**OS-LAB PROJECT REPORT**

**SUBMISSION BY: (IMT2023505, GARV RAJPUT)**

**1. Problem Statement**

Design and Development of Course Registration Portal (Academia).

Description: The project aims to develop a Academia Portal that is user-friendly and

multifunctional.

**2. Project Overview**

This project implements a client-server Academia Portal using C, sockets, and file-based storage. The server manages all data and concurrency, while clients interact via role-based menus. The system supports:

- Admin: User management, consistency checks

- Faculty: Course management

- Student: Course enrollment and viewing

- Concurrency: File locking and a global flag to prevent race conditions

**3. Main Code Files and Important Variables**

- server.c: Main server logic, request handlers, concurrency control

- `course\_file\_busy`: Global flag for course file locking

- faculty.c: Faculty client menu and actions

- student.c: Student client menu and actions

- admin.c: Admin client menu and actions

- common.h: Shared data structures, protocol message types, constants

- syscalls.c/h: File operations, locking helpers

- session\_utils.c: Checks for duplicate logins

**4. Features by File**

server.c:

- Handles all client requests (admin, faculty, student)

- File-based storage for users, courses, enrollments

- File locking for concurrency

- Global flag (`course\_file\_busy`) for strict course file access

- Logging of major actions

faculty.c:

- Add, update, remove courses

- View enrollments and students in courses

- Change password

- Lock protocol for course file updates

student.c:

- Enroll/drop courses

- View enrolled and all available courses

- Change password

- Handles course file lock errors gracefully

admin.c:

- Add, update, delete users

- Block/activate students

- Consistency check for enrollments

- Change admin password

common.h:

- Data structures: User, Course, Enrollment

- Protocol message types (e.g., MSG\_ADD\_COURSE, MSG\_LIST\_COURSES)

- Constants for file names, sizes

syscalls.c/h:

- File operations using system calls

- File locking helpers (lock\_file, unlock\_file)

- User/course/enrollment CRUD functions

**5. How to Build and Run**

1. Build the server and client:

We have included a makefile which you can use directly by typing

$ make

2. Start the server:

$ ./server

3. In separate terminals, start clients:

$ ./client <server\_ip>

4. Log in as Admin, Faculty, or Student and use the menu-driven interface.

- The system will enforce concurrency: if a faculty is adding/updating a course, students cannot view courses until the operation is complete.

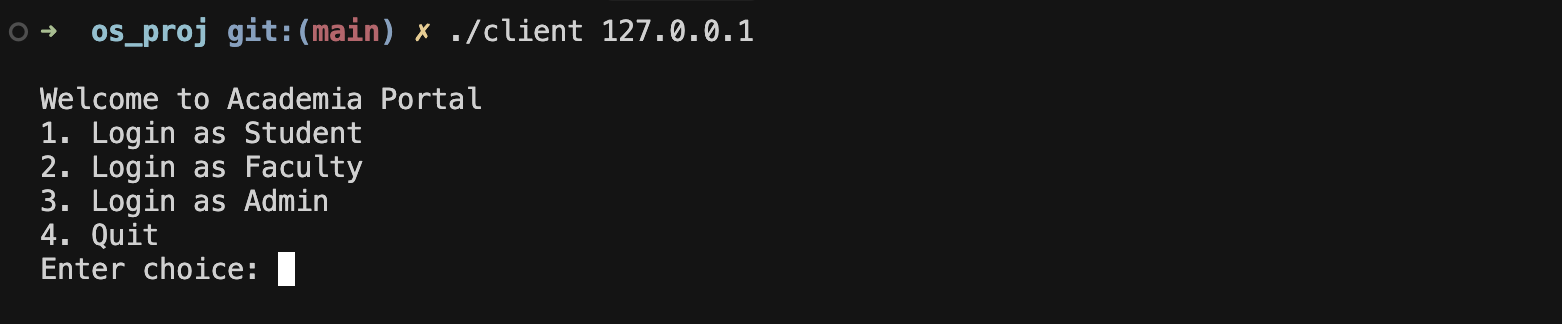
- All data is stored in files (users.dat, courses.dat, enrollments.dat, session\_utils.dat).

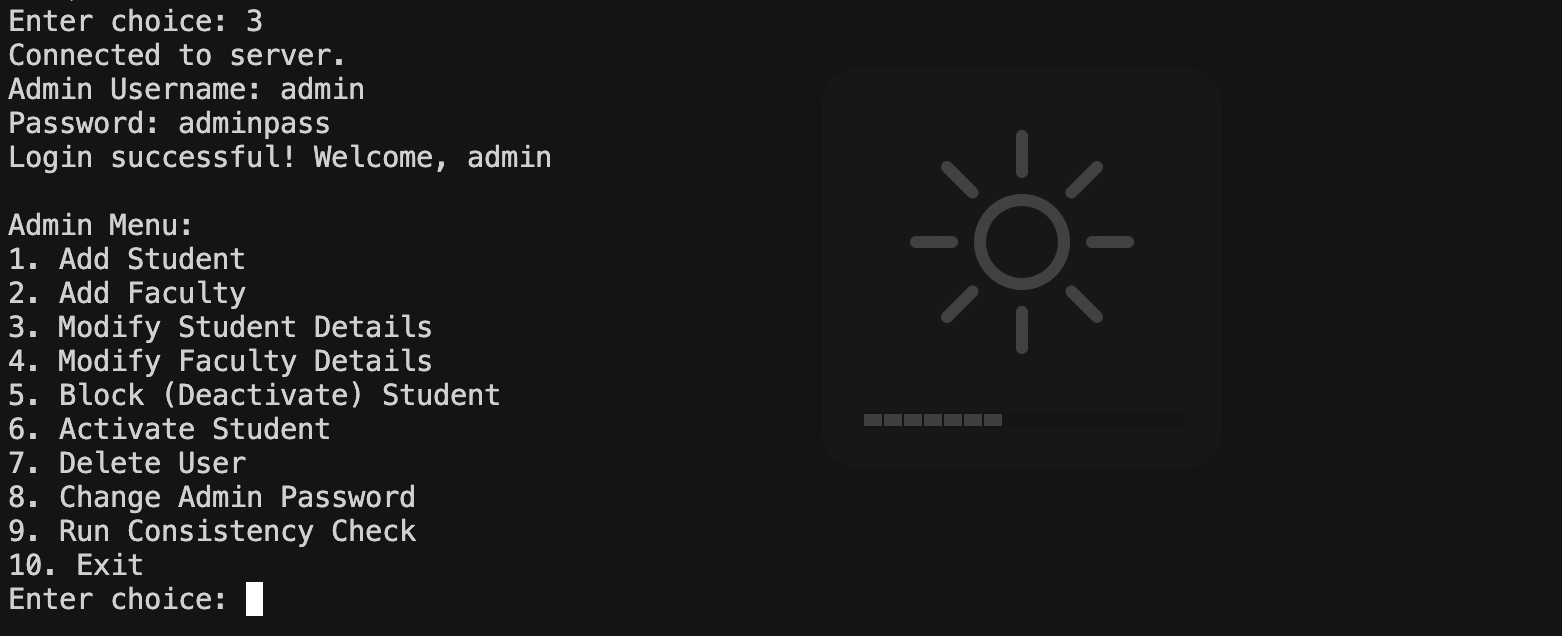
5. Clear the build after use by:

$ make clean

TERMINAL SCREENSHOTS

On-startup:

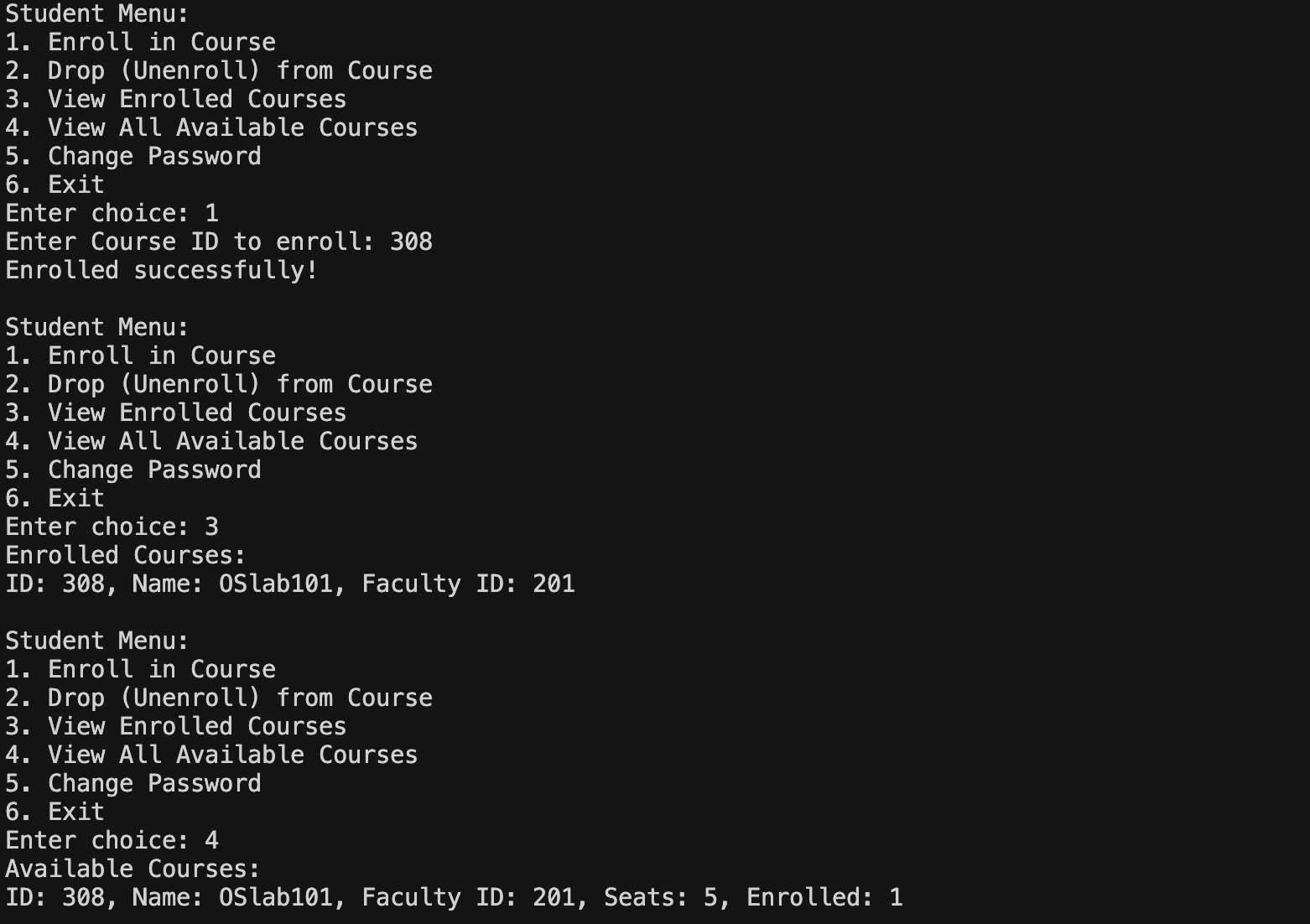


Admin:

Student:



Enrolling in a course

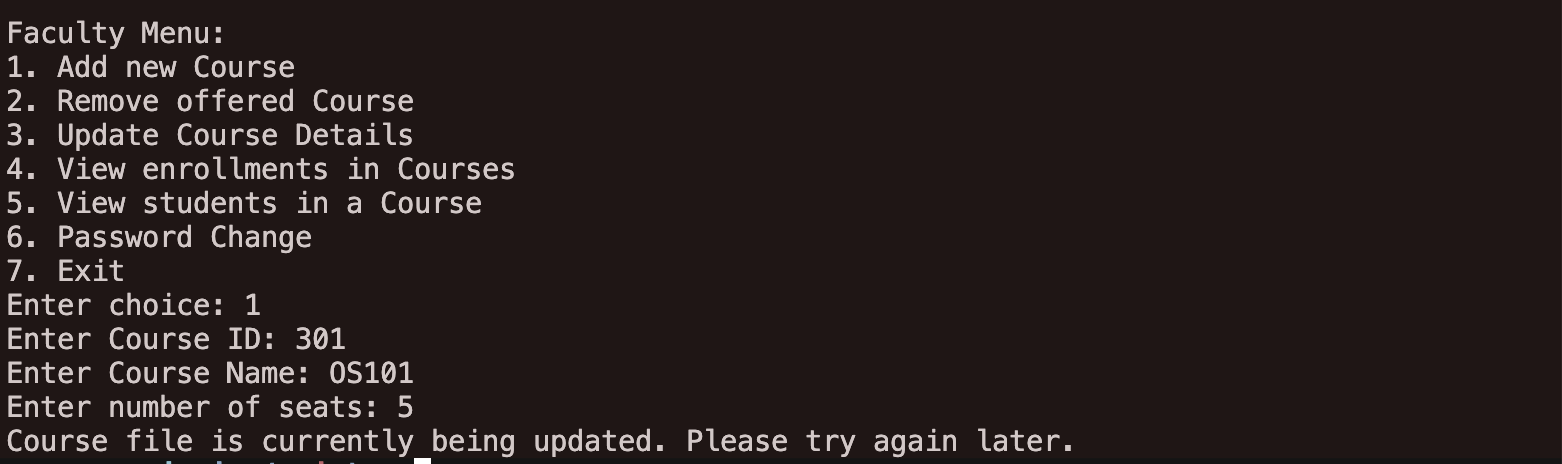


De-enrolling in a course:



Faculty:

TRYING TO ACCESS LOCKED COURSE FILE (some other faculty accessing it right now):



Normal Output:

