Project Proposal

Submitted by	Saad Ali (55780) Ghazanfar Pasha (55590) M. Roshaan Idrees (56177)
Submitted To	Sir Shahzad Ahmed Khan
Section	BSSE - 5-2

Riphah International University Fall 2025 Faculty of Computing

Project Title

CoreOS: An Interactive Educational Operating System with CPU & Memory Visualization

Introduction

Understanding how operating systems work internally is often challenging because most real-world systems (like Windows or Linux) are large and complex. Students typically learn concepts such as process states, CPU scheduling, and memory allocation theoretically, without seeing how the OS actually performs these operations.

CoreOS aims to solve this problem.

CoreOS is a **lightweight educational operating system** built from scratch using low-level programming. It boots directly on hardware and allows users to **interactively observe** how the core components of an OS function internally. The system provides real-time visualization of **CPU scheduling**, **process transitions**, and **memory usage**, helping students understand textbook concepts practically and intuitively.

Objectives

The primary objective of this project is to **build a fully functional mini operating system** that helps users learn core OS concepts through direct interaction and visualization.

Specific Objectives

- Implement a **custom bootloader** and **kernel** to run the OS directly on hardware.
- Provide a **command-line shell** for interacting with the system.
- Implement and visualize **multiple CPU scheduling algorithms**, including:
    - FCFS (First Come First Serve)
    - SJF (Shortest Job First)

    - Round Robin
    - Priority Scheduling

- Show **process state transitions** graphically between:
  READY → RUNNING → WAITING → TERMINATED
- Implement **memory management visualization** to display allocated and free blocks.
- Ensure the OS can run in **QEMU, VirtualBox, or real hardware**.

Tools and Technologies

Tool / Technology	Purpose
NASM (Assembly)	Bootloader and hardware initialization
GCC (32-bit Mode)	Compiling kernel and C code
GRUB Bootloader	Loading the operating system at startup
QEMU / VirtualBox	Testing and debugging the OS
Makefile	Automated build and compilation

Expected Output

At the end of the project, we will have a **working OS** that can:

- Boot independently without Windows/Linux.
- Run its own terminal-based user interface.
- Demonstrate how the CPU schedules processes.
- Visually show memory allocation and state changes.
- Allow students to test and compare algorithm behavior interactively.

Conclusion

CoreOS is designed to **bridge the gap between theory and practice** in operating system learning.
Unlike simulators, CoreOS runs directly on the machine and clearly shows how scheduling and memory management truly work inside a kernel. The project provides students with **hands-on, real-world OS development experience** and can be reused for **future academic learning, teaching, and research work**