## Name & Title: - Virtual Machine Orchestration Management System

**Introduction** :- A Virtual Machine Orchestration Management System is a software tool that manages the deployment, scaling, and monitoring of virtual machines in a cloud computing environment. It automates the provisioning and configuration of virtual machines and ensures that resources are used efficiently. The system allows administrators to manage virtual machines across multiple hosts and clusters, enabling them to orchestrate complex distributed applications with ease. It also provides features for load balancing, auto-scaling, and fault tolerance, ensuring that applications are highly available and resilient.

### Enums Used:

| Enum Name | Description |
|---|---|
| VM_STATE_RUNNING | This enum value represents the state of a virtual machine when it is running and actively executing tasks. |
| VM_STATE_STOPPED | This enum value represents the state of a virtual machine when it is completely powered off and not executing any tasks. |
| VM_STATE_PAUSED | This enum value represents the state of a virtual machine when it has been paused and is temporarily suspended from executing tasks. The virtual machine can later be resumed from this state. |

### Structures Used:

VM(Virtual Machine):

| Property | Data Type | Description |
|---|---|---|
| id | int | An integer identifier for the virtual machine. |
| name | char[50] | A string representing the name of the VM. |
| state | VM_State | An enumeration representing the state of the VM. |

VMO System:

| Property | Data Type | Description |
|---|---|---|
| vms | Pointer to VM | A pointer to an array of VMs in the VMO System |
| num_vms | Integer | The number of VMs in the VMO System |

### Problem Description:

The following functions will be used for the assignment:

| Function Name | Parameter | Return Type | Description |
|---|---|---|---|
| init_vmo_system | int num_vms | VMO_System | This function initializes a virtual machine operating system (VMO) with the specified number of virtual machines (VMs). The function returns a VMO_System struct, which contains an array of VMs and the number of VMs. |
| add_vm | VMO_System *vmo, char *name | int | This function adds a new virtual machine to the VMO system with the given name. The function returns the ID of the new virtual machine or -1 if the operation fails. |
| remove_vm | VMO_System *vmo, int id | int | This function removes the virtual machine with the specified ID from the VMO system. The function returns 0 on success, -1 if the VMO system is invalid, or -2 if the virtual machine is not found. |
| start_vm | VMO_System *vmo, int id | int | This function starts the virtual machine with the specified ID in the VMO system. The function returns 0 on success, -1 if the VMO system or VM array is not initialized, -2 if the VM is not found, or -3 if the VM is already running. |
| stop_vm | VMO_System *vmo, int id | int | Stops a virtual machine with the specified ID in the VMO system. Returns 0 on success, and a negative integer error code on failure. Error codes: -1 for uninitialized VMO system, -2 for no virtual machines in the VMO system, -3 for virtual machine not running, and -4 for virtual machine not found. |
| get_vm_state | VMO_System *vmo, int id | VM_State | Returns the state (running, paused, or stopped) of the virtual machine with the specified ID in the VMO system. Returns VM_STATE_STOPPED if the VMO system or virtual machine is null, or if the ID is invalid. |

**Test Cases** :

| Test Case Name | Input | Output | Explanation |
|---|---|---|---|
| testInitVmoSystem_ValidNumVms | Number of VMs: 3 | Initialized VMO_System object | Test case checks whether the VMO_System object is initialized correctly with 3 VMs, and their initial |

| | | | |
|---|---|---|---|
| | | | states are set to STOPPED. |
| testInitVMOSystem_ZeroNumVMs | Number of VMs: 0 | Initialized VMO_System object | Test case checks whether the VMO_System object is initialized correctly with 0 VMs. |
| testInitVMOSystem_PositiveNumVMs | Number of VMs: 5 | Initialized VMO_System object | Test case checks whether the VMO_System object is initialized correctly with 5 VMs, and their names are not the default names "VM0", "VM1", "VM2". |
| test_add_vm_with_null_name | NULL | -1 | Test case checks whether add_vm function returns -1 when a null name is passed as an argument. |
| testAddVm_Success | VM name: "vm1" | 1 | Test case checks whether a VM with the name "vm1" is successfully added to the VMO_System object. |
| testAddVm_MultipleSuccess | VM names: "vm1", "vm2", "vm3" | 1, 2, 3 | Test case checks whether multiple VMs with names "vm1", "vm2", "vm3" are successfully added to the VMO_System object with their respective IDs. |
| testRemoveVM_InvalidVMOSystem | VMO_System with NULL vms pointer | -1 | Test case checks whether remove_vm function returns -1 when the VMO_System object has a NULL vms pointer. |
| testRemoveVM_FirstVM | VMO_System with 2 VMs and ID to remove: 1 | 0 | Test case checks whether the VM with ID 1 is successfully removed from the VMO_System object, and the remaining VM has ID 2. |
| testRemoveVM_LastVM | VMO_System with 2 VMs and ID to remove: 2 | 0 | Test case checks whether the VM with ID 2 is successfully removed from the VMO_System object, and the remaining VM has ID 1 |

| | | | |
|---|---|---|---|
| testStartVM_NullVMOSystem | VMO_System vmo = {NULL, 0}; | -1 | Tests that start_vm function returns -1 when the VMO_System pointer is NULL. |
| testStartVM_AlreadyRunning | VMO_System vmo = {(VM *)malloc(2 * sizeof(VM)), 2}; vmo.vms[0].id = 1; vmo.vms[0].state = VM_STATE_RUNNING; | -3 | Tests that start_vm function returns -3 when the VM to start is already running. |
| testStartVM_PauseOtherVMs | VMO_System vmo = {(VM *)malloc(3 * sizeof(VM)), 3}; vmo.vms[0].id = 1; vmo.vms[0].state = VM_STATE_RUNNING; vmo.vms[1].id = 2; vmo.vms[1].state = VM_STATE_RUNNING; vmo.vms[2].id = 3; vmo.vms[2].state = VM_STATE_STOPPED; | 0 | Tests that start_vm function pauses all other running VMs and starts the requested VM when there are multiple VMs running. |
| testStopVm_NullVmoSystem | VMO_System vmo = {NULL, 0}; | -2 | Tests that stop_vm function returns -2 when the VMO_System pointer is NULL. |
| testStopVm_NoVms | VMO_System vmo = {NULL, 0}; | -2 | Tests that stop_vm function returns -2 when there are no VMs in the VMO_System. |
| testStopVm_VmNotFound | VM vms[3] = {{1, "vm1", VM_STATE_RUNNING}, {2, "vm2", VM_STATE_STOPPED}, {3, "vm3", VM_STATE_PAUSED}}; VMO_System vmo = {vms, 3}; | -4 | Tests that stop_vm function returns -4 when the requested VM to stop is not found in the VMO_System. |

| | | | |
|---|---|---|---|
| testStoppedVM | VMO_System vmo = init_vmo_system( 1); VM vm = {0, "test", VM_STATE_STO PPED}; vmo.vms[0] = vm; | VM_STATE_ST OPPED | Tests that get_vm_state function returns VM_STATE_STOPPED for a VM that is stopped. |
| testRunningVM | VMO_System vmo = init_vmo_system( 1); VM vm = {0, "test", VM_STATE_RUN NING}; vmo.vms[0] = vm; | VM_STATE_RU NNING | Tests that get_vm_state function returns VM_STATE_RUNNING for a VM that is running. |
| testPausedVM | VMO_System vmo = init_vmo_system( 1); VM vm = {0, "test", VM_STATE_PAU SED}; vmo.vms[0] = vm; | VM_STATE_PA USED | Tests that get_vm_state function returns VM_STATE_PAUSED for a VM that is paused |