Problem Statement

Title: Design and Development of Course Registration Portal (Academia)

Description: The project aims to develop a user-friendly and multifunctional Academia Portal to manage course registrations, user authentication, and administrative tasks in an educational institution. The portal supports three types of users: Admin, Faculty, and Students, each with specific functionalities. The system ensures secure access, concurrent user handling, and persistent data storage for users, courses, and enrollments.

Implementation Details

The Academia Portal is implemented as a client-server application in C, with the following key components:

1. **Architecture**:
    a. **Client-Server Model**: The server (`server.c`) handles multiple client connections using TCP sockets and threads. The client (`client.c`) provides a user interface for interaction.
    b. **Multithreading**: Each client connection spawns a new thread (`client_handler`) to handle requests concurrently.
    c. **Data Storage**: Data is stored in binary files (`users.dat`, `students.dat`, `faculty.dat`, `courses.dat`) with file locking for concurrent access.
2. **Key Features**:
    a. **Authentication**: Users log in by selecting a role (Admin, Faculty, Student) and providing credentials, verified against `users.dat`.
    b. **Role-Based Menus**:
        i. **Admin**: Manage students and faculty (add, view, modify, activate/block).
        ii. **Faculty**: Manage courses (add, remove, update, view) and change password.
        iii. **Student**: Enroll/drop courses, view courses, and change password.
    c. **Data Persistence**: File operations (`file_ops.c`) handle CRUD operations with validation and locking.
    d. **Concurrency Control**: Semaphores (`file_sem`) and file locks (`read_lock`, `write_lock`) ensure thread-safe file access.
3. **Communication Protocol**:

    a. Messages are sent with a length prefix (4-byte network byte order) to ensure accurate data transfer.
    b. The server sends menus and prompts, and the client responds with choices and inputs.
4. **Error Handling**:
    a. Input validation for IDs, names, passwords, and numbers.
    b. Specific error codes (e.g., `ERR_FULL`, `ERR_ALREADY_ENROLLED`) for course enrollment issues.
    c. Logging of server events to `server.log` for debugging.
5. **Networking**:
    a. TCP sockets with `TCP_NODELAY` to disable Nagle's algorithm for low-latency communication.
    b. SIGPIPE ignored to prevent crashes on broken client connections.
6. **File Structure**:
    a. **server.c**: Implements the server, handling client connections, authentication, and role-based menus.
    b. **client.c**: Implements the client, providing a CLI for user interaction.
    c. **file_ops.c**: Handles file operations for data persistence and concurrency.
    d. **academia.h**: Defines constants, structures, and function prototypes.
7. **Data Structures**:
    a. **User**: Stores ID, role, and password.
    b. **Student**: Stores ID, name, active status, and enrolled courses.
    c. **Faculty**: Stores ID and name.
    d. **Course**: Stores ID, name, faculty ID, total seats, enrolled count, and enrolled students.
8. **Security and Robustness**:
    a. Input validation prevents invalid data.
    b. File locking prevents race conditions.
    c. Error messages guide users on failures (e.g., "Course is full").

Brief Explanation of Source Code

Below is a summarized explanation of the provided source code files, focusing on their purpose and key functionalities.

1. academia.h

- **Purpose**: Header file defining constants, data structures, and function prototypes.
- **Key Components**:
  - **Constants**: `PORT` (8080), `MAX_CLIENTS` (10), `MAX_COURSES` (100), `MAX_USERS` (100), `MAX_NAME` (50), `MAX_ID` (10).
  - **Error Codes**: `ERR_NOT_FOUND`, `ERR_FULL`, `ERR_ALREADY_ENROLLED`, etc.
  - **Enums and Structs**: `Role` (ADMIN, STUDENT, FACULTY), `User`, `Course`, `Student`, `Faculty`.
  - **Prototypes**: Functions for validation, file operations, and server/client handling.
- **Explanation**: Centralizes definitions to ensure consistency across files. Includes necessary headers for socket programming, threading, and file operations.

2. server.c

- **Purpose**: Implements the server, handling client connections, authentication, and role-based operations.
- **Key Functions**:
  - **main()**: Initializes the server, sets up the socket, and listens for connections. Spawns a thread per client.
  - **client_handler()**: Handles individual client sessions, including authentication and role-based menu navigation.
  - **handle_admin()**, **handle_student()**, **handle_faculty()**: Implement role-specific menus and operations.
  - **send_with_length()**: Sends messages with a length prefix for reliable communication.
  - **log_message()**: Logs server events to `server.log`.
  - **ignore_sigpipe()**: Prevents server crashes on broken client connections.
- **Explanation**:
  - Uses `pthread` for concurrent client handling.
  - Authenticates users by reading `users.dat` with read locks.
  - Sends menus and processes client choices, calling file operations from `file_ops.c`.
  - Logs all significant events for debugging.

3. client.c

- **Purpose**: Implements the client, providing a command-line interface for users to interact with the server.
- **Key Functions**:
  - **main()**: Establishes a connection, handles login, and directs to role-specific handlers.
  - **handle_admin()**, **handle_student()**, **handle_faculty()**: Display menus, collect user inputs, and send them to the server.
  - **read_with_length()**: Reads messages with a length prefix.
  - **clear_input_buffer()**: Clears stdin to prevent input issues.
  - **ignore_sigpipe()**: Prevents client crashes on broken server connections.
- **Explanation**:
  - Connects to the server using TCP sockets.
  - Reads server prompts (e.g., login screen, menus) and sends user inputs.
  - Handles responses, displaying success/error messages.
  - Uses `fsync` to ensure data is sent immediately.

4. file_ops.c

- **Purpose**: Manages file-based data storage and concurrency for users, students, faculty, and courses.
- **Key Functions**:
  - **Validation**: `validate_id()`, `validate_name()`, `validate_password()`, `validate_number()` ensure input correctness.
  - **File Locking**: `read_lock()`, `write_lock()`, `unlock()` for thread-safe file access.
  - **User Operations**: `add_user()`, `change_password()`.
  - **Student Operations**: `add_student()`, `update_student()`, `activate_deactivate_student()`, `view_all_students()`.
  - **Faculty Operations**: `add_faculty()`, `update_faculty()`, `view_all_faculty()`.
  - **Course Operations**: `add_course()`, `update_course()`, `remove_course()`, `enroll_course()`, `unenroll_course()`, `view_all_courses()`, `view_faculty_courses()`, `view_enrolled_courses()`.

    - **initial_setup()**: Adds default admin, student, and faculty if `users.dat` is empty.
- **Explanation**:
    - Uses binary files for persistent storage.
    - Employs semaphores (`file_sem`) and file locks to prevent race conditions.
    - Returns dynamically allocated strings for view functions, which must be freed by the caller.
    - Validates inputs to prevent invalid data storage.

5. Compilation:

```
gcc -c server.c -o server.o

gcc -c client.c -o client.o

gcc -c file_ops.c -o file_ops.o

gcc server.o file_ops.o -o server -pthread

gcc client.o -o client -pthread
```

6. Example Screenshots (All choices not shown) :

```
ramya@Ramya:~/proj$ ./client
.....................Welcome Back to Academia :: Course Registration.....................
Login Type
Enter Your Choice { 1.Admin , 2.Professor, 3. Student } : 1
Client: Sending login choice: 1
Enter User ID: admin1
Client: Sending user ID: admin1
Enter Password: adminpass
Client: Sending password: adminpass
Login successful
Client: Raw bytes of response: 4c 6f 67 69 6e 20 73 75 63 63 65 73 73 66 75 6c 0a
Client: Login successful, proceeding to handle role
....... Welcome to 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: Client: Waiting for user input...
4
Client: Sending choice: 4
Client: Sent choice (1 bytes)
Client: Waiting for server response...
All Faculty:
1. ID: f1, Name: Dr. Smith
....... Welcome to 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: Client: Waiting for user input...
3
Client: Sending choice: 3
Client: Sent choice (1 bytes)
Enter Faculty ID: f2
Enter Faculty Name: teacher
Enter Password: 123
Client: Waiting for server response...
Faculty added successfully
....... Welcome to Admin Menu .......
1. Add Student
2. View Student Details
3. Add Faculty
4. View Faculty Details
```

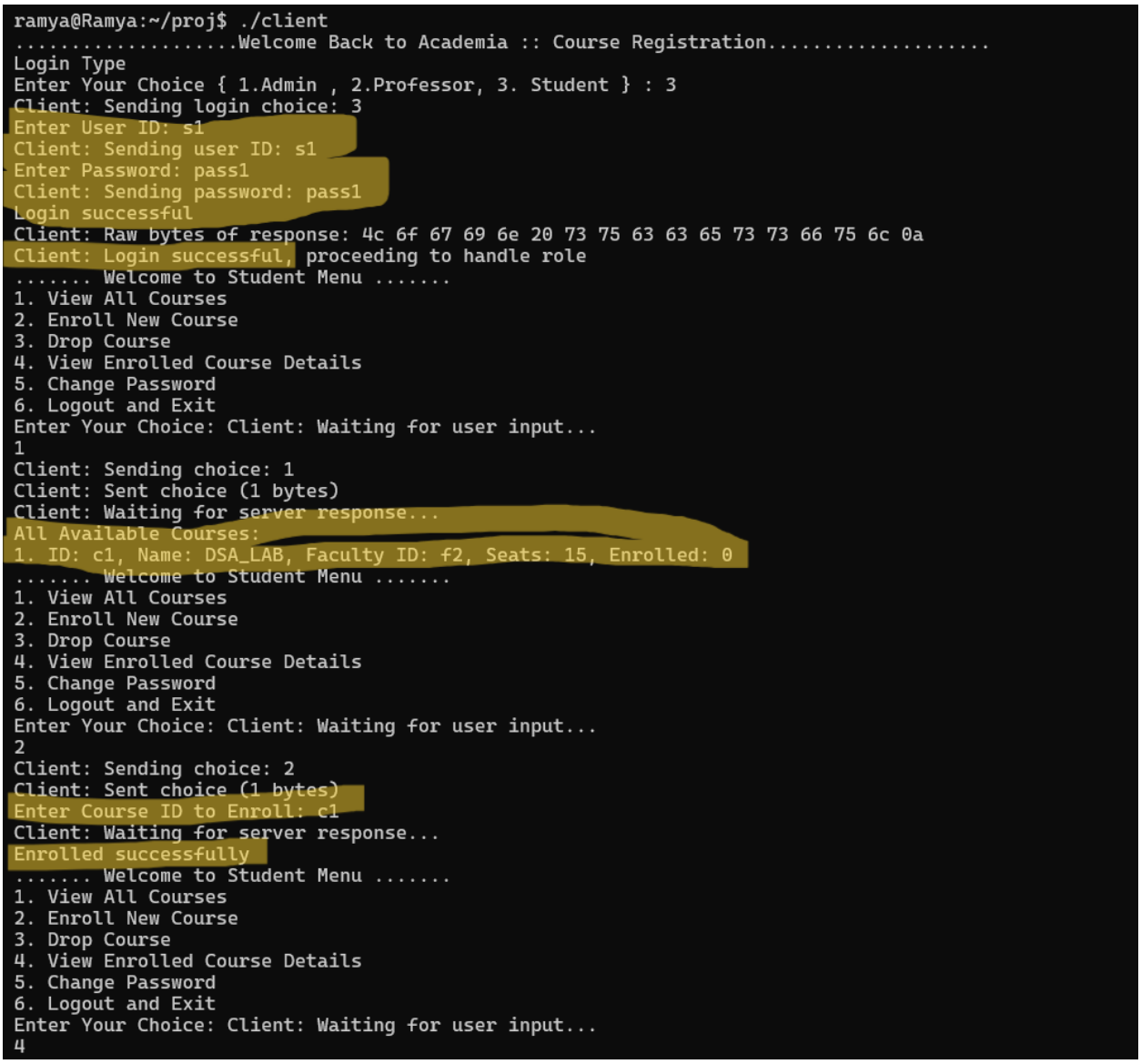
```
Enter Faculty ID: f2
Enter Faculty Name: teacher
Enter Password: 123
Client: Waiting for server response...
Faculty added successfully
....... Welcome to Admin Menu .......
1. Add Student
2. View Student Details
3. Add Faculty
4. View Faculty Details
5. Activate Student
6. Block Student
7. Modify Student Details
8. Modify Faculty Details
9. Logout and Exit
Enter Your Choice: Client: Waiting for user input...
4
Client: Sending choice: 4
Client: Sent choice (1 bytes)
Client: Waiting for server response...
All Faculty:
1. ID: f1, Name: Dr. Smith
2. ID: f2, Name: teacher
....... Welcome to 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: Client: Waiting for user input...
2
Client: Sending choice: 2
Client: Sent choice (1 bytes)
Client: Waiting for server response...
All Students:
1. ID: s1, Name: John, Status: Active
....... Welcome to 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: Client: Waiting for user input...
6
Client: Sending choice: 6
```

```
....... Welcome to 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: Client: Waiting for user input...
6
Client: Sending choice: 6
Client: Sent choice (1 bytes)
Enter Student ID to Block: s1
Client: Waiting for server response...
Student blocked successfully
....... Welcome to 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: Client: Waiting for user input...
2
Client: Sending choice: 2
Client: Sent choice (1 bytes)
Client: Waiting for server response...
All Students:
1. ID: s1, Name: John, Status: Blocked
....... Welcome to 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: Client: Waiting for user input...
9
Client: Sending choice: 9
Client: Sent choice (1 bytes)
Logout successful
Client: Admin logged out
ramya@Ramya:~/proj$
```

```
ramya@Ramya:~/proj$ ./client
..................Welcome Back to Academia :: Course Registration...................
Login Type
Enter Your Choice { 1.Admin , 2.Professor, 3. Student } : 2
Client: Sending login choice: 2
Enter User ID: f2
Client: Sending user ID: f2
Enter Password: 123
Client: Sending password: 123
Login successful
Client: Raw bytes of response: 4c 6f 67 69 6e 20 73 75 63 63 65 73 73 66 75 6c 0a
Client: Login successful, proceeding to handle role
....... Welcome to 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: Client: Waiting for user input...
1
Client: Sending choice: 1
Client: Sent choice (1 bytes)
Client: Waiting for server response...
Courses Offered by Faculty f2:
No courses offered.
....... Welcome to 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: Client: Waiting for user input...
2
Client: Sending choice: 2
Client: Sent choice (1 bytes)
Enter Course ID: c1
Enter Course Name: DSA
Enter Total Seats: 10
Client: Waiting for server response...
Course added successfully
....... Welcome to 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: Client: Waiting for user input...
1
Client: Sending choice: 1
Client: Sent choice (1 bytes)
```

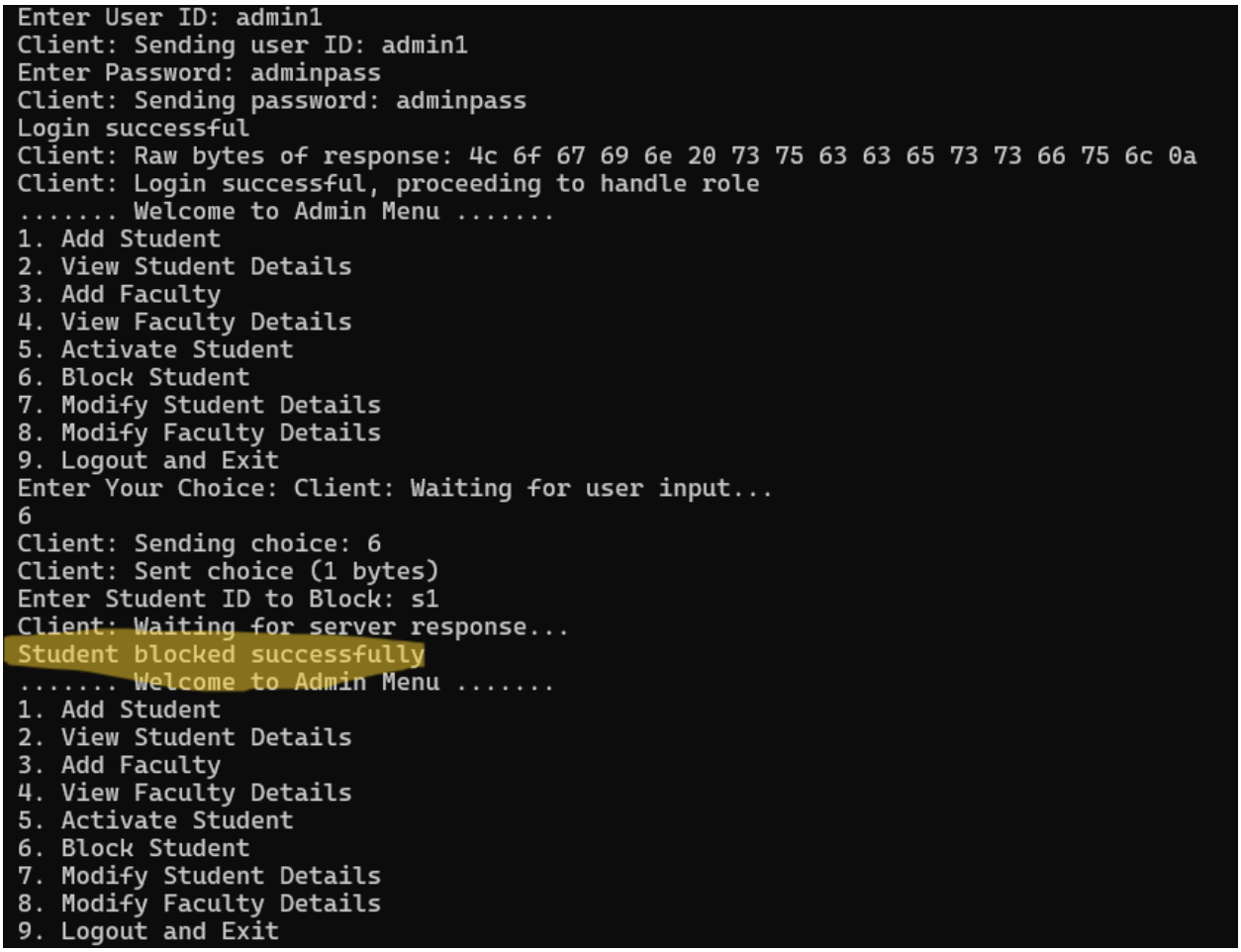
```
....... Welcome to Faculty Menu .......
1. View Offering Courses
2. Add New Course
3. Remove Course from Catalog
4. Update Course Details
5. Change Password
6. Logout and Exit
Enter Your Choice: Client: Waiting for user input...
1
Client: Sending choice: 1
Client: Sent choice (1 bytes)
Client: Waiting for server response...
Courses Offered by Faculty f2:
1. ID: c1, Name: DSA, Seats: 10, Enrolled: 0
....... Welcome to 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: Client: Waiting for user input...
4
Client: Sending choice: 4
Client: Sent choice (1 bytes)
Enter Course ID to Update: c1
Enter New Course Name: DSA_LAB
Enter New Total Seats: 15
Client: Waiting for server response...
Course updated successfully
....... Welcome to 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: Client: Waiting for user input...
1
Client: Sending choice: 1
Client: Sent choice (1 bytes)
Client: Waiting for server response...
Courses Offered by Faculty f2:
1. ID: c1, Name: DSA_LAB, Seats: 15, Enrolled: 0
....... Welcome to 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: Client: Waiting for user input...
6
```

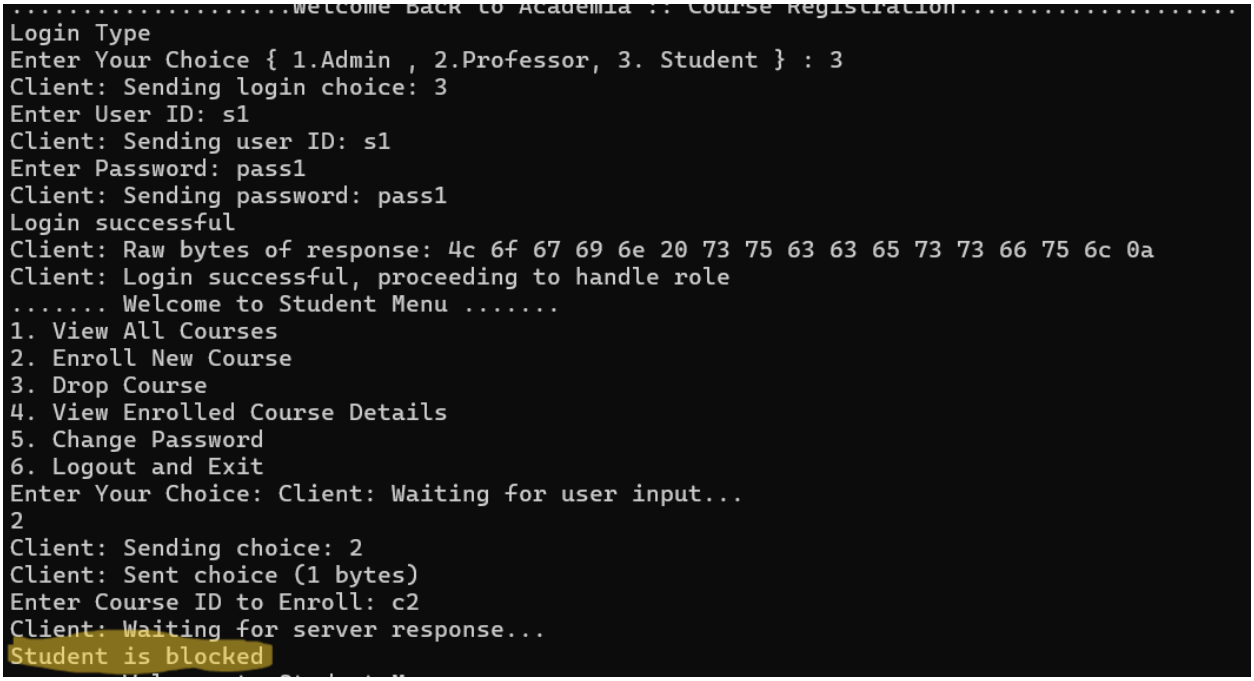
The student was activated before enrolling (not shown in this screenshot) , last 4 screenshots shows that blocked student are unable to enroll until activated.

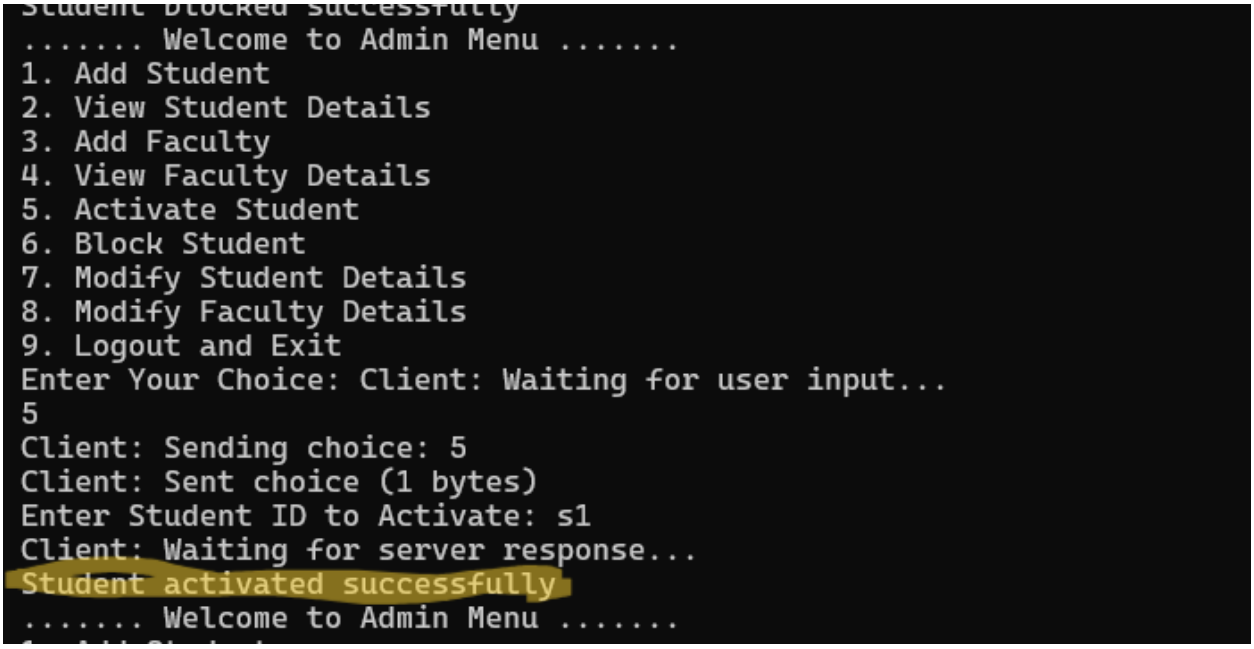


```
3. Drop Course
4. View Enrolled Course Details
5. Change Password
6. Logout and Exit
Enter Your Choice: Client: Waiting for user input...
2
Client: Sending choice: 2
Client: Sent choice (1 bytes)
Enter Course ID to Enroll: c1
Client: Waiting for server response...
Enrolled successfully
....... Welcome to Student Menu .......
1. View All Courses
2. Enroll New Course
3. Drop Course
4. View Enrolled Course Details
5. Change Password
6. Logout and Exit
Enter Your Choice: Client: Waiting for user input...
4
Client: Sending choice: 4
Client: Sent choice (1 bytes)
Client: Waiting for server response...
Enrolled Courses:
1. c1
....... Welcome to Student Menu .......
1. View All Courses
2. Enroll New Course
3. Drop Course
4. View Enrolled Course Details
5. Change Password
6. Logout and Exit
Enter Your Choice: Client: Waiting for user input...
1
Client: Sending choice: 1
Client: Sent choice (1 bytes)
Client: Waiting for server response...
All Available Courses:
1. ID: c1, Name: DSA_LAB, Faculty ID: f2, Seats: 15, Enrolled: 1
....... Welcome to Student Menu .......
1. View All Courses
2. Enroll New Course
3. Drop Course
4. View Enrolled Course Details
5. Change Password
6. Logout and Exit
Enter Your Choice: Client: Waiting for user input...
6
Client: Sending choice: 6
Client: Sent choice (1 bytes)
Logout successful
Client: Student logged out
ramya@Ramya:~/proj$
```

The blocking and multiuser functionality is shown, admin is logged through one terminal and student through the other.




.....Welcome Back to Academia :: Course Registration.....
Login Type
Enter Your Choice { 1.Admin , 2.Professor, 3. Student } : 3
Client: Sending login choice: 3
Enter User ID: s1
Client: Sending user ID: s1
Enter Password: pass1
Client: Sending password: pass1
Login successful
Client: Raw bytes of response: 4c 6f 67 69 6e 20 73 75 63 63 65 73 73 66 75 6c 0a
Client: Login successful, proceeding to handle role
..... Welcome to Student Menu .....
1. View All Courses
2. Enroll New Course
3. Drop Course
4. View Enrolled Course Details
5. Change Password
6. Logout and Exit
Enter Your Choice: Client: Waiting for user input...
2
Client: Sending choice: 2
Client: Sent choice (1 bytes)
Enter Course ID to Enroll: c2
Client: Waiting for server response...
Student is blocked


Student Blocked Successfully
..... Welcome to 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: Client: Waiting for user input...
5
Client: Sending choice: 5
Client: Sent choice (1 bytes)
Enter Student ID to Activate: s1
Client: Waiting for server response...
Student activated successfully
..... Welcome to Admin Menu .....

```
Student is blocked
....... Welcome to Student Menu .......
1. View All Courses
2. Enroll New Course
3. Drop Course
4. View Enrolled Course Details
5. Change Password
6. Logout and Exit
Enter Your Choice: Client: Waiting for user input...
2
Client: Sending choice: 2
Client: Sent choice (1 bytes)
Enter Course ID to Enroll: c2
Client: Waiting for server response...
Enrolled successfully
```