# University Of Westminster

**Coursework Report – 5COSC019C Object Oriented Programming**

**Test Plan**

Module Leader: Mr. Guhanathan Poravi

Student Name:  Sinali Tharindi Senarathne

UOW Number: w2087753

IIT Number: 20230514

## Phase 1 – CLI Implementation

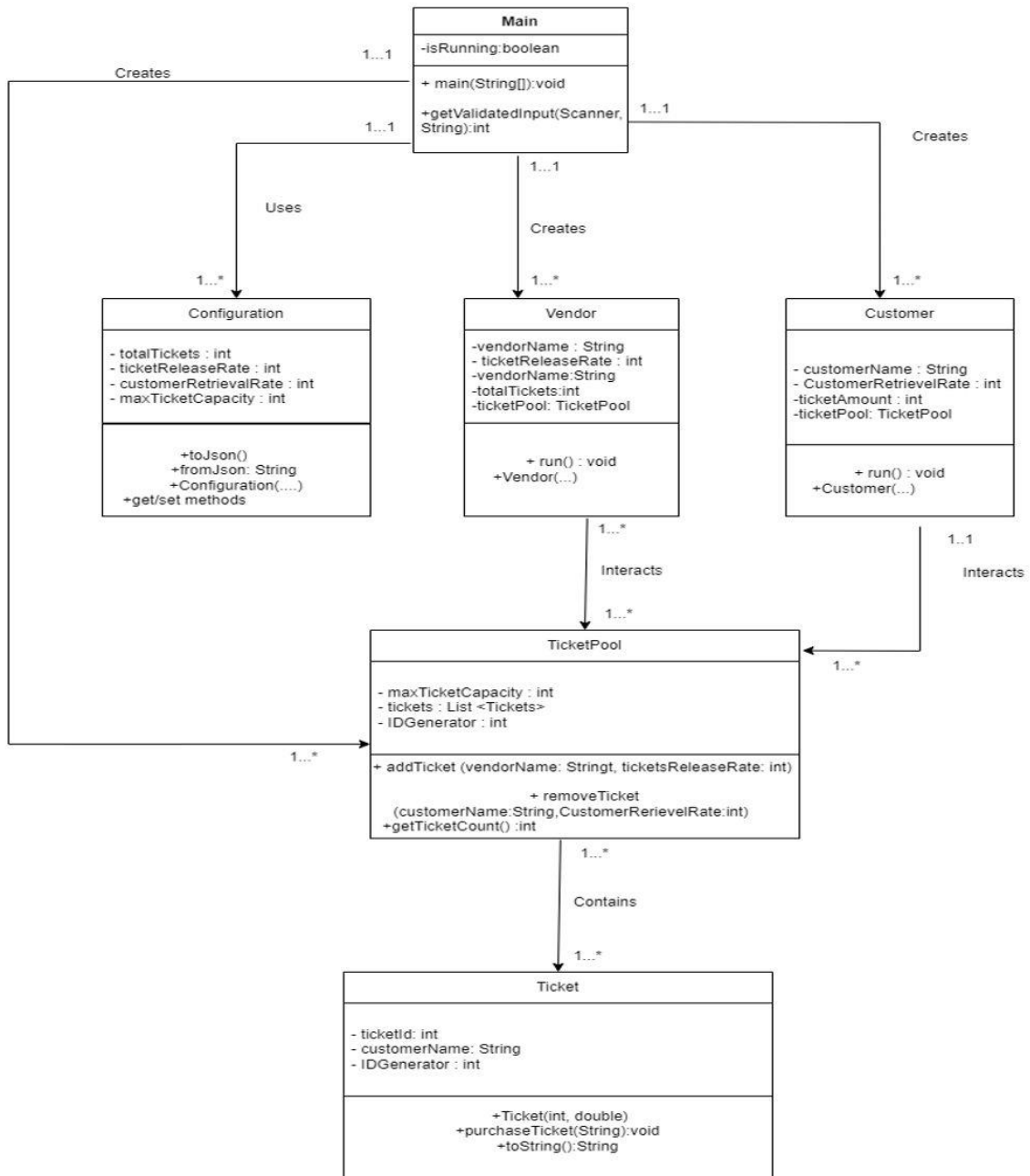| Test Scenario | Input | Expected Result | Status |
|---|---|---|---|
| Verify system accepts valid configurations | Input valid maxTicketCapacity, ticketReleaseRate, and customerRetrievalRate | System saves the configuration and initializes correctly | PASS |
| Verify system rejects invalid configuration values | Input negative or zero values for any configuration field | System displays an error message and prompts for re-entry | PASS |
| Verify configuration is saved to a file | Use saveToFile functionality after setting configuration | A JSON file is created with correct configuration details | PASS |
| Verify configuration is saved to a file | Load an existing configuration JSON file | Configuration details are successfully loaded into the system | PASS |
| Verify tickets are added correctly by vendors | Vendors add tickets to the pool | Tickets are incremented in the pool, up to the maximum capacity | PASS |
| Verify customers retrieve tickets correctly | Customers retrieve tickets from the pool | Tickets are decremented in the pool | PASS |
| Verify thread safety during simultaneous ticket operations | Multiple vendors add and customers retrieve tickets simultaneously | System maintains integrity of the ticket count | PASS |
| Verify vendor can add tickets | A vendor attempts to add more tickets than allowed | System increments the ticket pool by 10 | PASS |
| Verify vendor cannot exceed maximum ticket pool capacity | A vendor attempts to add more tickets than allowed | System displays an error message | PASS |
| Verify vendor details are logged | Run the system and check logs for vendor actions | Vendor details and ticket transactions are logged | PASS |

## Phase 2 –Backend Implementation

| Test Scenario | Input | Expected Result |
|---|---|---|
| Save Configuration | Valid configuration data | Configuration is saved successfully and accessible via the GET endpoint. |
| Get Configuration | Fetch configuration | Returns the latest configuration settings. |
| Add Vendors | Valid vendor names | Vendors are saved and retrievable from the database. |
| Add Customers | Valid customer names | Customers are saved and retrievable from the database. |
| Start Simulation | Vendor count: 3, Customer count: 5 | Simulation starts with expected ticket addition and retrieval behavior |
| Stop Simulation | API call | All vendor and customer threads stop gracefully. |
| View Simulation Status | API call | Returns the status of running vendor and customer threads. |
| Invalid Configuration Input | Negative or missing values | Returns appropriate validation error messages. |

## Phase 3 –Frontend Implementation

| Test Scenario | Action | Expected Result |
|---|---|---|
| Load Configuration Form | Navigate to the Configuration page | Form fields for configuration appear with proper placeholders. |
| Submit Configuration Form | Fill and submit the form | Configuration is saved, and a success message is displayed. |
| Add Vendors | Use the "Add Vendor" form | Vendors are added, and the list of vendors is updated dynamically. |
| Add Customers | Use the "Add Customer" form | Customers are added, and the list of customers is updated dynamically. |
| Start Simulation | Click "Start Simulation" button | Simulation starts, and the status page reflects the active vendors and customers. |
| Stop Simulation | Click "Stop Simulation" button | Simulation stops, and the status page reflects no active vendors or customers. |

| View Simulation Status | Navigate to the Status page | Status page shows the live count of vendors and customers actively participating in the simulation. |
|---|---|---|
| Navigation Links | Use navigation menu | All links direct the user to the correct pages without errors. |

## CLASS DIAGRAM

**Main**

-isRunning:boolean

+ main(String[]):void

+getValidatedInput(Scanner, String):int

1...1 — Creates

1...1 — Uses

1...1 — Creates

1...1 — Creates

**Configuration**

- totalTickets : int
- ticketReleaseRate : int
- customerRetrievalRate : int
- maxTicketCapacity : int

+toJson()
+fromJson: String
+Configuration(....)
+get/set methods

**Vendor**

-vendorName : String
- ticketReleaseRate : int
-vendorName:String
-totalTickets:int
-ticketPool: TicketPool

+ run() : void
+Vendor(...)

**Customer**

- customerName : String
- CustomerRetrievelRate : int
-ticketAmount : int
-ticketPool: TicketPool

+ run() : void
+Customer(...)

Interacts

Interacts

**TicketPool**

- maxTicketCapacity : int
- tickets : List <Tickets>
- IDGenerator : int

+ addTicket (vendorName: Stringt, ticketsReleaseRate: int)

+ removeTicket
(customerName:String,CustomerRerievelRate:int)
+getTicketCount() :int

Contains

**Ticket**

- ticketId: int
- customerName: String
- IDGenerator : int

+Ticket(int, double)
+purchaseTicket(String):void
+toString():String

## SEQUENCE DIAGRAM