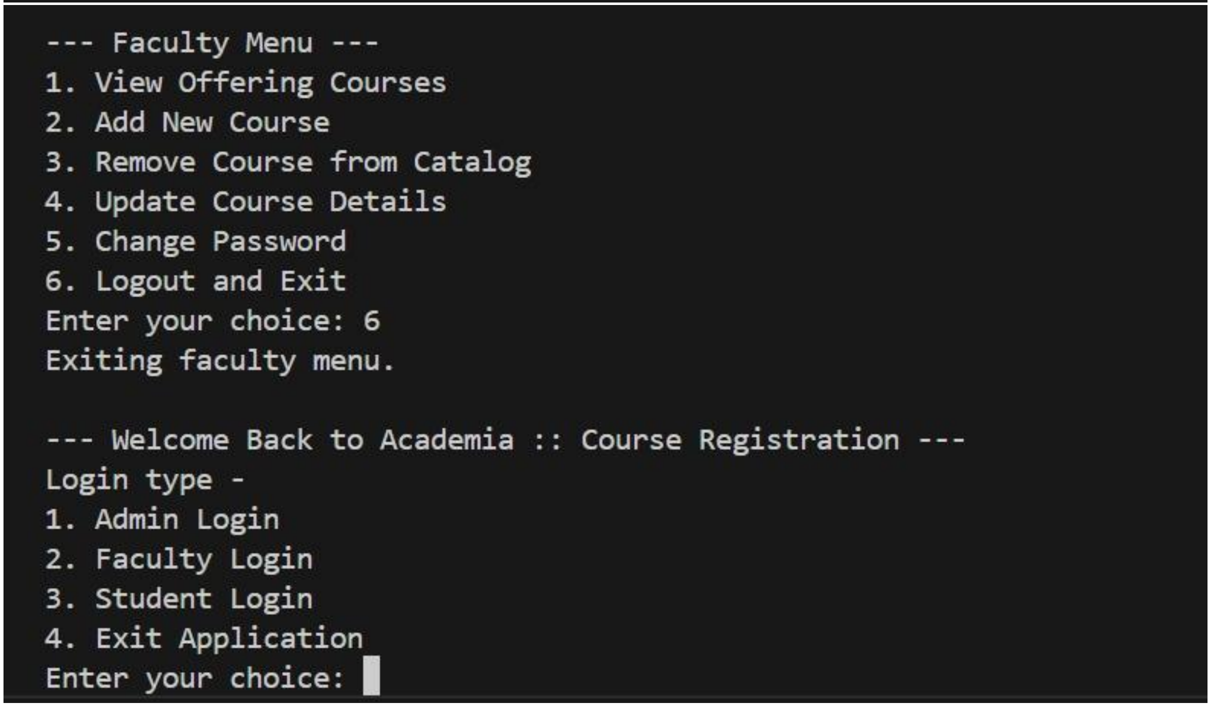
```
--- Welcome Back to Academia :: Course Registration ---
Login type -
1. Admin Login
2. Faculty Login
3. Student Login
4. Exit Application
Enter your choice: 1
Enter username: admin1
Enter password: admin1pwd

✅ Login successful. Role: admin

--- Admin Menu ---
1. Add Student
2. View Student Details
3. Add Faculty
4. View Faculty Details
5. Activate Student
6. Block Student
7. Modify Student Details
8. Modify Faculty Details
9. Logout and Exit
Enter your choice:
```



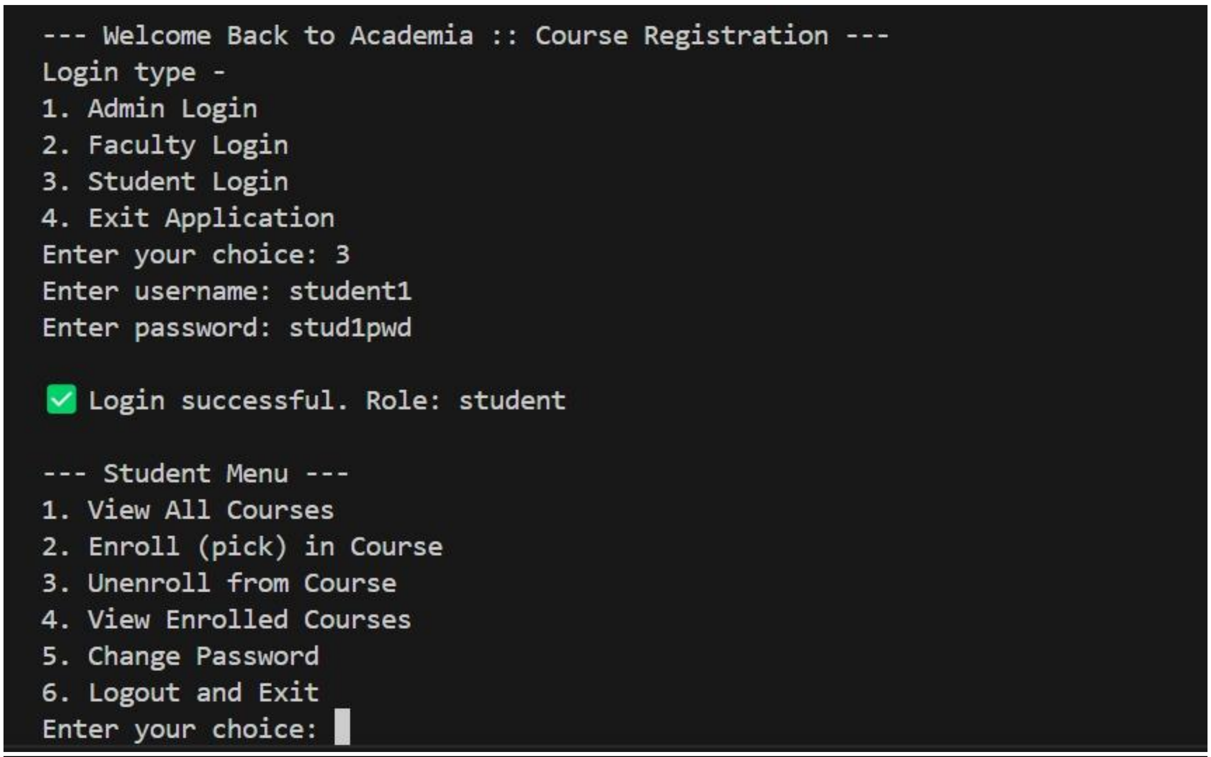
```
--- Welcome Back to Academia :: Course Registration ---
Login type -
1. Admin Login
2. Faculty Login
3. Student Login
4. Exit Application
Enter your choice: 1
Enter username: admin1
Enter password: admin1pwd

✅ Login successful. Role: admin

--- Admin Menu ---
1. Add Student
2. View Student Details
3. Add Faculty
4. View Faculty Details
5. Activate Student
6. Block Student
7. Modify Student Details
8. Modify Faculty Details
9. Logout and Exit
Enter your choice: 9
Exiting admin menu.
Login type -
1. Admin Login
2. Faculty Login
3. Student Login
4. Exit Application
Enter your choice: 
```



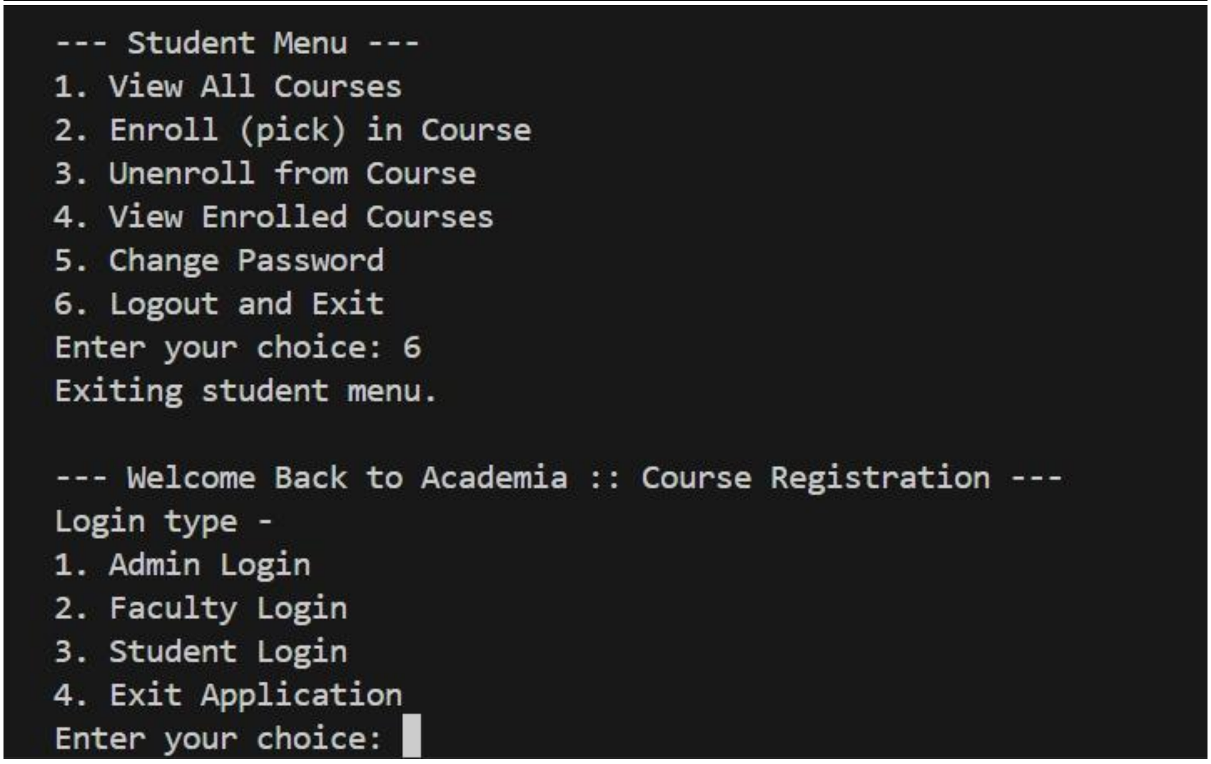
```
Login type -
1. Admin Login
2. Faculty Login
3. Student Login
4. Exit Application
Enter your choice: 2
Enter username: faculty2
Enter password: fact2pwd

✅ Login successful. Role: faculty

--- Faculty Menu ---
1. View Offering Courses
2. Add New Course
3. Remove Course from Catalog
4. Update Course Details
5. Change Password
6. Logout and Exit
Enter your choice:
```



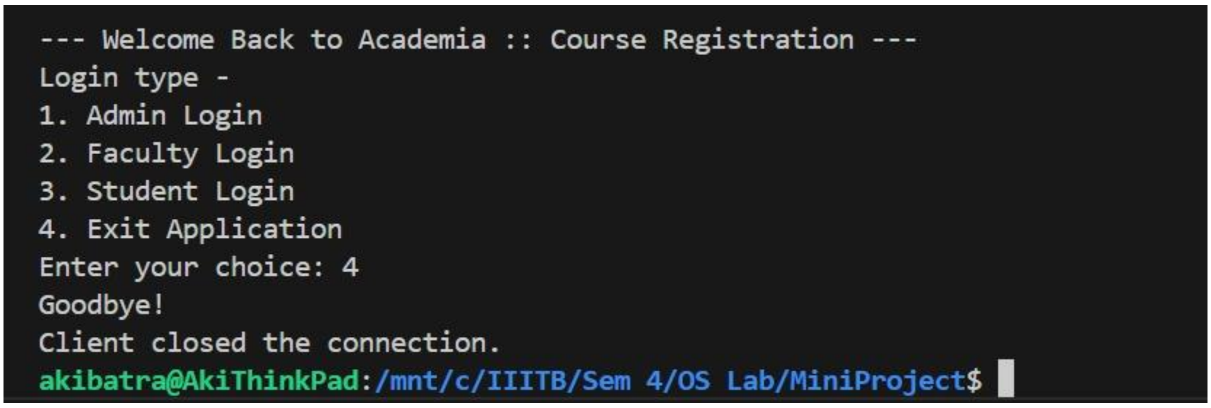
```
--- Faculty Menu ---
1. View Offering Courses
2. Add New Course
3. Remove Course from Catalog
4. Update Course Details
5. Change Password
6. Logout and Exit
Enter your choice: 6
Exiting faculty menu.

--- Welcome Back to Academia :: Course Registration ---
Login type -
1. Admin Login
2. Faculty Login
3. Student Login
4. Exit Application
Enter your choice:
```



```
--- Welcome Back to Academia :: Course Registration ---
Login type -
1. Admin Login
2. Faculty Login
3. Student Login
4. Exit Application
Enter your choice: 3
Enter username: student1
Enter password: stud1pwd

✅ Login successful. Role: student

--- Student Menu ---
1. View All Courses
2. Enroll (pick) in Course
3. Unenroll from Course
4. View Enrolled Courses
5. Change Password
6. Logout and Exit
Enter your choice:
```



```
--- Student Menu ---
1. View All Courses
2. Enroll (pick) in Course
3. Unenroll from Course
4. View Enrolled Courses
5. Change Password
6. Logout and Exit
Enter your choice: 6
Exiting student menu.

--- Welcome Back to Academia :: Course Registration ---
Login type -
1. Admin Login
2. Faculty Login
3. Student Login
4. Exit Application
Enter your choice:
```



```
--- Welcome Back to Academia :: Course Registration ---
Login type -
1. Admin Login
2. Faculty Login
3. Student Login
4. Exit Application
Enter your choice: 4
Goodbye!
Client closed the connection.
akibatra@AkiThinkPad:/mnt/c/IIITB/Sem 4/OS Lab/MiniProject$
```