Information Tech.
DEPARTMENT

Introduction of Operating System

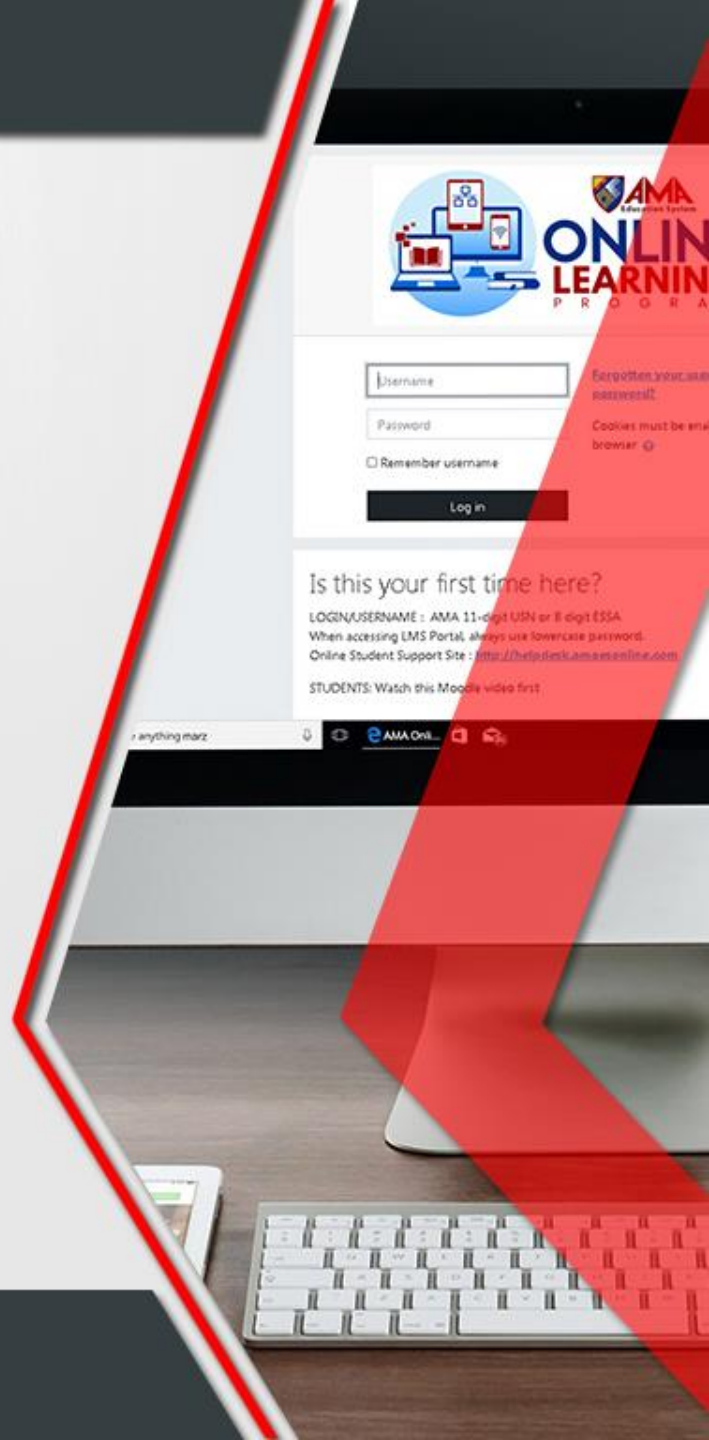
PHILANDER G. OSO
Instructor

Computer System Components

1. Hardware
   - ➢ Provides basic computing resources (CPU, memory, I/O devices).
2. Software
   - ❖ **Operating System**
     - ➢ Controls and coordinates the use of hardware among application programs.
   - ❖ Application Programs
     - ➢ Solve computing problems of users (compilers, database systems, video games, business programs).
3. Peopleware
   - ➢ Users, Programmers, etc

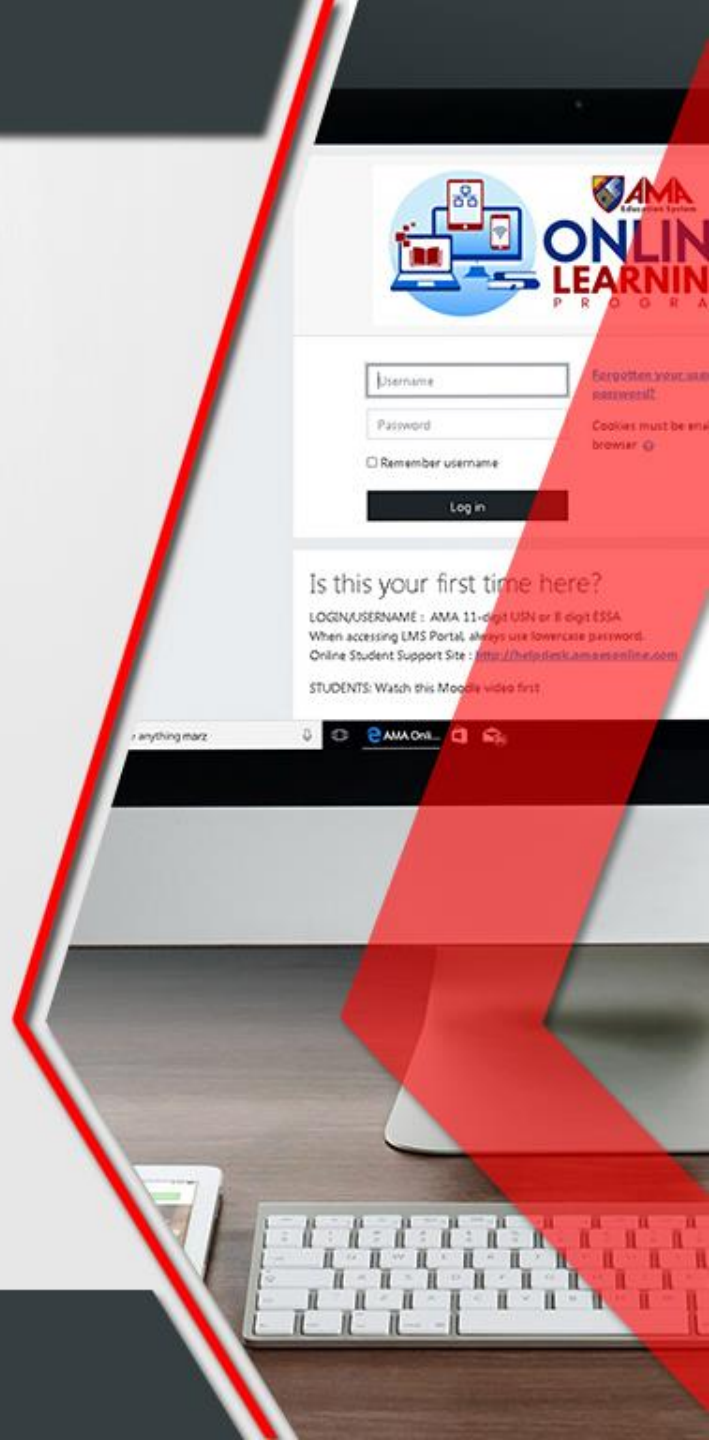
Software

- Programs that control the operations of the computer and its devices
  - ➢Starting up the computer
  - ➢Opening, executing, and running applications
  - ➢Disk formatting
  - ➢File compression
  - ➢Backups

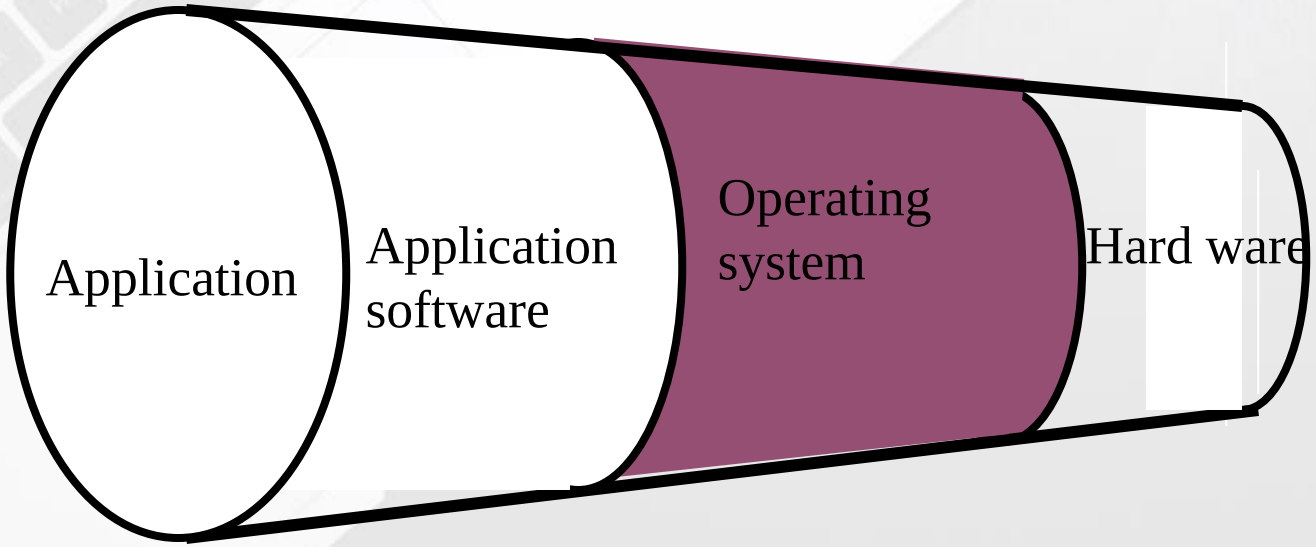


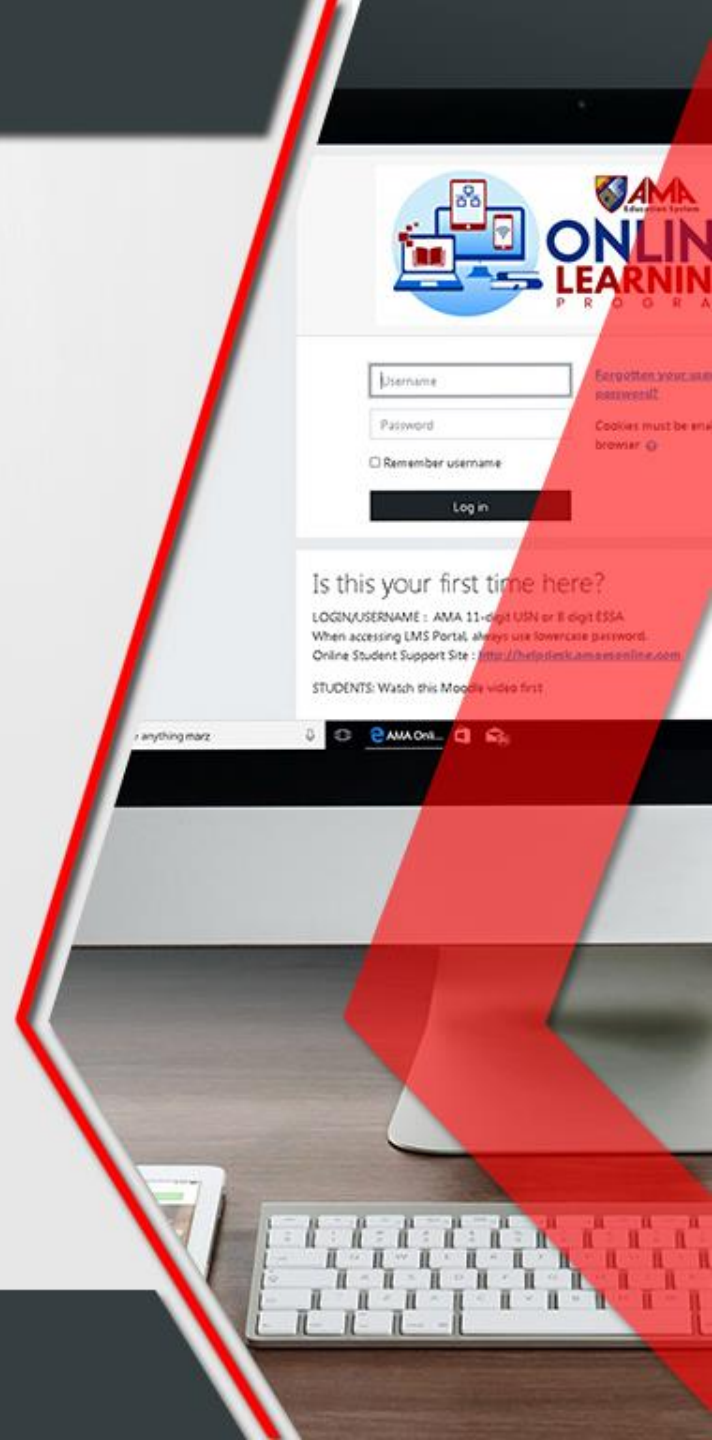
LIVE ONLINE CLASS

- Computer Software
  - System Software
    - User Interface
    - Operating System
    - Utilities
  - Application Software
    - Packaged Software
    - Custom Software
    - Shareware
    - Cardware
    - Freeware
    - Public Domain



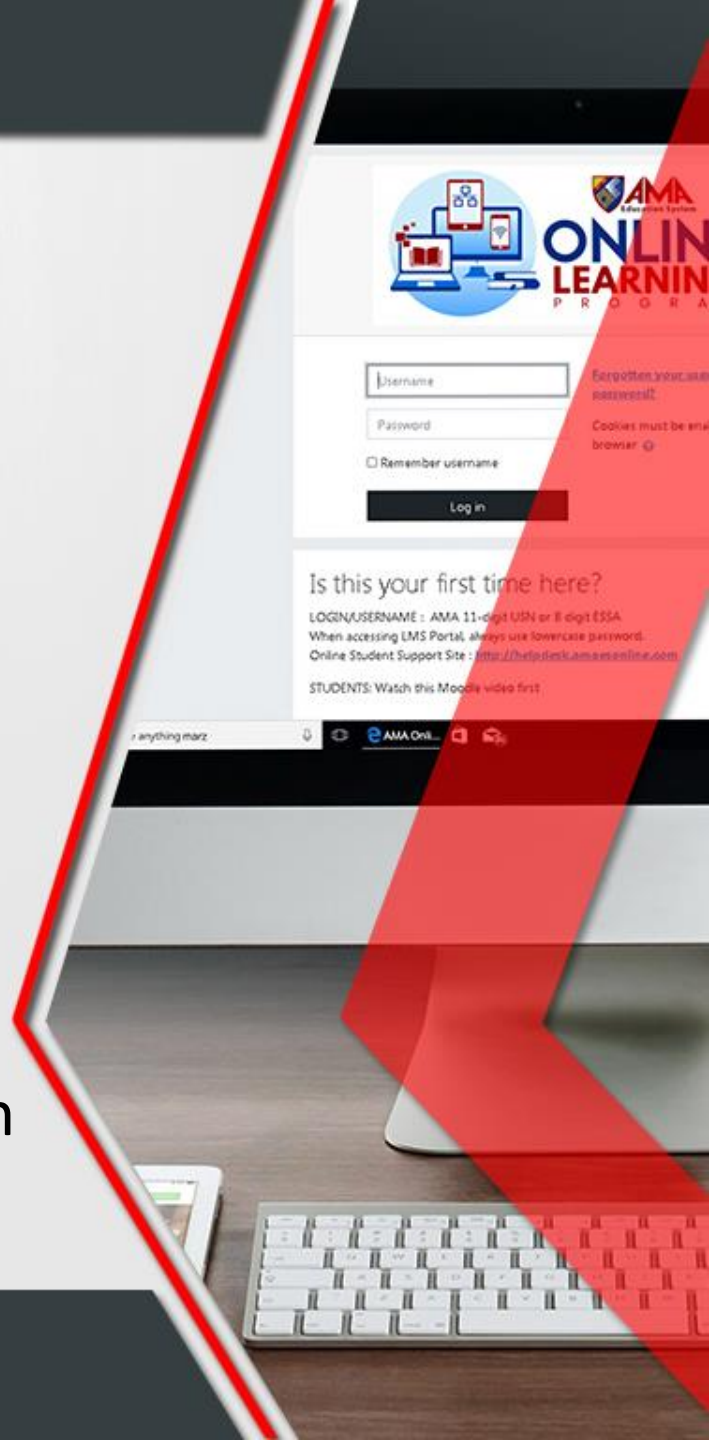
Operating System Position

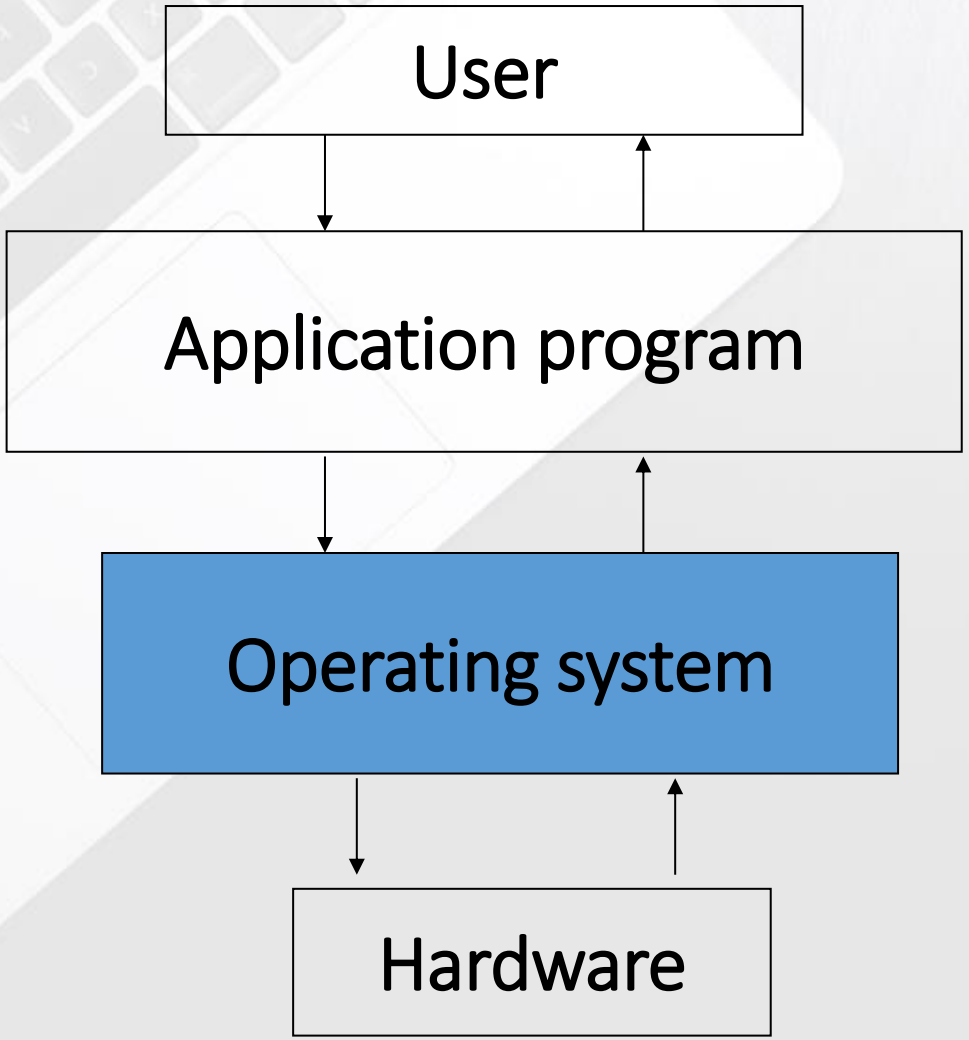


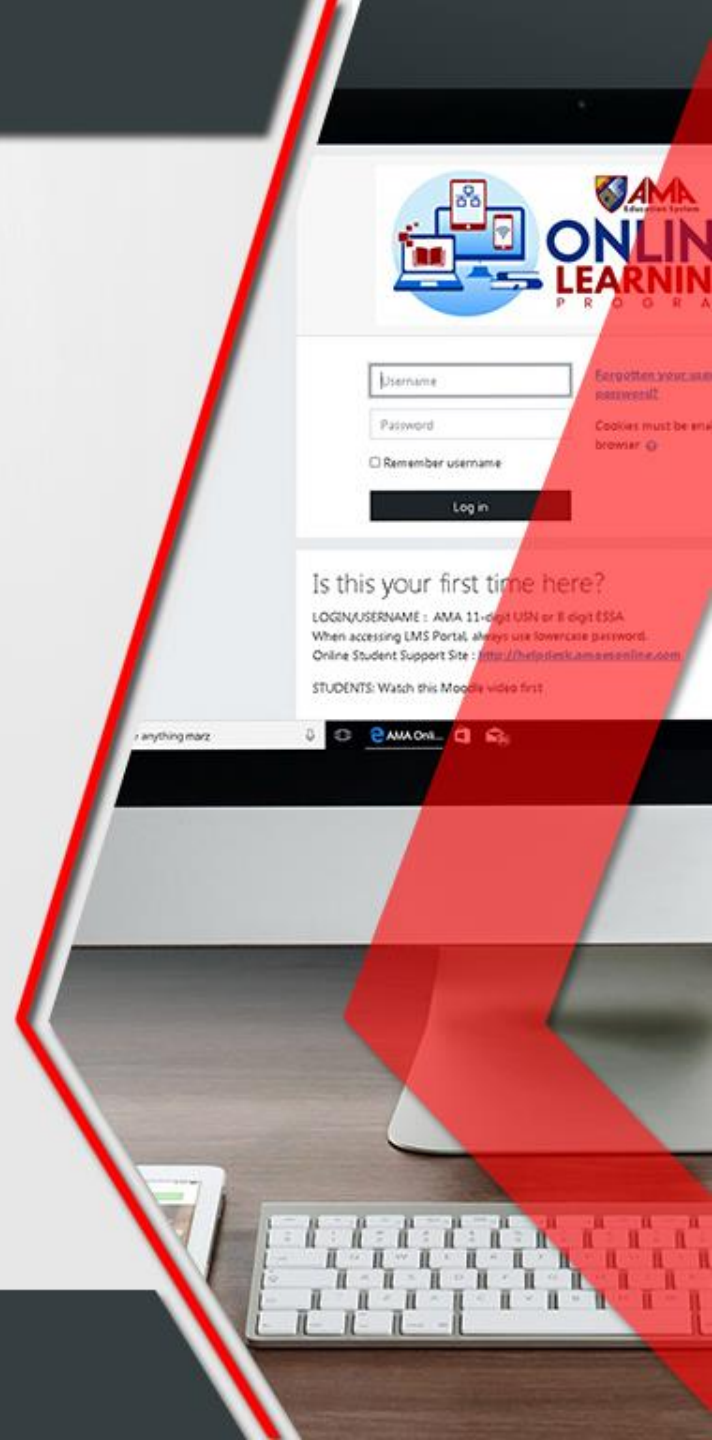


What is an Operating System?

- A software that acts an intermediary between the applications and computer hardware.
- A set of system software routines that sits between the application program and the hardware.
- Serves as a hardware/software interface, application programmers and users rarely communicate directly with hardware (simplifies programming).
- Acts as repository for command, shared routines, and defines a platform for constructing and executing application software.
- The operating system (OS) controls almost all functions on a computer.

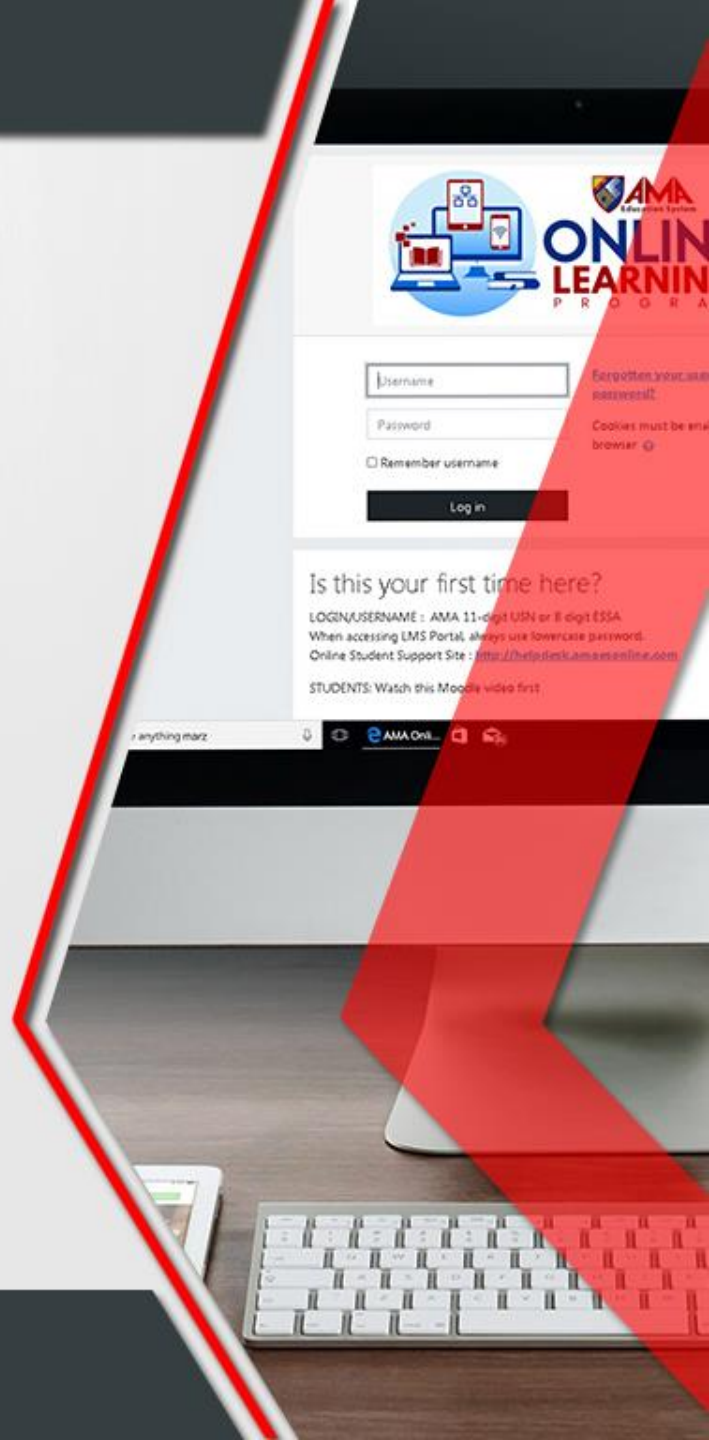



User
Application program
Operating system
Hardware


Username
Password
Remember username
Log in
Is this your first time here?

Primary purposes of an OS:

1. Resource Manager
2. Process Manager
3. Memory Manager
4. File System Manager
5. Device Manager
6. User Interface
7. Security and Protection

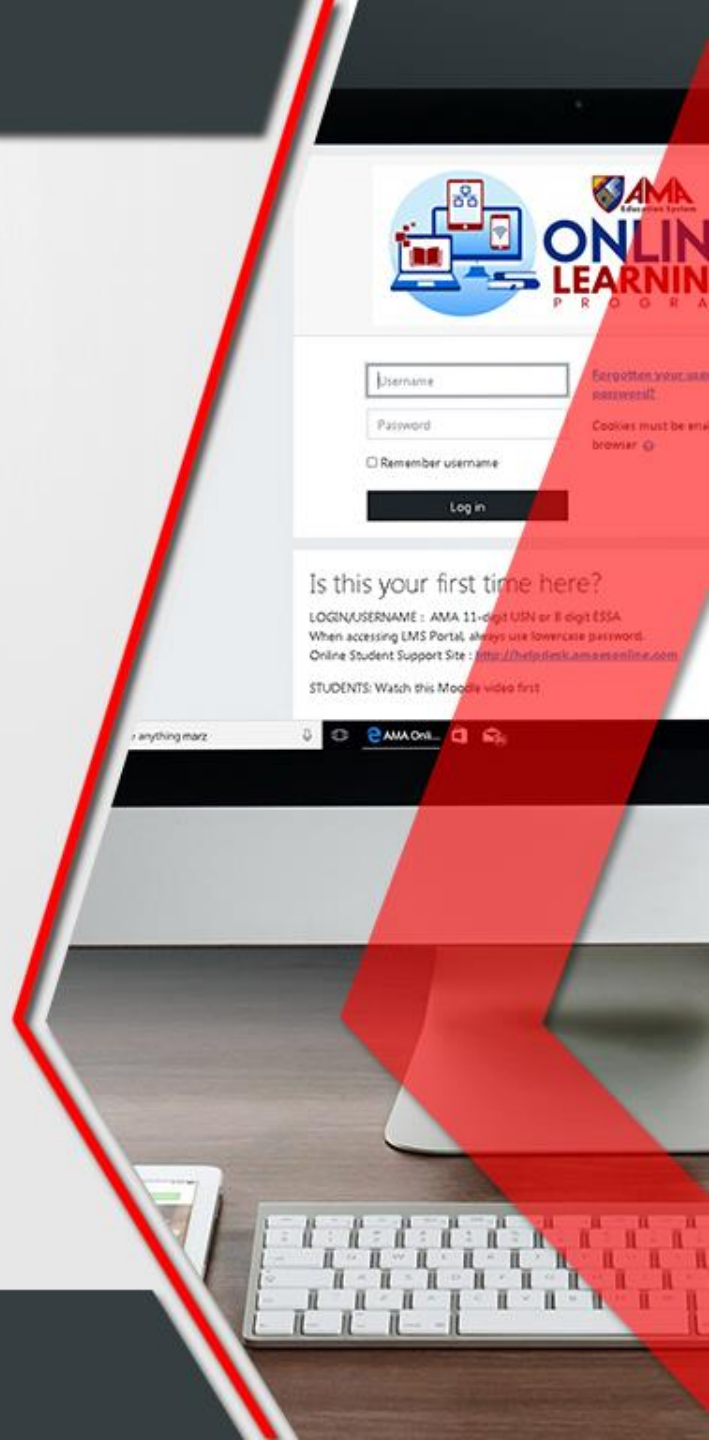


Resource Management:

- The OS manages hardware resources such as CPU, memory, storage devices, and input/output (I/O) devices. It allocates these resources efficiently among different processes and users to ensure optimal performance and utilization.

Process Management:

- It facilitates the creation, scheduling, and execution of processes or programs.
- The OS manages process synchronization, communication, and inter-process coordination to ensure smooth operation and prevent conflicts.

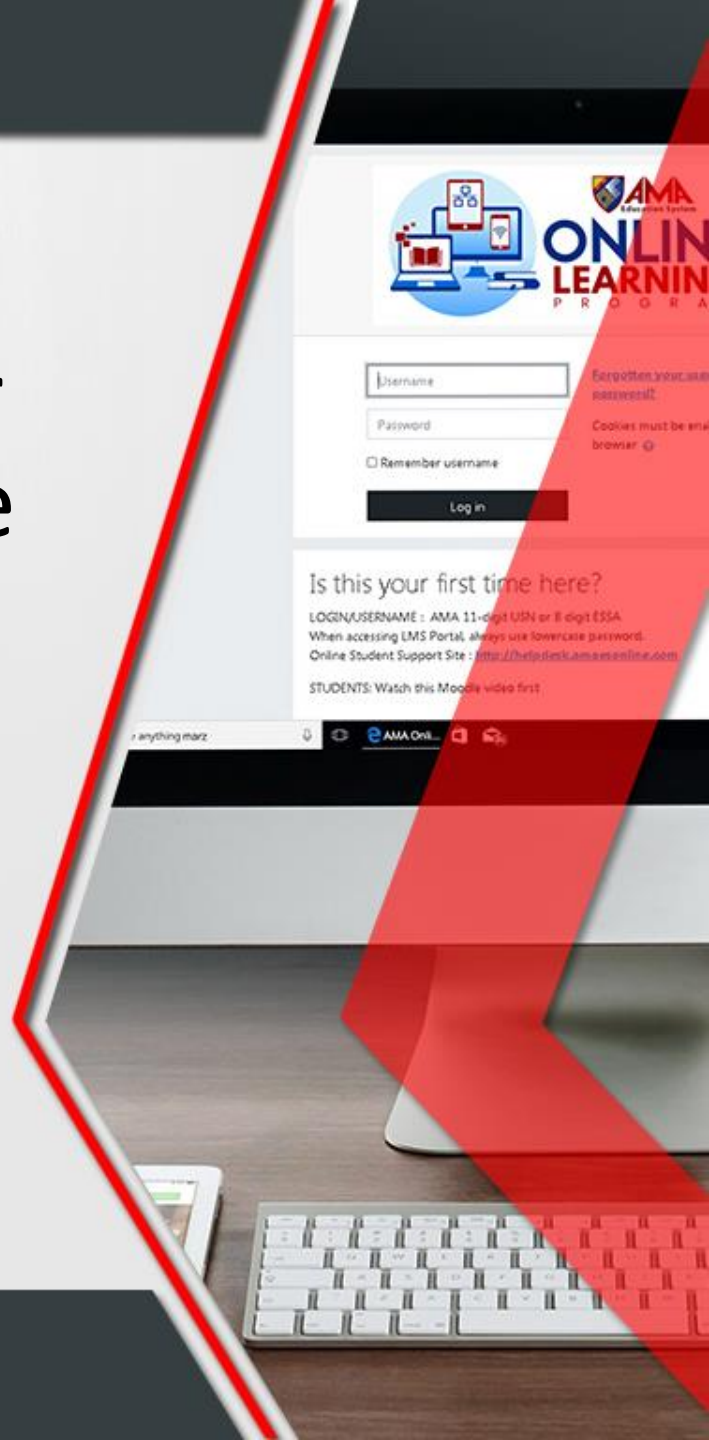


Memory Management:

- The OS allocates and deallocates memory space for processes, manages virtual memory, and handles memory protection to prevent unauthorized access and ensure data integrity.

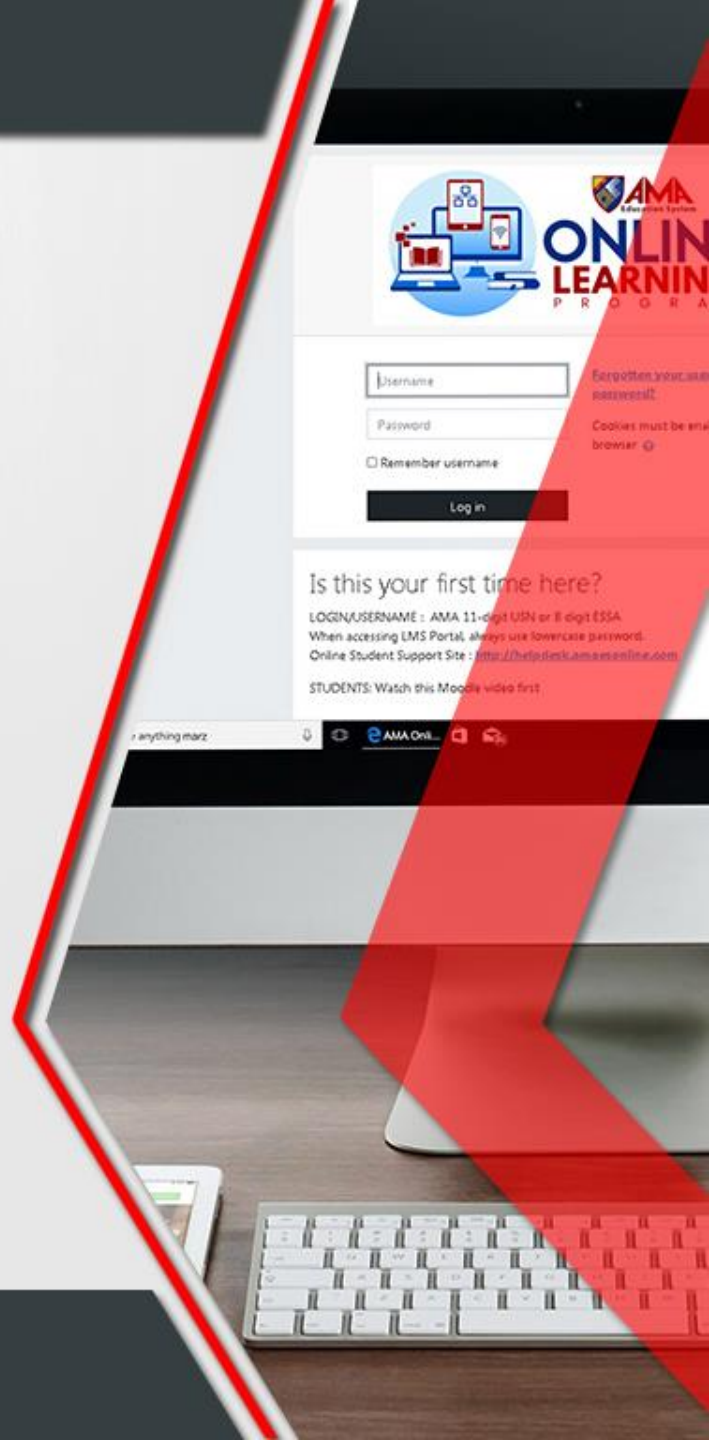
File System Management:

- It provides a hierarchical structure for organizing and storing files on storage devices. The OS handles file creation, deletion, access control, and data storage/retrieval operations.



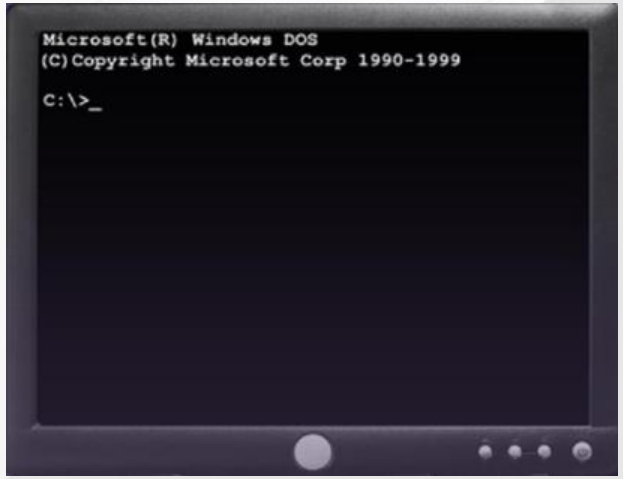
Device Management:

- The OS controls communication with input/output devices such as keyboards, mice, printers, and network interfaces. It provides device drivers and manages device access, ensuring efficient data transfer and device utilization.



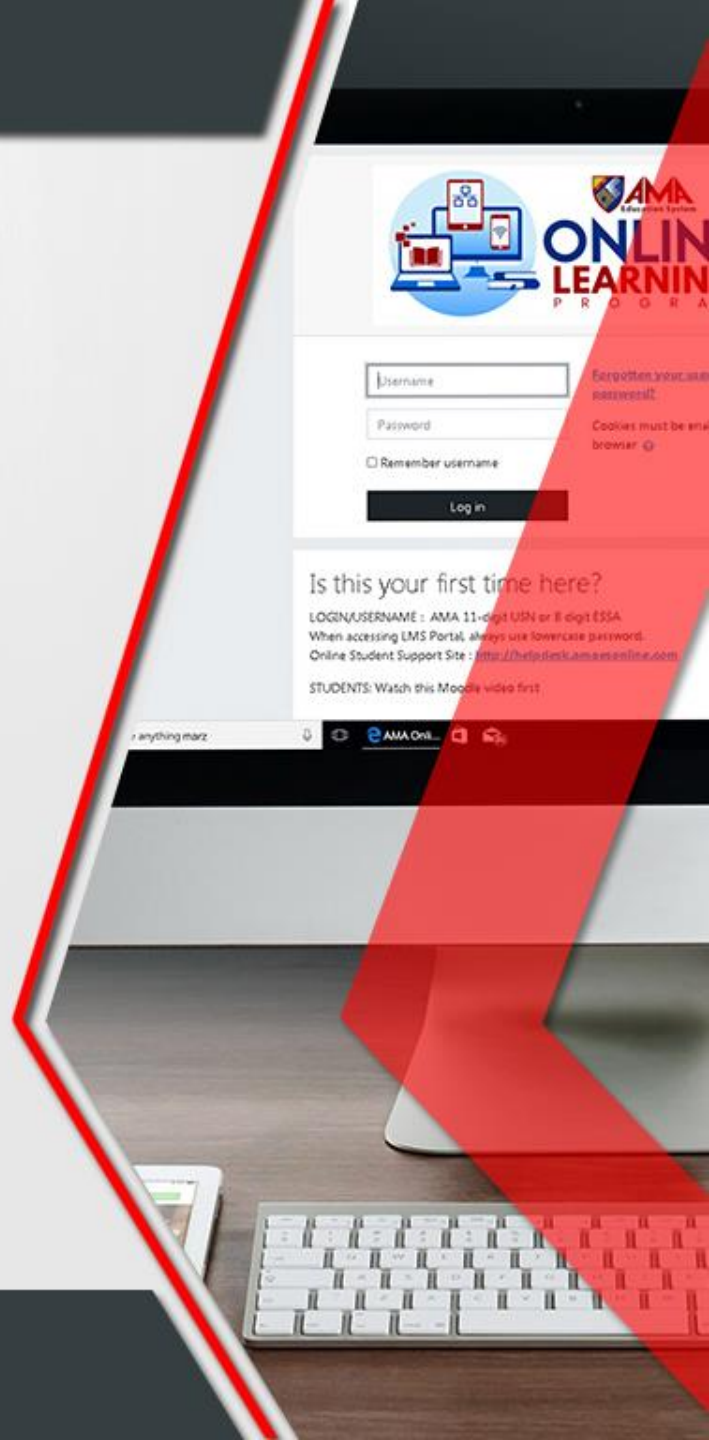
User Interface:

- The OS provides a user-friendly interface for interacting with the computer system. This can include command-line interfaces (CLI), graphical user interfaces (GUI), or a combination of both, depending on the OS and user preferences.



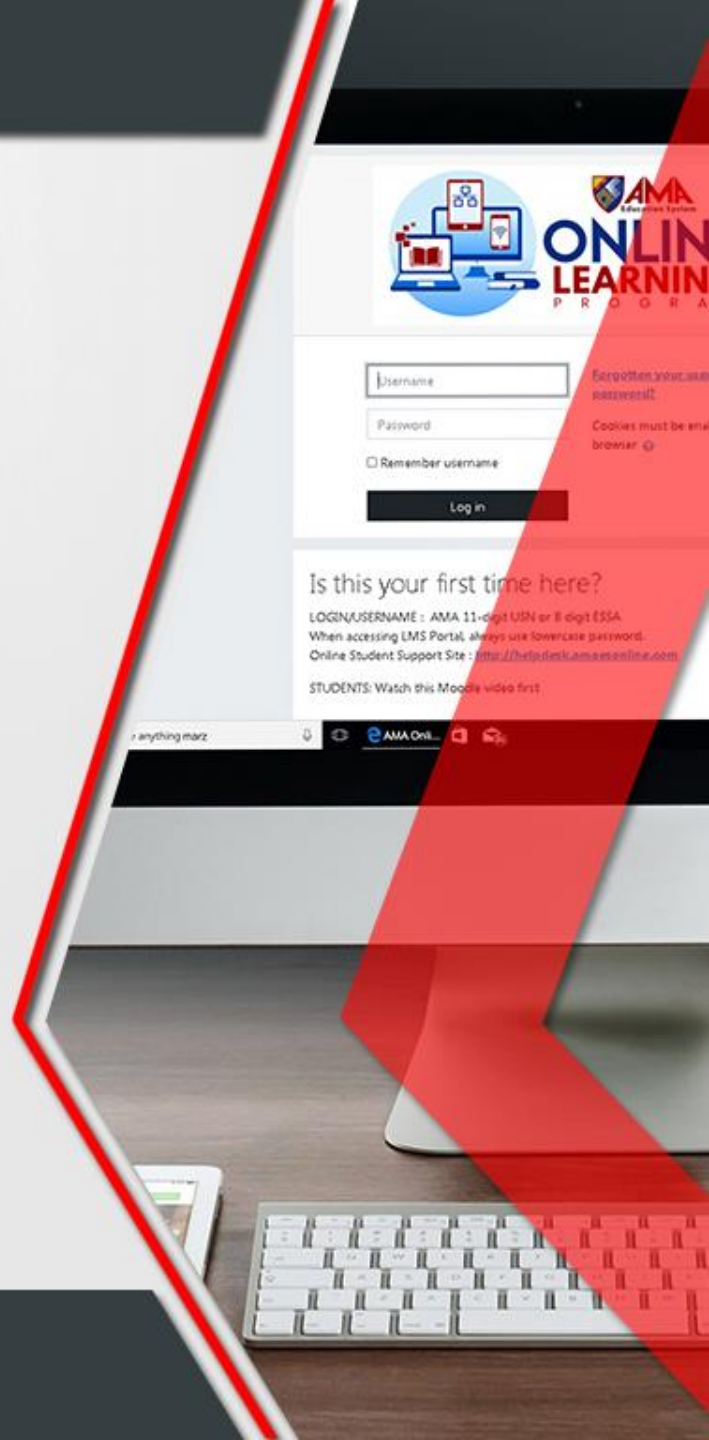
Security and Protection:

- It implements security measures to protect system resources, data, and user privacy.
- This includes user authentication, access control mechanisms, encryption, and malware detection/prevention.



Evolution of Operating Systems:

1. Serial Processing Systems
2. Batch Processing Systems
3. Time-Sharing Systems / Multitasking
4. Distributed Systems
5. Real-Time Systems

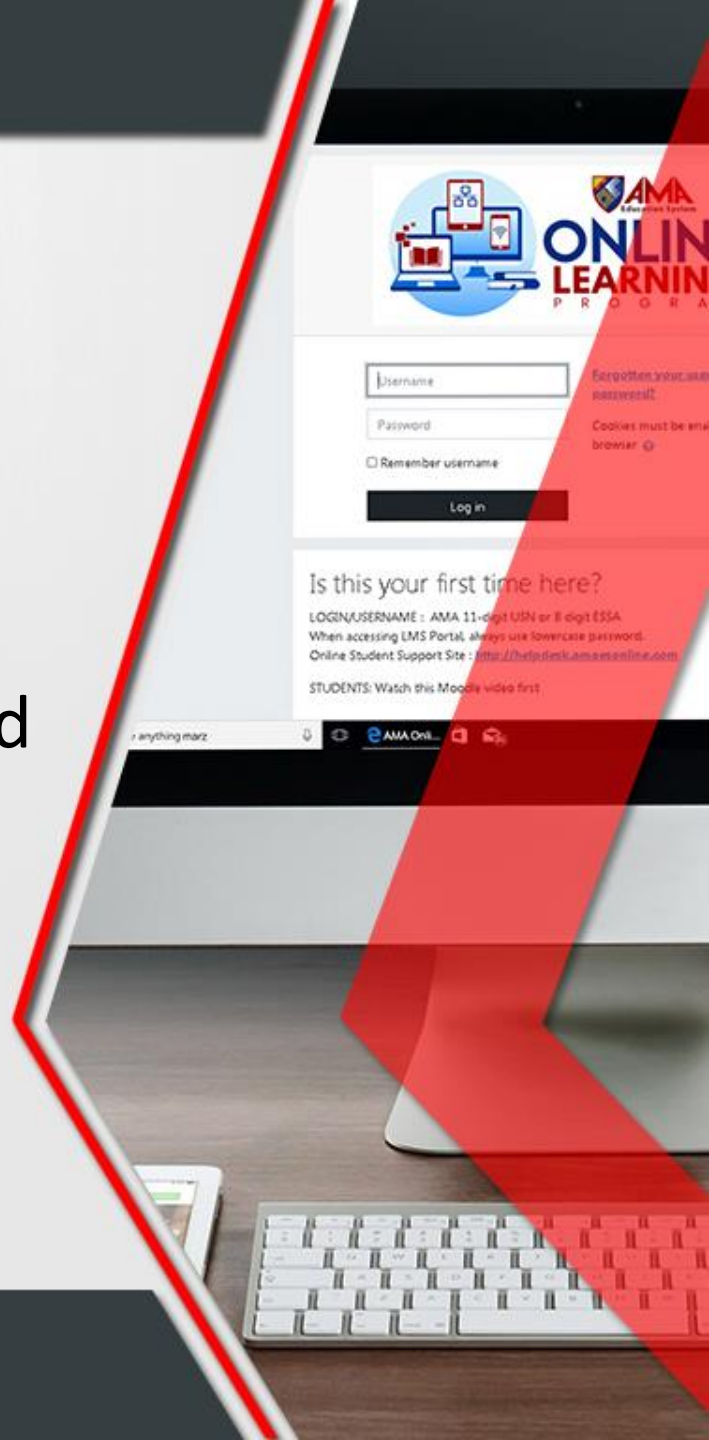


Serial Processing Systems

- Early computers operated in a serial processing mode, executing one program at a time without multitasking capabilities. Examples include the ENIAC and UNIVAC systems.

Batch Processing Systems:

- These systems used a simple job control language (JCL) to manage job execution.
- Batch processing systems emerged in the 1950s, allowing multiple jobs to be executed sequentially without user intervention. Programs were submitted in batches, and the OS managed job scheduling, resource allocation, and I/O operations. IBM's OS/360 is a notable example of a batch processing OS.
- Provided basic job scheduling and simple error handling.

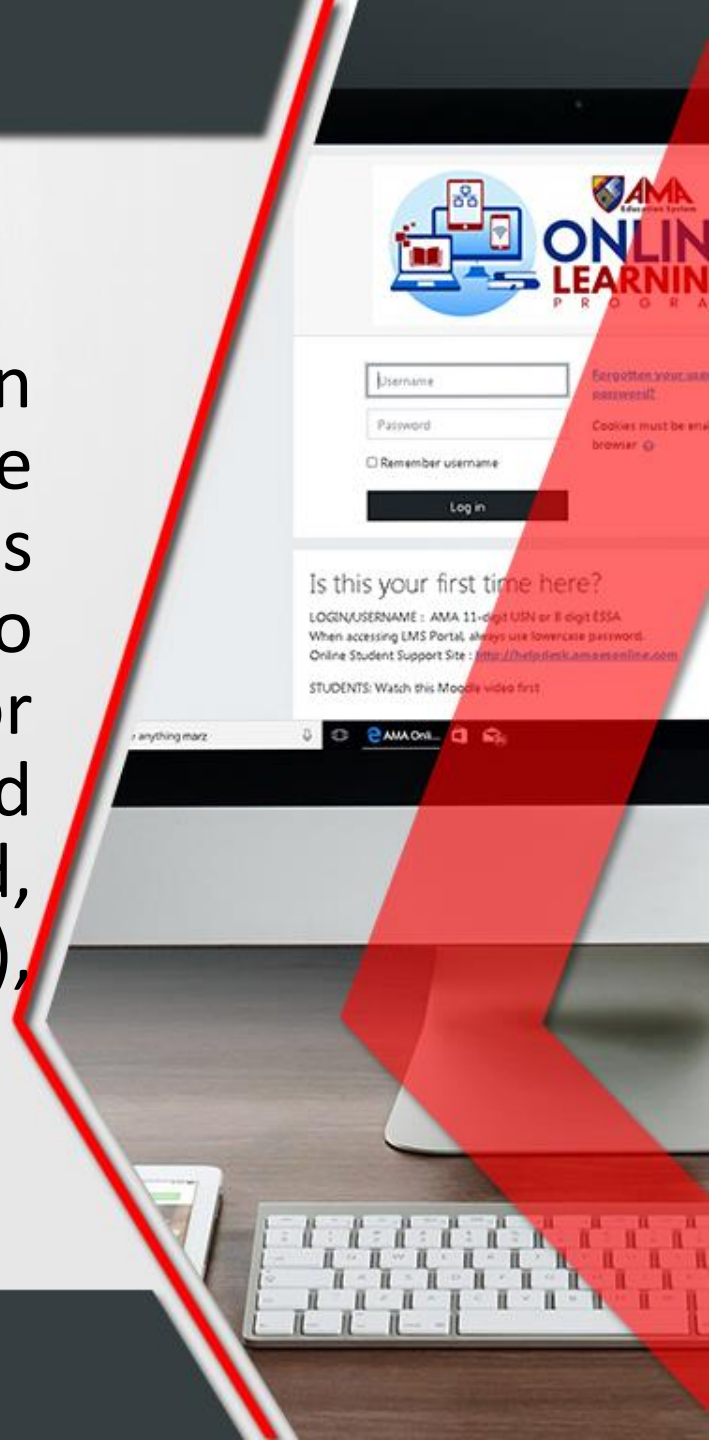


Time-Sharing Systems:

- Developed in the 1960s, enabled multiple users to interact with a computer simultaneously. These systems divided CPU time among multiple processes, providing each user with the illusion of having a dedicated computer. The introduction of interactive terminals and multi-user operating systems like CTSS (Compatible Time-Sharing System) and UNIX revolutionized computing by enabling real-time interaction and collaboration.

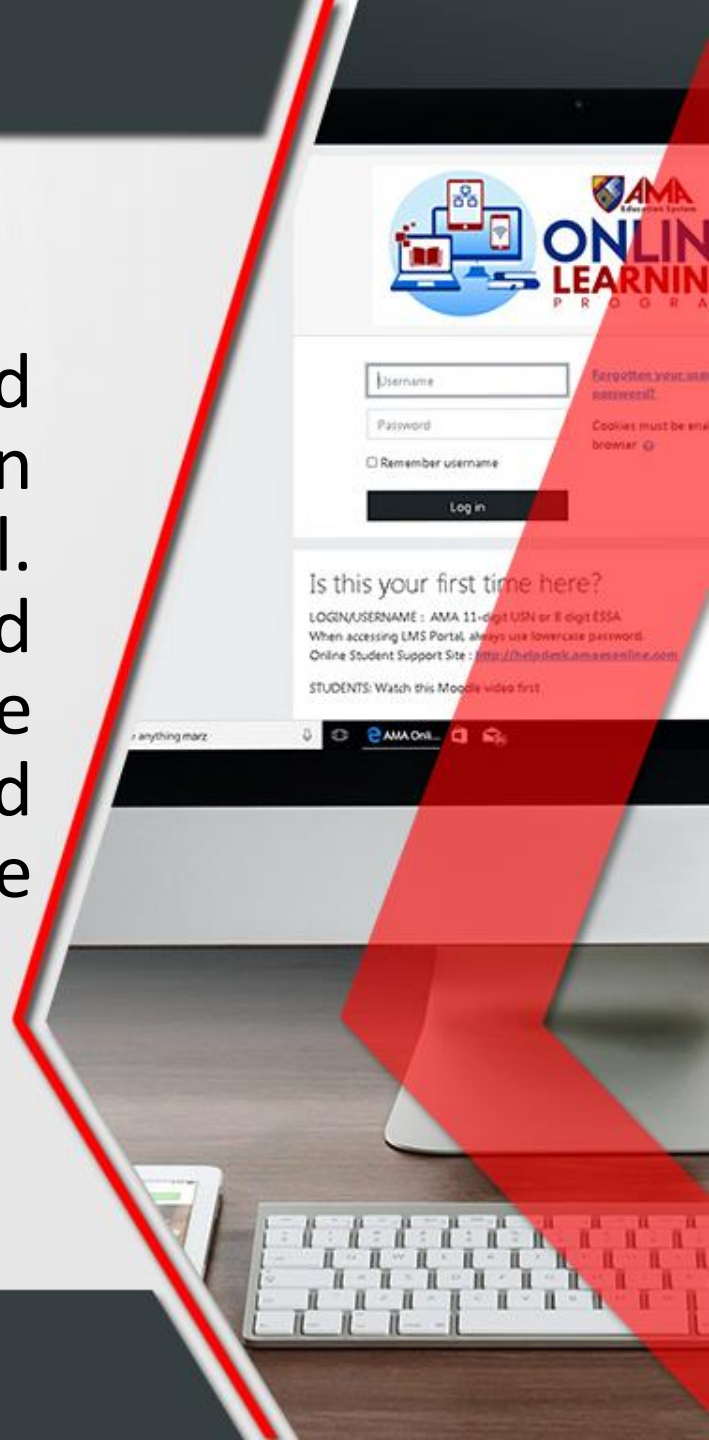
Distributed Systems:

- Emerged in the 1980s, enabling the coordination and management of resources across multiple interconnected computers. Distributed OSs facilitate distributed computing, allowing tasks to be divided among networked machines for improved scalability, fault tolerance, and performance. Examples include Google's Android, Microsoft's Windows Distributed File System (DFS), and various cluster computing systems.



Real-Time Systems:

- Real-Time Operating Systems (RTOS) are designed to meet strict timing constraints and deadlines in applications where timely response is critical. RTOSs prioritize tasks based on their urgency and guarantee timely execution, making them suitable for embedded systems, control systems, and mission-critical applications. Examples include VxWorks, QNX, and FreeRTOS.

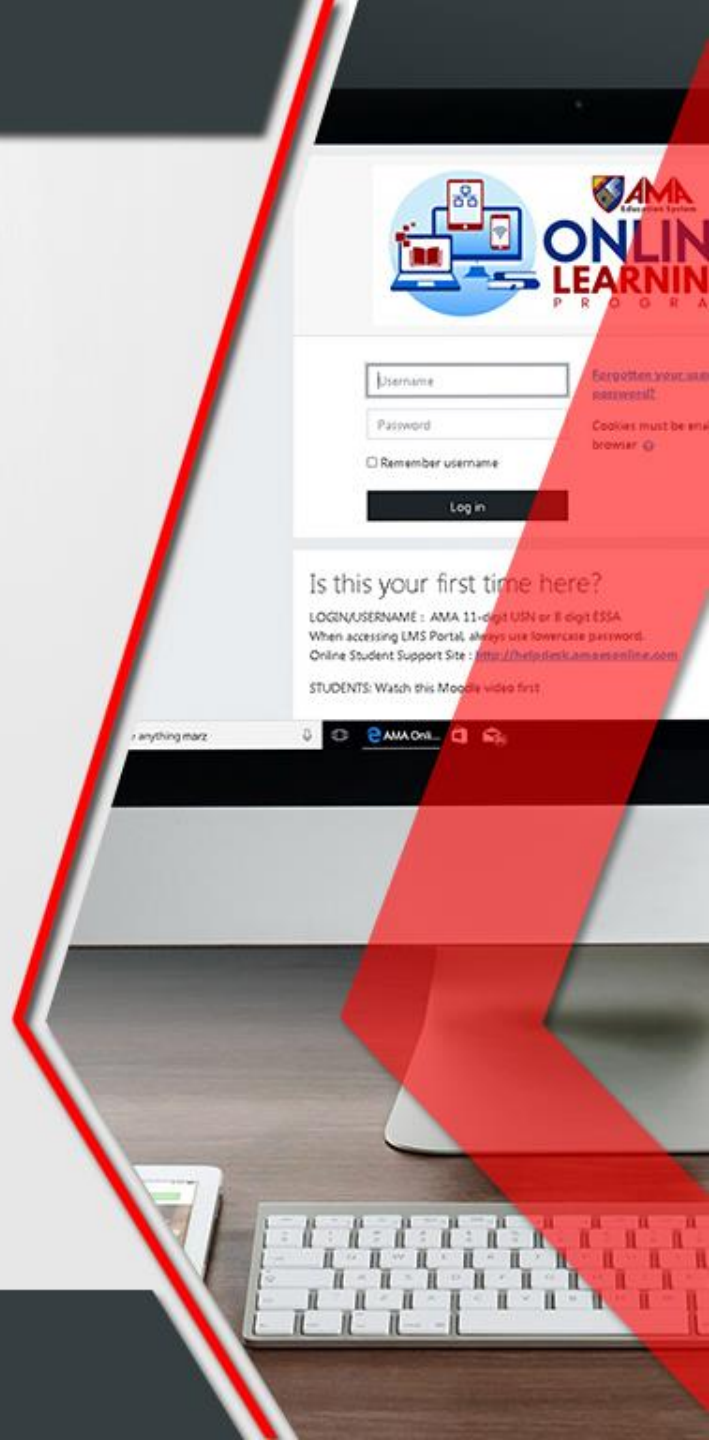


Generations of Operating Systems:

- Operating systems have evolved significantly over time, each generation introducing new features and capabilities.

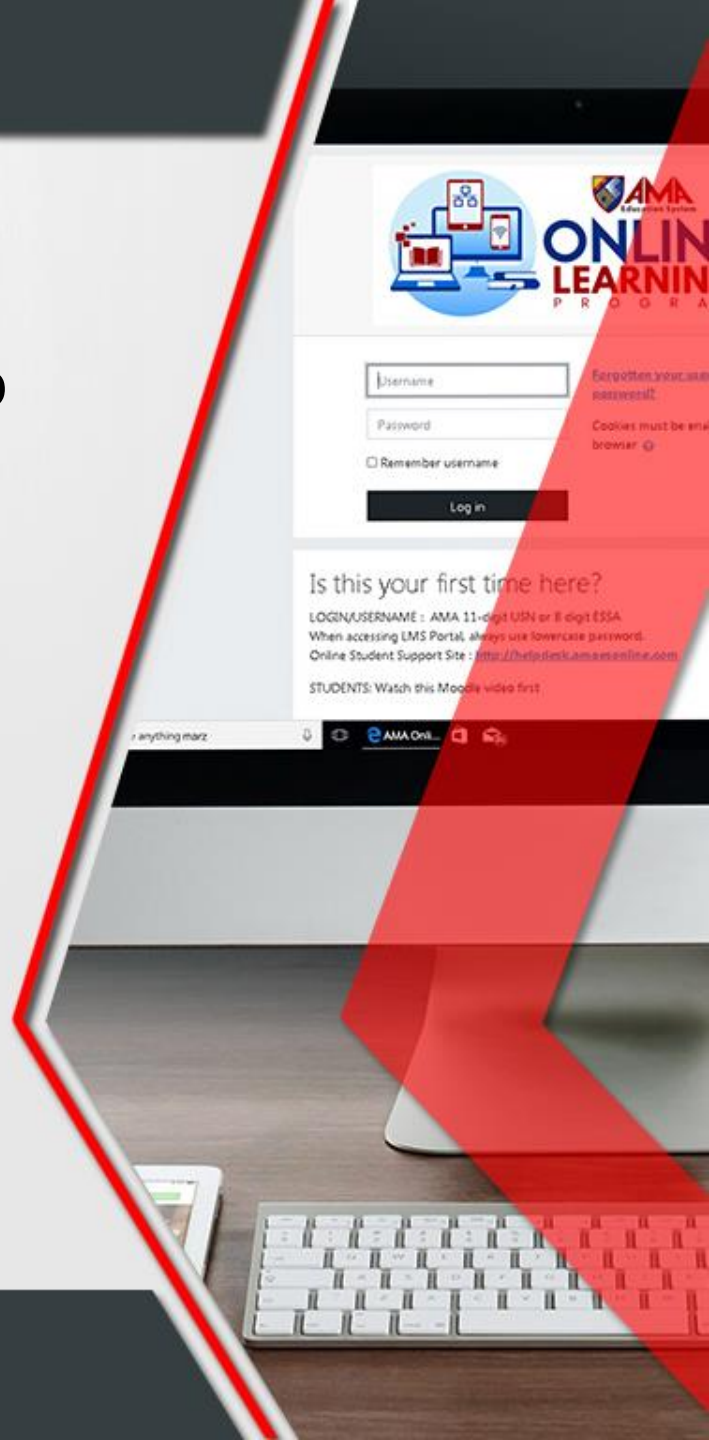
First Generation (1940s-1950s)

- **Machine Language:** Programs were written directly in machine language, requiring extensive knowledge of the hardware.
- **Manual Operation:** Computers were operated manually, with operators loading programs and data.
- **Batch Processing:** Jobs were grouped together into batches and processed sequentially.



Second Generation (1950s-1960s)

- **Assembly Language:** Introduced assembly language to make programming easier, but still required a deep understanding of the hardware.
- **Early Operating Systems:** Simple operating systems began to emerge, providing basic functions like job scheduling and memory management.
- **Time-Sharing:** The concept of time-sharing was introduced, allowing multiple users to share a single computer.



Third Generation (1960s-1970s)

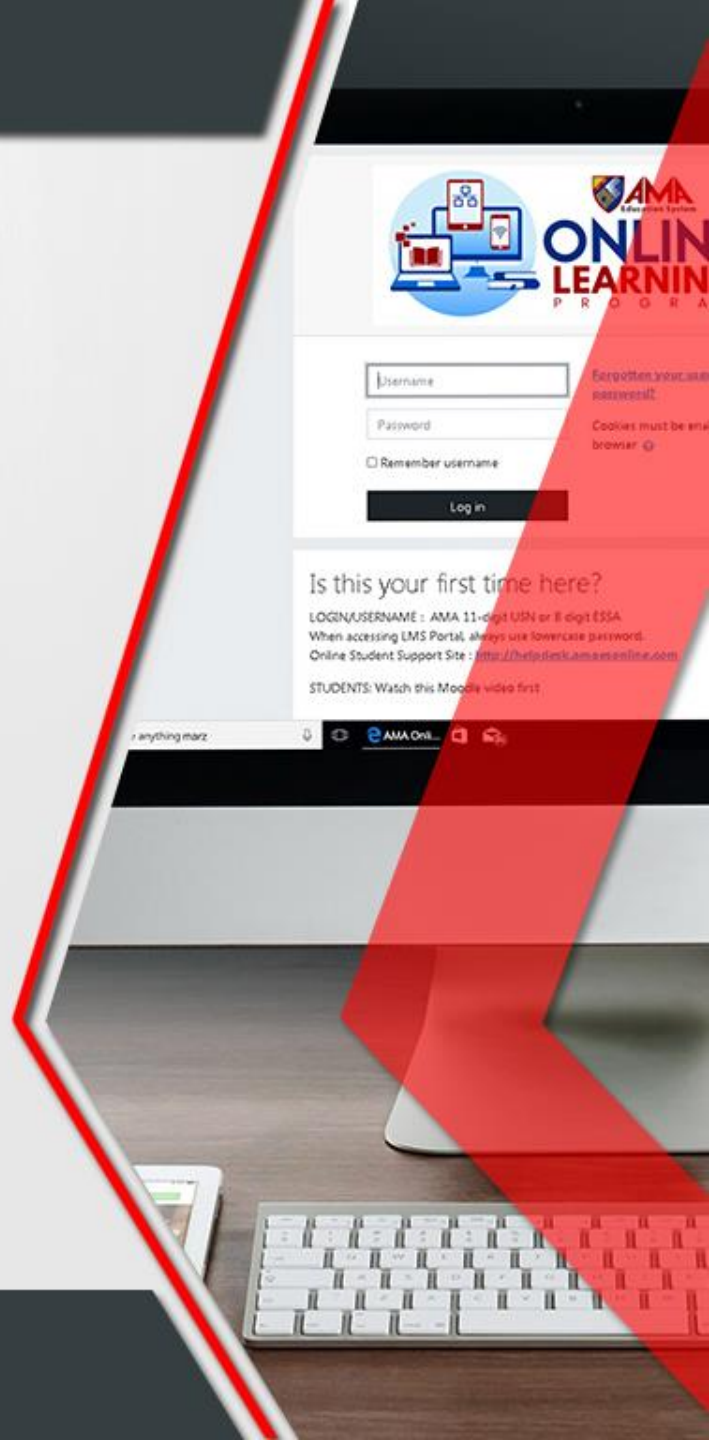
- **High-Level Languages:** High-level languages like FORTRAN, COBOL, and BASIC became popular, making programming more accessible.
- **Multiprogramming:** Multiple programs could be loaded into memory simultaneously, allowing for better resource utilization.
- **Multitasking:** Operating systems could handle multiple tasks concurrently, providing a more interactive experience.

Fourth Generation (1980s-1990s)

- **Graphical User Interfaces (GUIs):** GUIs made computers more user-friendly, replacing command-line interfaces.
- **Personal Computers:** The rise of personal computers led to the development of operating systems specifically designed for individual users.
- **Networking:** Operating systems began to support networking, allowing computers to communicate and share resources.

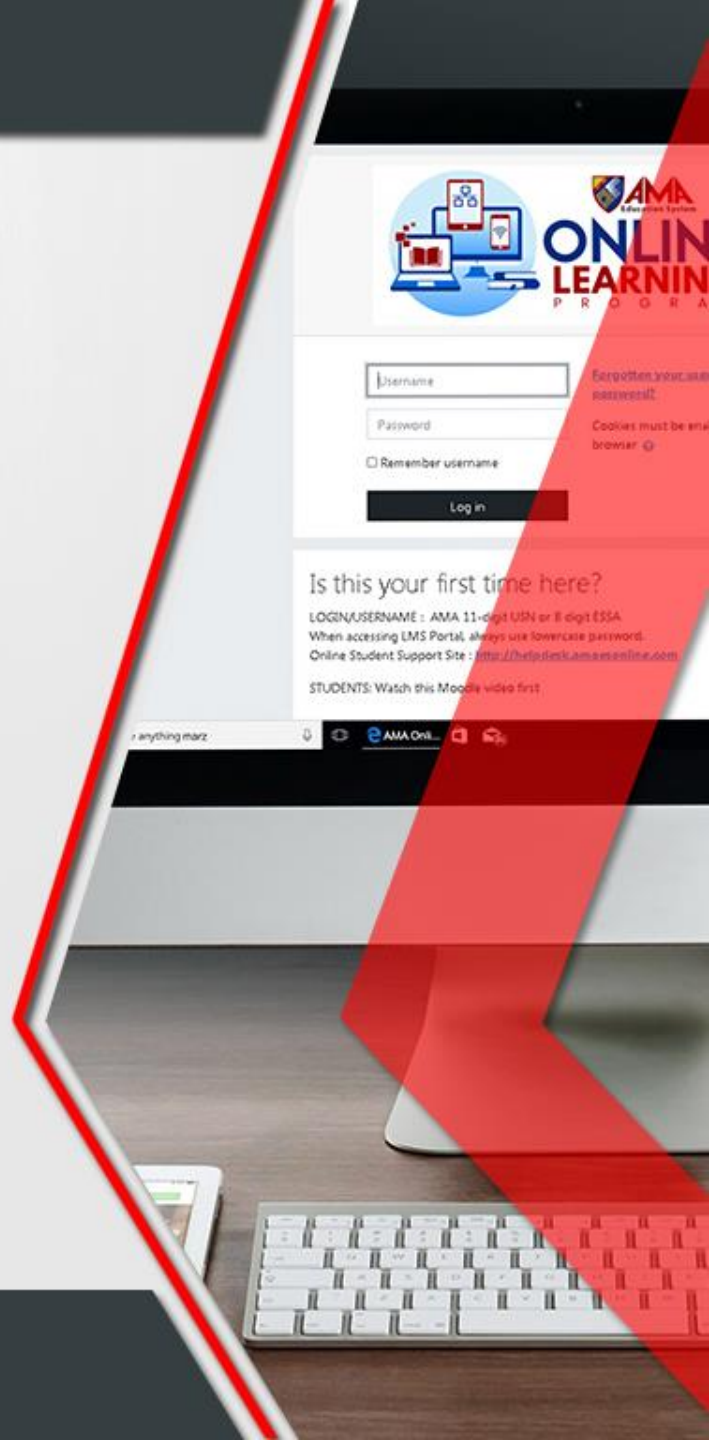
Fifth Generation (1990s-Present)

- **Distributed Systems:** Operating systems for distributed systems, such as networks and the cloud, became prevalent.
- **Mobile Operating Systems:** The development of mobile operating systems for smartphones and tablets.
- **Internet of Things (IoT) Operating Systems:** Specialized operating systems for IoT devices, often with limited resources and specific requirements.
- **AI and Machine Learning Integration:** Operating systems are increasingly incorporating AI and machine learning features for tasks like natural language processing and automation.



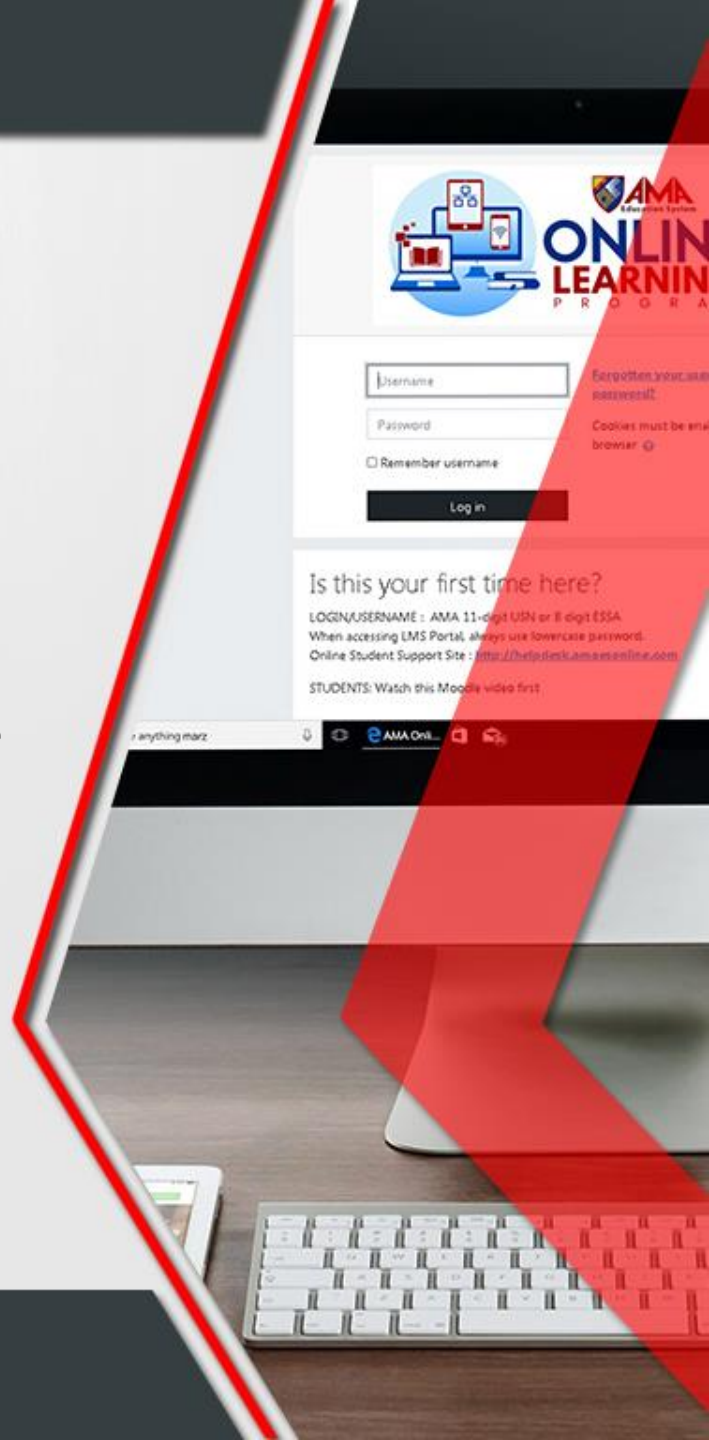
Classifications of Operating Systems:

- Batch Operating Systems
- Multi-user Operating Systems
- Single-user Operating Systems
- Multi-tasking Operating Systems
- Real-time Operating Systems (RTOS)
- Embedded Operating Systems
- Distributed Operating Systems
- Network Operating Systems
- Mobile Operating Systems
- Desktop Operating Systems
- Cloud Operating Systems



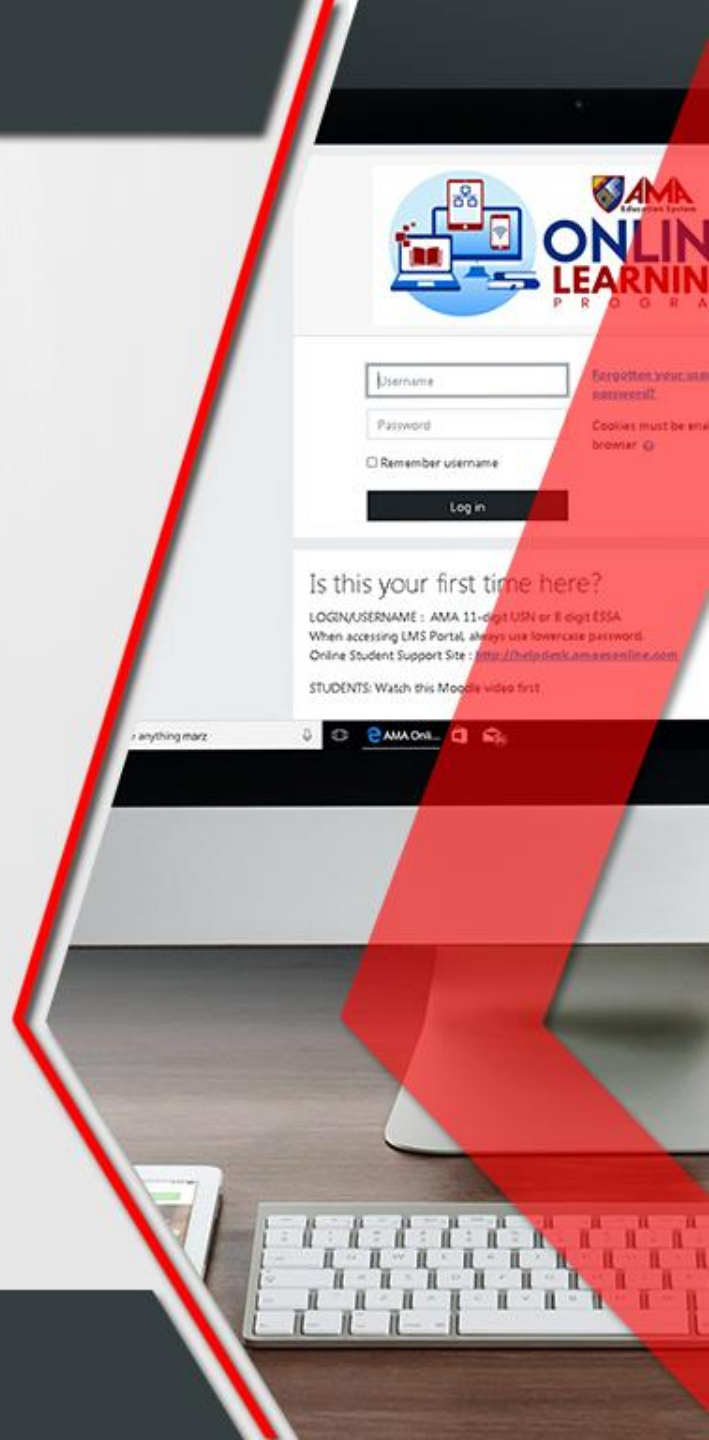
Batch Operating Systems

- **Description**: These systems execute batches of jobs without user interaction. Jobs are submitted to the system, processed sequentially, and output results are delivered back.
- **Characteristics**: No real-time user interaction; jobs are processed in batches.
- **Examples**: Early systems like IBM's OS/360.



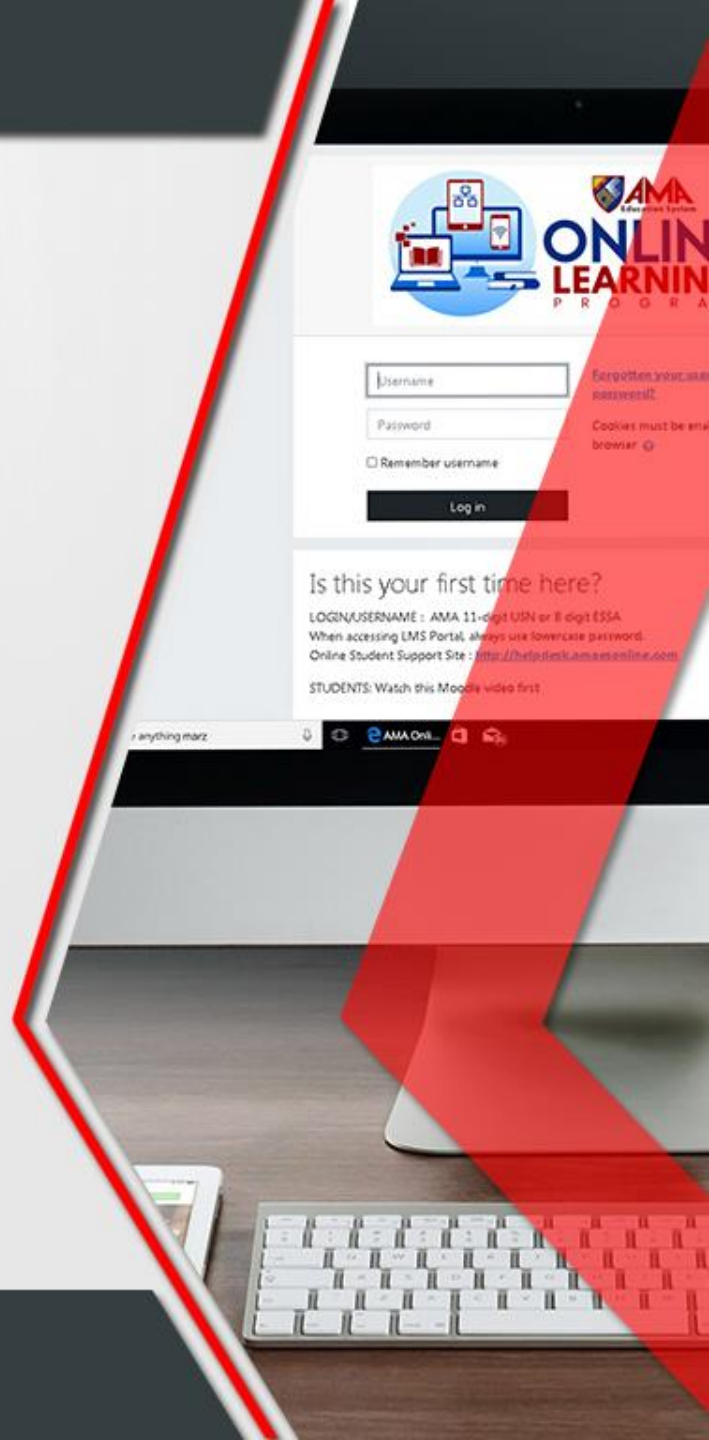
Multi-user Operating Systems

- **Description**: These OSs allow multiple users to access and use the system resources simultaneously. They manage user sessions, permissions, and resource allocation to ensure fair and secure usage.
- **Characteristics**: User accounts, permissions, and concurrent access management.
- **Examples**: UNIX, Linux, Windows Server.



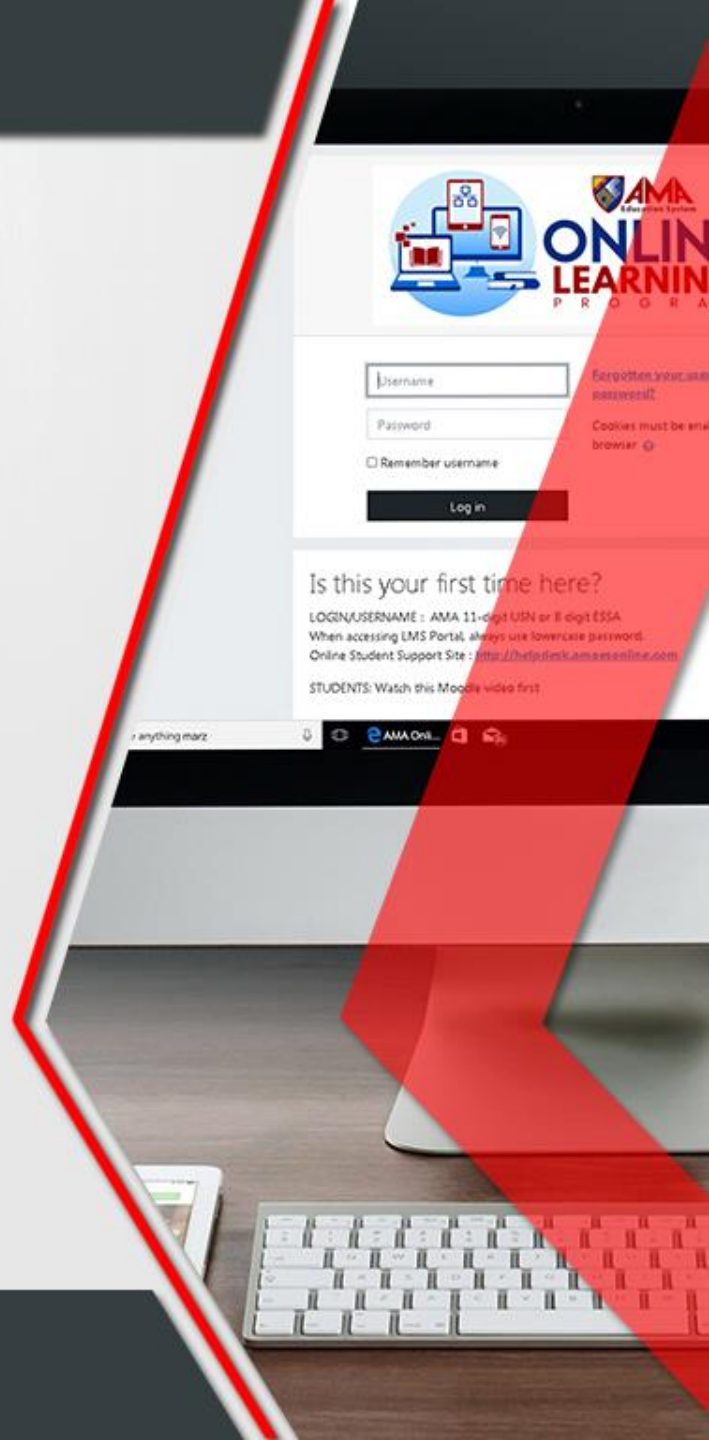
Single-user Operating Systems

- **Description**: Designed for use by one user at a time. While it may support multiple tasks, it does not need to handle simultaneous user sessions.
- **Characteristics**: Focuses on individual user experience and resources.
- **Examples**: Microsoft Windows 10/11 (in its standard consumer versions), macOS.



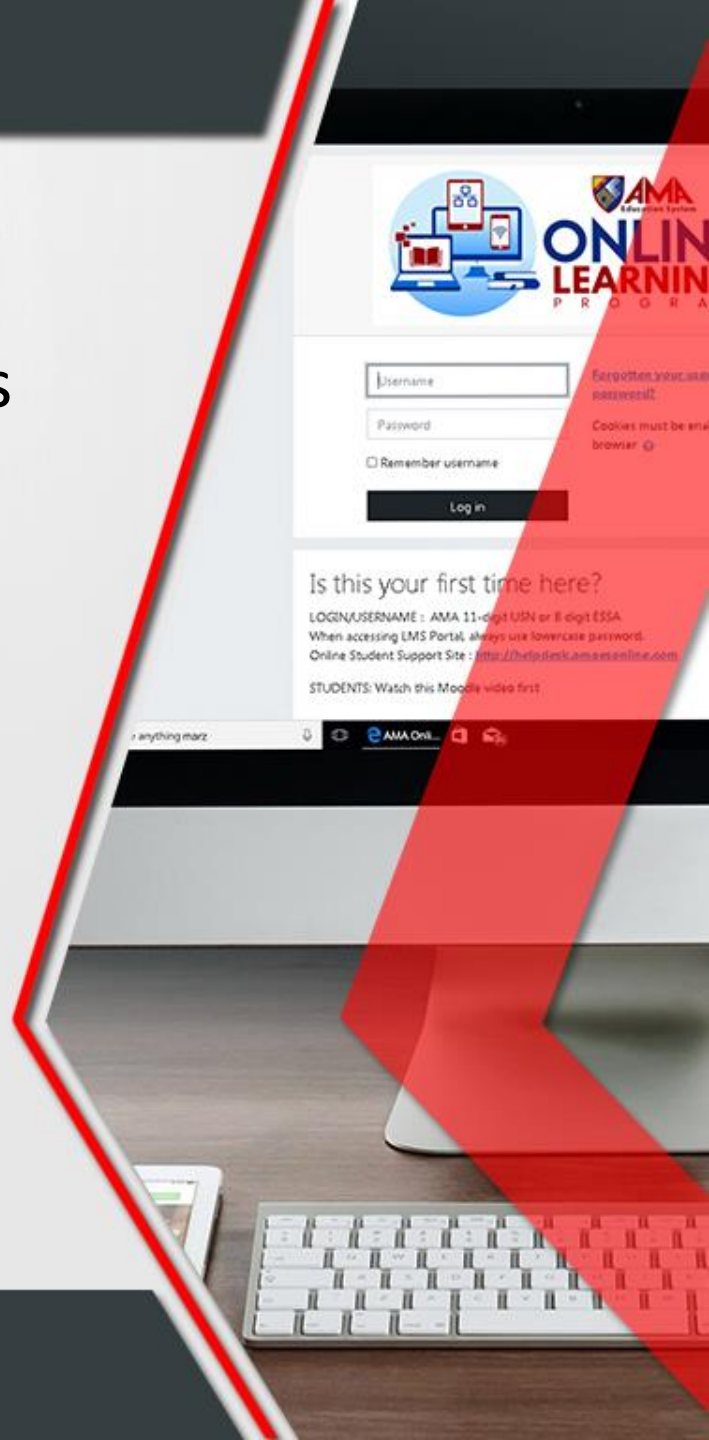
Multi-tasking Operating Systems

- **Description**: Capable of executing multiple tasks or processes concurrently, often by using techniques like time-sharing to allocate CPU time among tasks.
- **Characteristics**: Efficient CPU utilization, process management, and context switching.
- **Examples**: Linux, macOS, Windows 10/11.



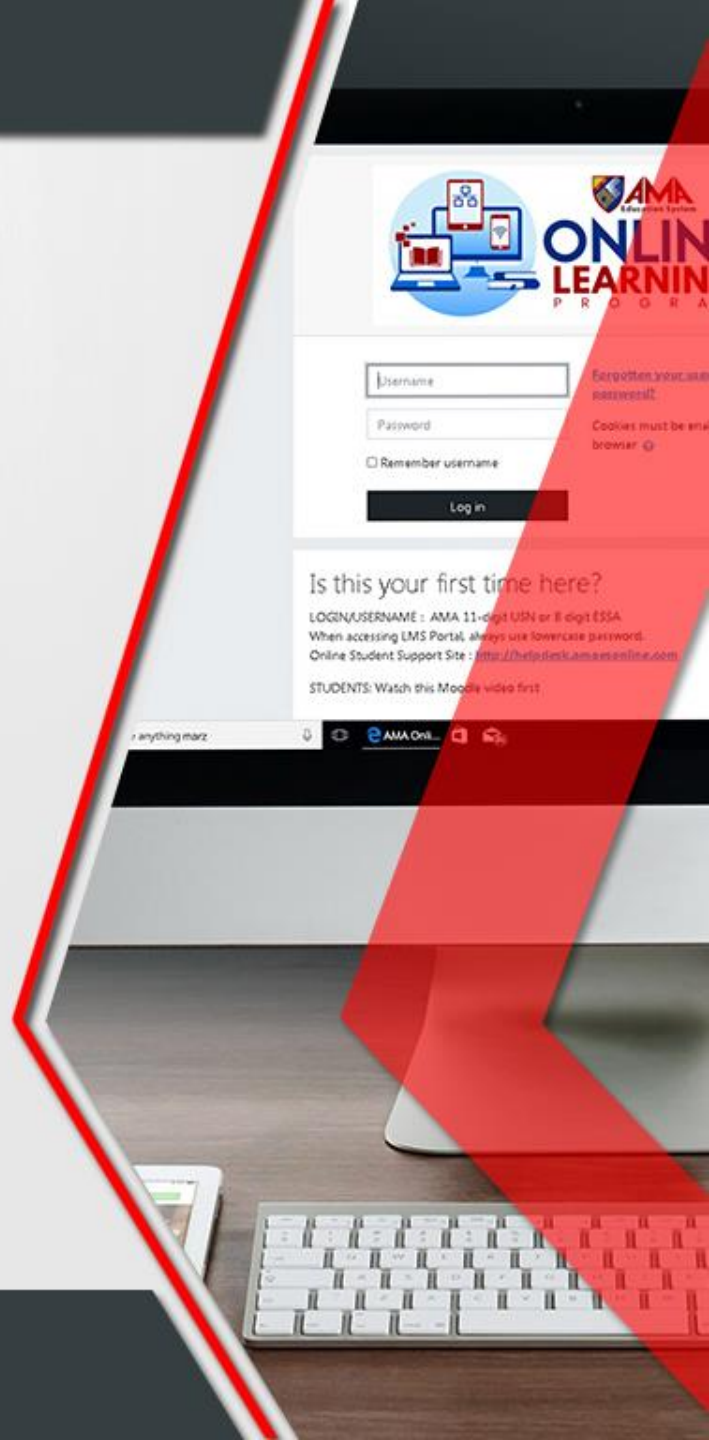
Real-time Operating Systems (RTOS)

- **Description**: Designed to handle real-time applications that require strict timing constraints. They provide immediate processing and response to events.
- **Characteristics**: Predictable response times, deterministic behavior, minimal latency.
- **Examples**: VxWorks, QNX, RTEMS.



Embedded Operating Systems

- **Description**: Tailored for specific devices or systems with dedicated functions. They are optimized for performance, efficiency, and low resource usage.
- **Characteristics**: Lightweight, specialized for specific hardware.
- **Examples**: Embedded Linux, FreeRTOS, Windows Embedded.

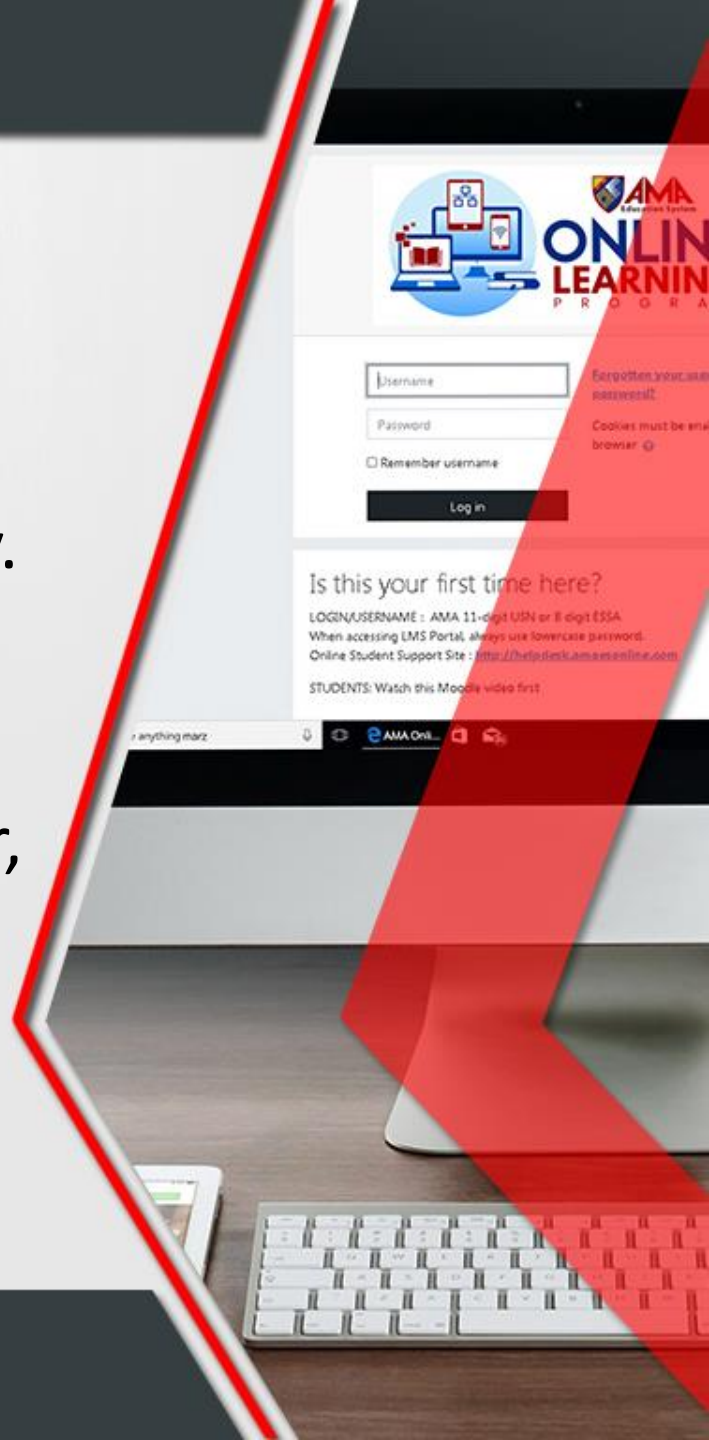


Distributed Operating Systems

- **Description**: Manage a network of computers as if they were a single system. They coordinate and share resources across multiple machines.
- **Characteristics**: Resource sharing, network transparency, and fault tolerance.
- **Examples**: Google's Android (in its distributed approach), Apache Hadoop, some implementations of Unix.

Network Operating Systems

- **Description**: Specifically designed to manage network resources and provide services over a network. They handle networking functions, file sharing, and security.
- **Characteristics**: Network management, user authentication, and data sharing.
- **Examples**: Novell NetWare, Microsoft Windows Server, Linux-based network solutions.



Mobile Operating Systems

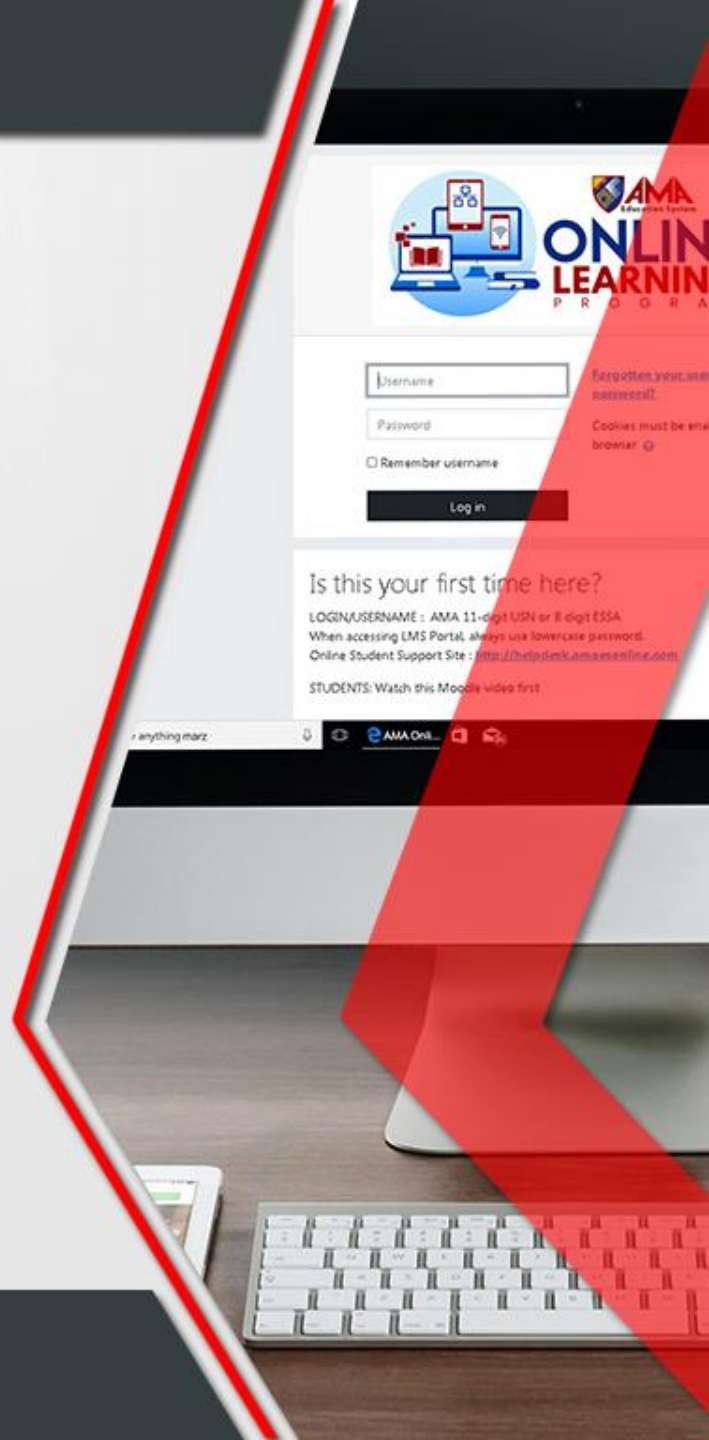
- **Description**: Designed for mobile devices, focusing on touch interfaces, battery efficiency, and mobile-specific applications.
- **Characteristics**: Touchscreen support, mobile applications, and power management.
- **Examples**: Android, iOS.

Desktop Operating Systems

- **Description**: Designed for personal computers and workstations, offering user-friendly interfaces and support for a wide range of applications.
- **Characteristics**: Graphical user interface (GUI), application support, and multitasking.
- **Examples**: Windows 10/11, macOS, Linux distributions like Ubuntu.

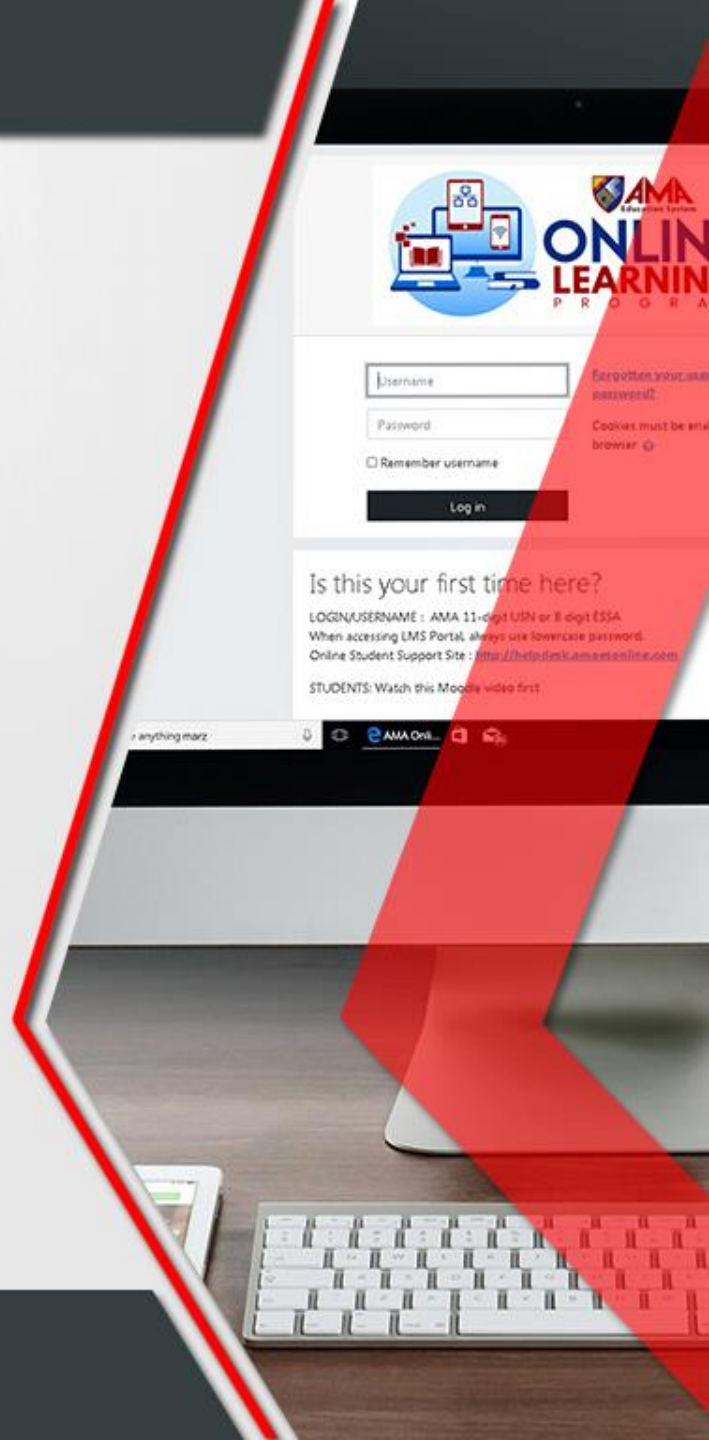
Cloud Operating Systems

- **Description**: Designed to manage cloud computing environments and provide scalable and flexible resources over the internet.
- **Characteristics**: Virtualization support, cloud service management, and resource scaling.
- **Examples**: Google Cloud Platform (GCP), Microsoft Azure, Amazon Web Services (AWS) OS offerings.



Components of Operating Systems:

1. Kernel
2. Process Management
3. Memory Management
4. File System
5. Device Management
6. User Interface
7. System Calls and APIs
8. Security and Protection
9. Networking
10. Utilities and System Services
11. Shell

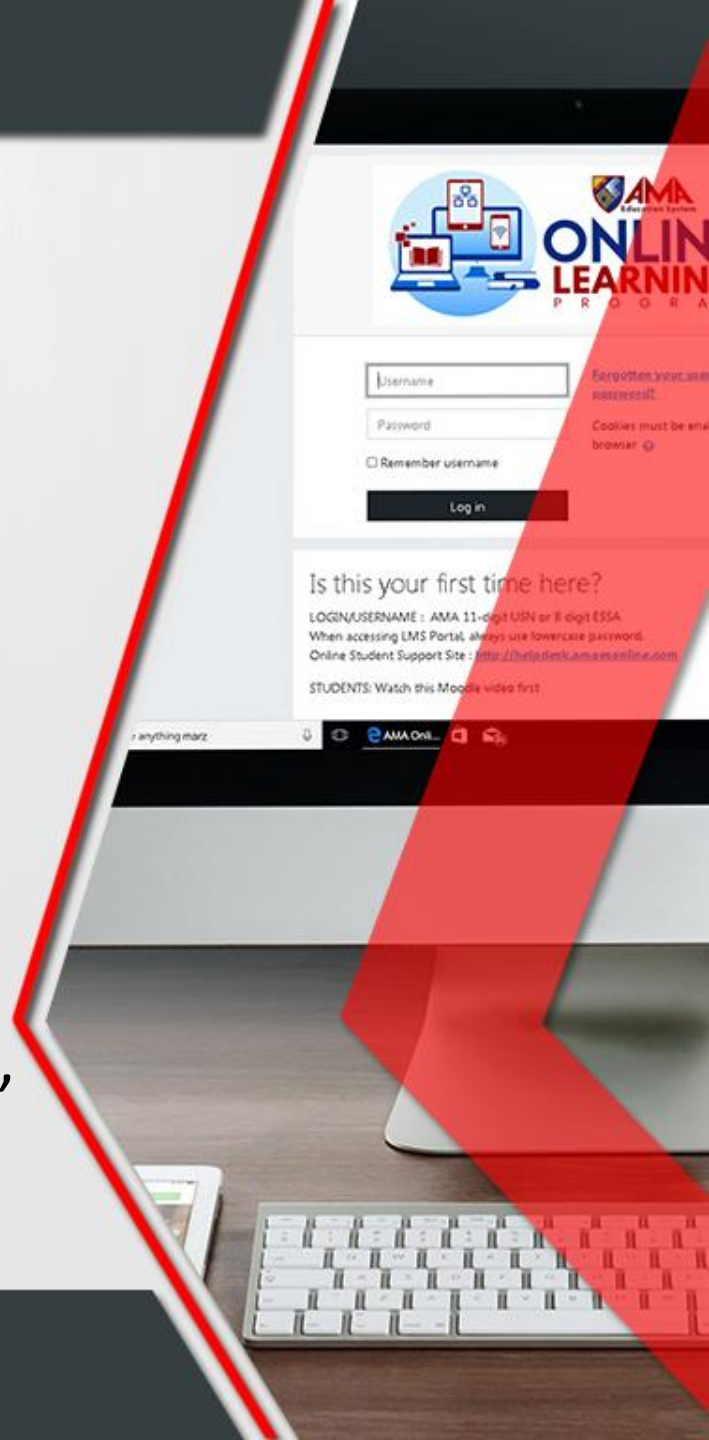


Kernel

- The kernel is the core component of an operating system, responsible for managing hardware resources and providing essential services to user programs.
- It handles tasks such as process management, memory management, device management, and system calls.
- The kernel operates in privileged mode, with direct access to hardware resources, and ensures proper isolation and protection between user processes.

Process Management

- Process management involves creating, scheduling, synchronizing, and terminating processes.
- The OS maintains a process table containing information about active processes, including their state, priority, and resource usage.
- It schedules processes for execution on the CPU, using algorithms such as round robin, priority-based scheduling, or multi-level feedback queues.
- Process synchronization mechanisms, such as semaphores, mutexes, and monitors, are used to coordinate access to shared resources and prevent race conditions.



Memory Management

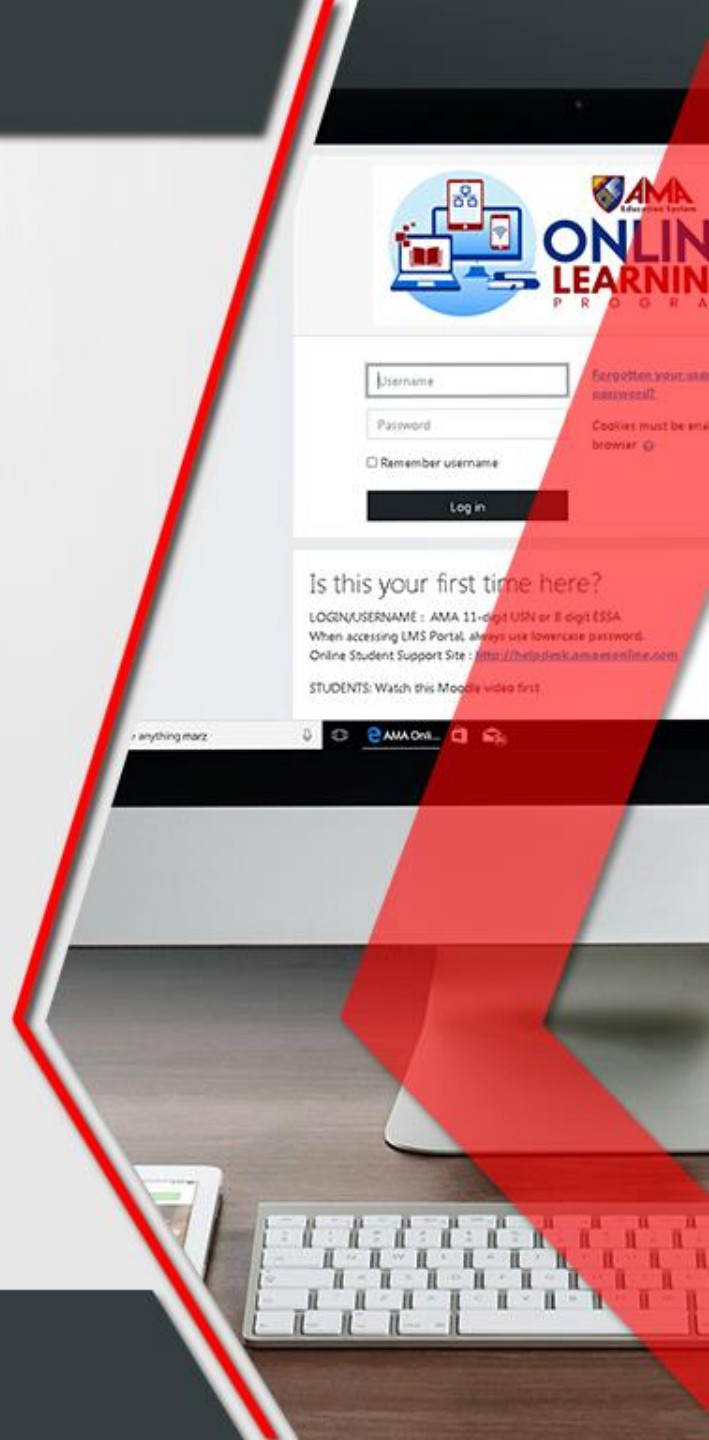
- Memory management involves managing system memory, including allocation, deallocation, and protection.
- The OS provides mechanisms for virtual memory, allowing processes to use more memory than physically available by swapping data between main memory and secondary storage.
- Memory protection mechanisms prevent processes from accessing memory locations outside their allocated address space, ensuring memory safety and security.
- Techniques such as paging, segmentation, and demand paging are used to optimize memory usage and minimize fragmentation.

File System

- The file system provides a hierarchical organization for storing and accessing files and directories on storage devices.
- It manages file metadata, including file attributes (e.g., name, size, permissions, timestamps) and file allocation information.
- The OS provides file system drivers to support different file system formats, such as FAT, NTFS, ext4, and HFS+.
- File system operations, such as file creation, deletion, reading, and writing, are performed through system calls and file APIs.

Device Management

- Device management involves controlling input/output devices such as keyboards, mice, displays, printers, and network interfaces.
- The OS provides device drivers to communicate with hardware devices and abstract device-specific details from user applications.
- Device drivers handle device initialization, configuration, and communication, translating high-level I/O requests into device-specific commands.
- Input/output operations are managed through device drivers and device controllers, ensuring efficient data transfer and error handling.

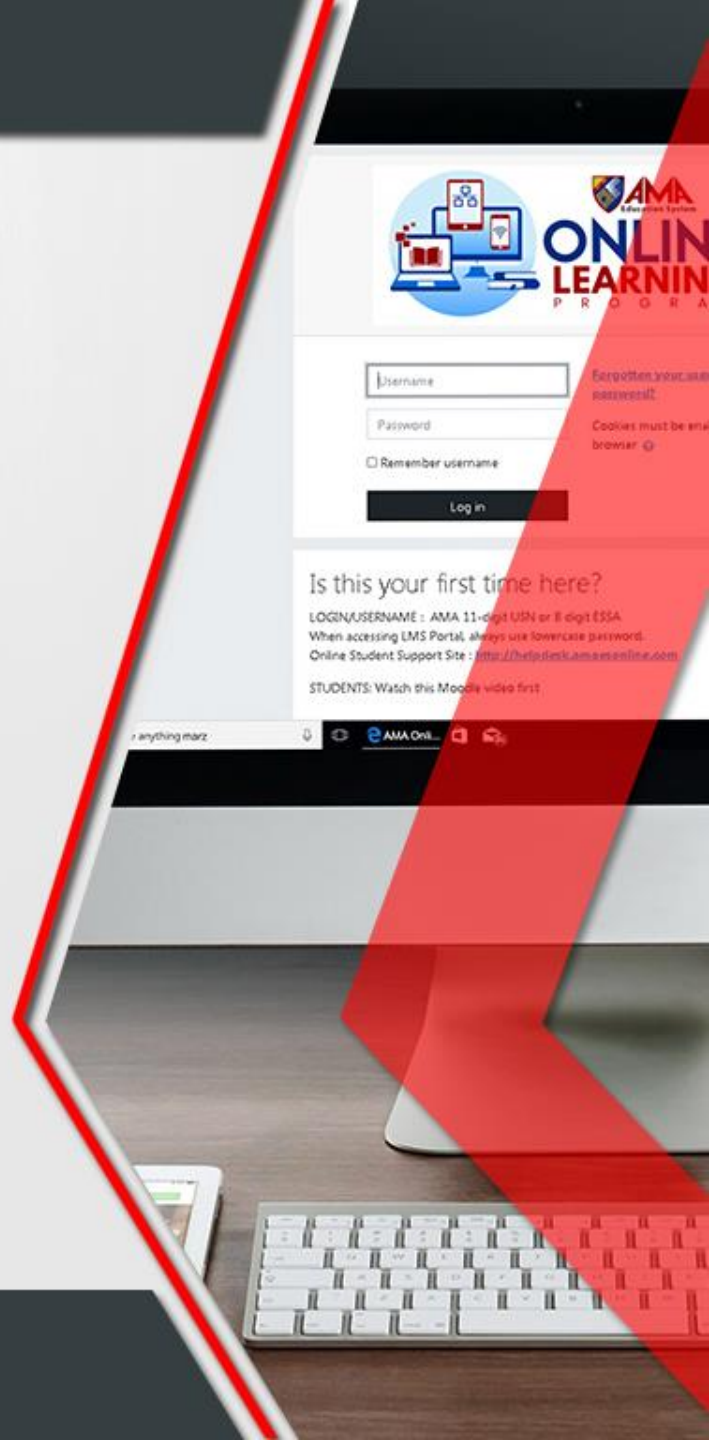


User Interface

- The user interface allows users to interact with the operating system and execute commands or applications.
- Graphical user interfaces (GUIs) provide visual elements such as windows, icons, menus, and buttons for user interaction.
- Command-line interfaces (CLIs) allow users to interact with the system through text-based commands and shell programs.
- The OS provides user interface components, such as window managers, desktop environments, and command interpreters, to facilitate user interaction.

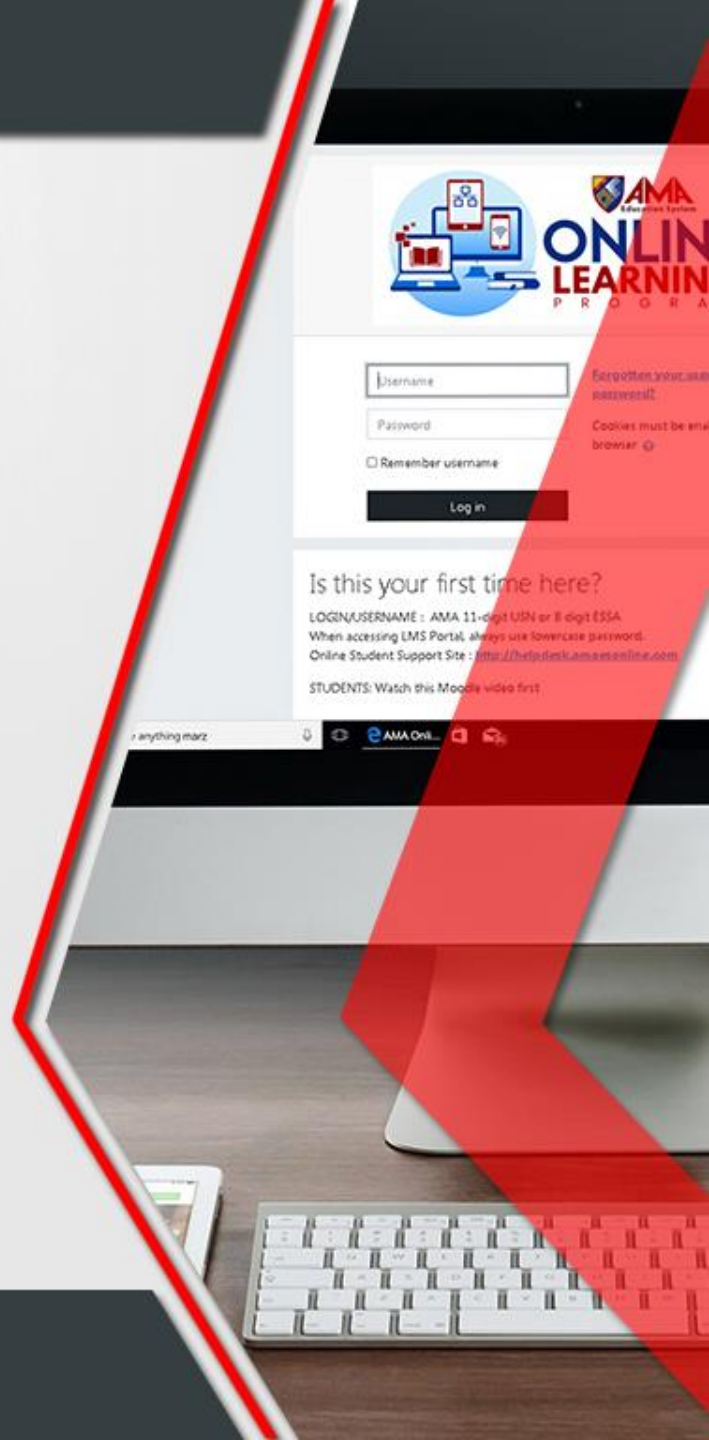
System Calls and APIs

- Provide an interface for applications to request services from the OS.
- High-level interfaces that abstract system calls into more user-friendly functions.



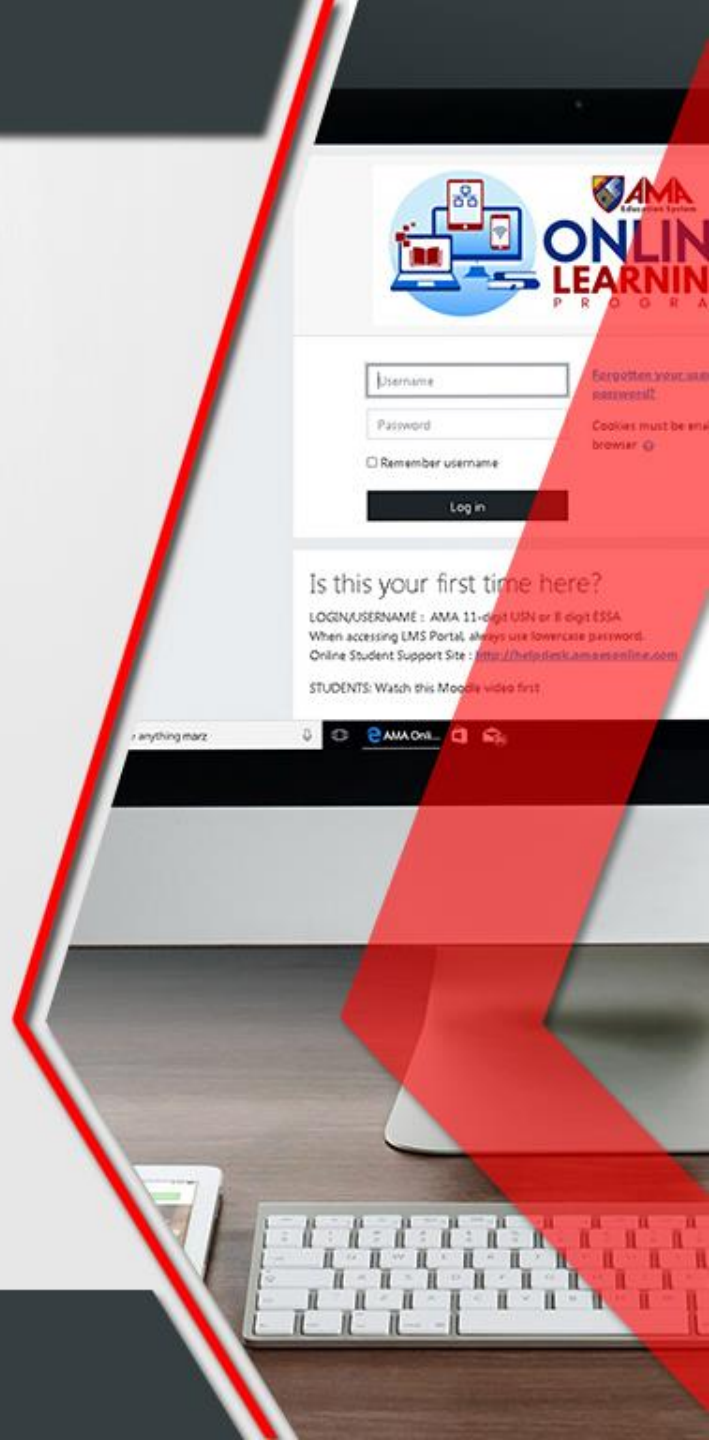
Security and Protection

- Ensures the integrity and security of system resources and data.
  - ➢ Verifies user identities (e.g., passwords, biometrics).
  - ➢ Manages user permissions and access control.
  - ➢ Protects data through encoding techniques.



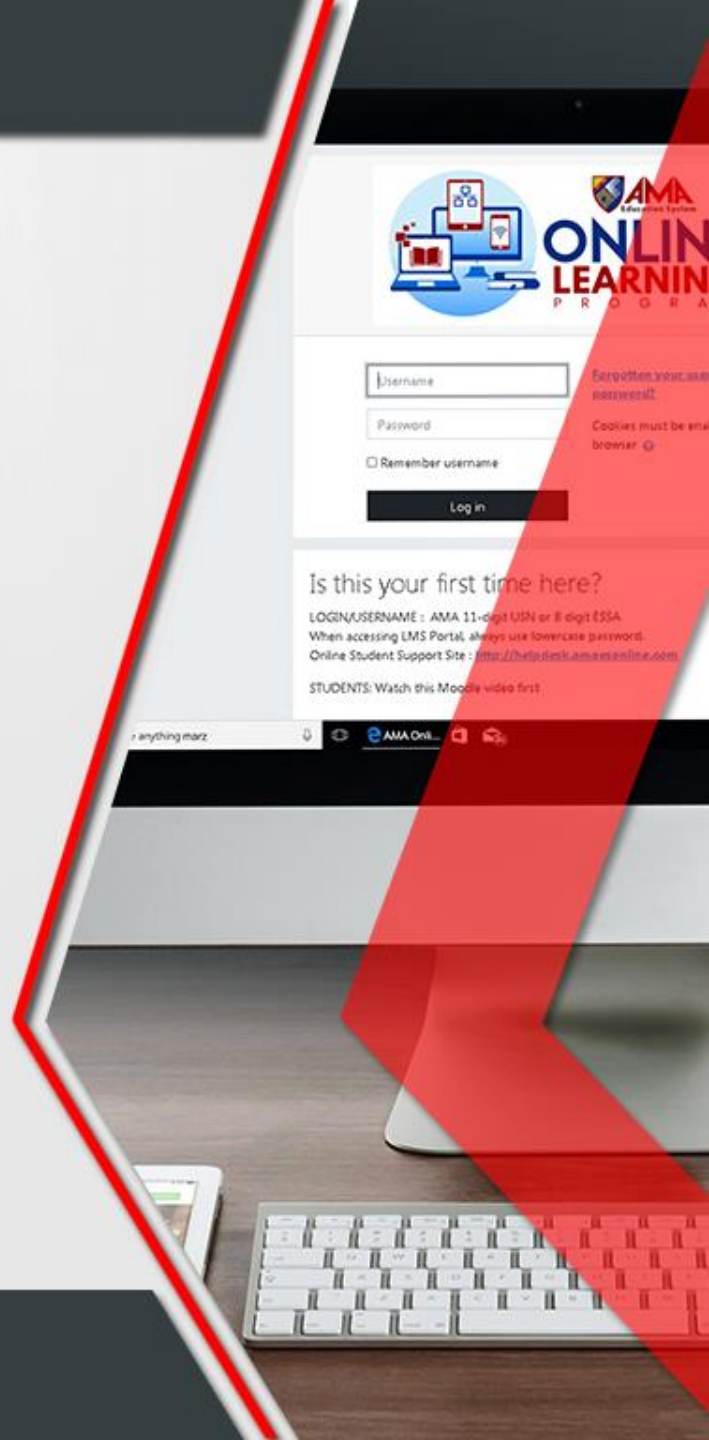
Networking

- Manages network connections and communication between devices.
- Implements network protocols such as TCP/IP.
- Handles communication with network hardware.



Utilities and System Services

- Provide additional functionalities and support for system management and maintenance.
- System Utilities: Tools for managing files, processes, and system performance (e.g., disk cleanup, task manager).



Shell

- Acts as an intermediary between the user and the operating system, particularly in CLI-based systems.
- Allows users to write scripts for automating tasks.