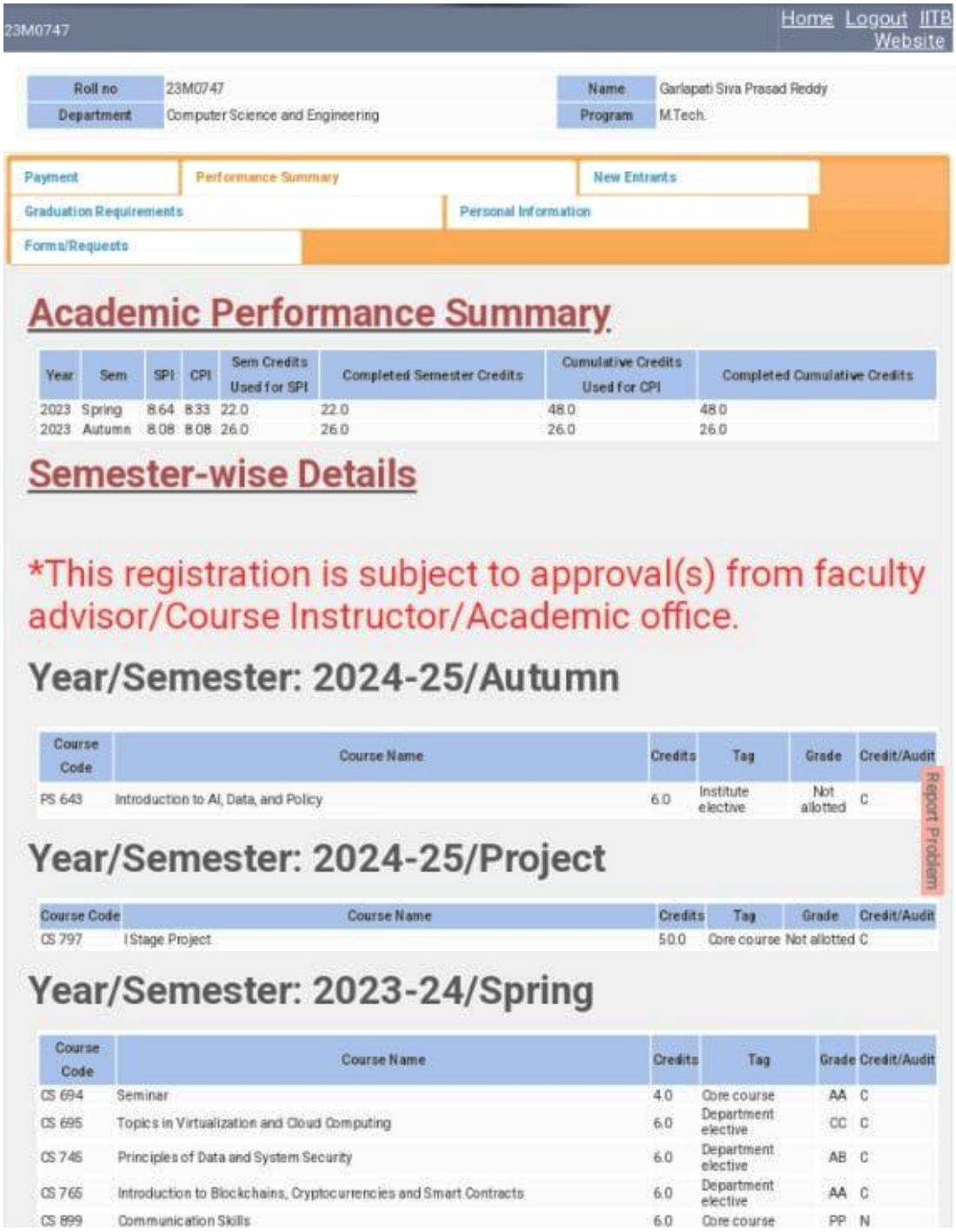
(CS695) Container Management Tool

Siva Prasad Reddy Garlapati

23m0747

This is screenshot for verification. **2nd course** in 2023-24/ Spring **(CS695)**



23M0747 Home Logout IITB Website

Roll no	23M0747	Name	Garlapati Siva Prasad Reddy
Department	Computer Science and Engineering	Program	M.Tech.

Payment | Performance Summary | New Entrants | Graduation Requirements | Personal Information | Forms/Requests

Academic Performance Summary

Year	Sem	SPI	CPI	Sem Credits Used for SPI	Completed Semester Credits	Cumulative Credits Used for CPI	Completed Cumulative Credits
2023	Spring	8.64	8.33	22.0	22.0	48.0	48.0
2023	Autumn	8.08	8.08	26.0	26.0	26.0	26.0

Semester-wise Details

*This registration is subject to approval(s) from faculty advisor/Course Instructor/Academic office.

Year/Semester: 2024-25/Autumn

Course Code	Course Name	Credits	Tag	Grade	Credit/Audit
PS 643	Introduction to AI, Data, and Policy	6.0	Institute elective	Not allotted	C

Report Problem

Year/Semester: 2024-25/Project

Course Code	Course Name	Credits	Tag	Grade	Credit/Audit
CS 797	I Stage Project	50.0	Core course	Not allotted	C

Year/Semester: 2023-24/Spring

Course Code	Course Name	Credits	Tag	Grade	Credit/Audit
CS 694	Seminar	4.0	Core course	AA	C
CS 695	Topics in Virtualization and Cloud Computing	6.0	Department elective	CC	C
CS 745	Principles of Data and System Security	6.0	Department elective	AB	C
CS 765	Introduction to Blockchains, Cryptocurrencies and Smart Contracts	6.0	Department elective	AA	C
CS 899	Communication Skills	6.0	Core course	PP	N

This is a deep dive into the fundamentals of containerization by leveraging key Linux features such as namespaces, chroot, and cgroups. The objective is to build a simplified version of a container management tool akin to Docker, providing a hands-on approach to understanding the core components that underlie modern containerization technologie**s.**

1. Environment Setup

Objective:
To set up and configure the virtual environment necessary for containerization tasks.

Approach:

- **Virtualization Tool Selection:**
  Depending on the hardware architecture, VirtualBox is utilized for x86_64 users, while UTM is chosen for Apple Silicon users. These tools create isolated environments for running the provided Virtual Machine (VM).
- **VM Configuration:**
  The VM comes pre-configured with essential tools and libraries needed for the assignment. The provided credentials include `cs695` as the username and `1234` as the password for both user and root accounts.
- **System Preparation:**
  After logging in, the VM is updated to ensure all packages are current. This step guarantees compatibility and the presence of necessary dependencies for subsequent tasks.

2. Namespaces with System Calls

Objective:
To explore process isolation through the use of Linux namespaces, particularly focusing on the UTS (Unix Time-sharing System) and PID (Process ID) namespaces.

Approach:

- **Creating a New Child Process in a New UTS and PID Namespace:**
  - **Clone System Call:**
    The `clone` system call is employed to fork a new process that operates in a separate UTS and PID namespace. The `CLONE_NEWPID` and `CLONE_NEWUTS` flags are used to ensure the child process is isolated in terms of both hostname and process ID space. The child process will start as PID 1 within its namespace, simulating the behavior of an init process in a container.

    - **Child Process Configuration:**
      Within this isolated namespace, the child process can be assigned a different hostname from the parent, ensuring complete UTS namespace isolation. This mimics the container's ability to have its network identity.
- **Creating a Second Child Process in the Same UTS and PID Namespace:**
    - **setns System Call:**
      The second child process is initially created in the parent's namespace. Using the `setns` system call, it is reassigned to the first child's UTS and PID namespaces. This method allows the second child process to inherit the same isolated environment, maintaining consistency in hostname and process ID space.

3. CLI Containers

Objective:
To manually create and manage a basic container using command-line tools, thereby gaining an understanding of filesystem isolation, process isolation, and resource constraints.

Approach:

- **Filesystem Isolation:**
    - **chroot Command:**
      The `chroot` command is utilized to change the root directory of the process, effectively isolating the process's view of the filesystem. This step ensures that the process only sees a restricted set of files and directories, much like a container's root filesystem.
    - **Dependency Management:**
      It's essential to ensure that all required dependencies of the containerized application (referred to as `container_prog`) are available within the new root directory. This might involve copying libraries and binaries into the isolated environment.
- **PID and UTS Namespace Isolation:**
    - **unshare Command:**
      The `unshare` command is used to create new UTS and PID namespaces, effectively isolating the process's view of its hostname and process ID space. The process now sees itself as PID 1, similar to an init process in a container, and can be assigned a unique hostname (`new_hostname`).
- **Resource Provisioning with cgroups:**
    - **cgroups v2 Implementation:**
      A new cgroup is created to manage and limit the resources available to processes within the container. Specifically, CPU usage is restricted to 50%, ensuring that the container does not overconsume host resources. Processes are then moved into this cgroup, enforcing the configured resource limits.

4. Containers in the Wild

Objective:
To create a more sophisticated container management tool, similar to Docker, using the `conductor.sh` script. This involves implementing container creation, execution, and networking features.

Approach:

- **Implementing `run`:**
  - **Container Execution:**
    The `run` command is implemented to start a container by invoking the `unshare` command to create new namespaces for PID, UTS, and others. The `chroot` command is used to change the root filesystem, isolating the container's environment from the host.
  - **Filesystem Mounting:**
    Essential filesystems, such as `/proc`, are mounted within the container to enable standard process operations. This step is critical for ensuring that the containerized process has access to necessary system information.
- **Implementing `exec`:**
  - **Command Execution:**
    The `exec` command allows running additional commands inside an existing container's namespaces. By attaching to the container's namespaces using `setns`, new commands can be executed in the same isolated environment.
- **Implementing Networking:**
  - **Virtual Ethernet (veth) Link Creation:**
    Networking is established by creating a virtual Ethernet pair (veth link), which connects the container's network namespace to the host's network namespace. The veth interface within the container is assigned an IP address, enabling communication with both the host and external networks. This setup simulates the networking capabilities of a Docker container, allowing for connectivity and data exchange.