

UNIVERSITY OF  
WESTMINSTER 



INFORMATICS  
INSTITUTE OF  
TECHNOLOGY

**Informatics Institute of Technology (IIT Sri Lanka)**

**Object-Oriented Programming**

**5COSC019C.1**

**Event-Hive Ticket System**

**Name - M. Jude Maria Bevan**

**UoW no - w1959430**

**IIT no - 20223075**

# 1. Table of Contents

<b>Event-Hive Ticket System</b>	<b>1</b>
1. Table of Contents	2
2. Introduction	3
3. Test Cases for Event-System CLI	3
4. Test Cases for Event-Hive configurations	5
5. API Endpoints	7
5.1 Configs	7
5.2 Tickets	8
5.3 Login, Registration	10
6. Conclusion	13

## 2. Introduction

An **Event Ticketing System** is a digital platform enabling efficient event management and ticket sales. It streamlines tasks like online ticketing, and secure access control using threads, ensuring a seamless experience for producers and consumers. This system enhances efficiency, reduces errors, and provides real-time insights, catering to events of all scales.

## 3. Important Links

- 3.1. GitHub repository - <https://github.com/judebevan/event-ticketing-system>
- 3.2. API documentation - [Event-Hive API collection](#)
- 3.3. Swagger (localhost) - [swagger](#)
- 3.4. YouTube link - [event-system-YouTube](#)

## 4. Test Cases for Event-System CLI

### 4.1. String Inputs

```
Welcome to the Event Ticketing System!  
Enter Total Tickets: s  
Invalid input. Please enter a valid integer.  
Re-enter Total Tickets: e  
Invalid input. Please enter a valid integer.  
Re-enter Total Tickets: |
```

### 4.2. Negative integers

```
Welcome to the Event Ticketing System!  
Enter Total Tickets: -2  
Total Tickets cannot be negative or zero.  
Re-enter Total Tickets: |
```

4.3. Below, test cases are tested using my CLI but cannot add screenshot proofs in this document, it contains large amount of logs.

Test Case	Total Tickets	Ticket Release Rate (seconds)	Customer Retrieval Rate (seconds)	Max Ticket Capacity	Vendor Count	Customer Count	Actual Outcome
1	10	1	1	5	2	2	Matches expected. Balanced production and consumption.
2	15	2	1	5	3	1	Matches expected. Customers waited.
3	20	1	3	10	2	2	Matches expected. Vendors paused at full capacity.
4	25	2	2	10	1	1	Matches expected. Handled concurrent operations.
5	0	1	1	10	2	2	Matches expected. Cannot input 0 values
6	10	1	1	10	3	3	Matches expected. No race conditions.
7	100	1	2	20	1	4	Matches expected. Handled high load effectively.

## 5. Test Cases for Event-Hive configurations

### 5.1. String inputs

**Ticketing System Configuration**

Total Tickets  
d  
Total tickets must be a positive number.

Ticket Release Rate  
  
Ticket release rate must be a positive number.

Customer Retrieval Rate

Max Ticket Capacity

Start

### 4.2. negative integer inputs

**Ticketing System Configuration**

Total Tickets  
-9  
Total tickets must be a positive number.

Ticket Release Rate  
4

Customer Retrieval Rate

Max Ticket Capacity

Start

### 4.3. Max Ticket Capacity should not be grater than Total Tickets

**Failed to save configuration!**

Total Tickets  
10

Ticket Release Rate  
5

Customer Retrieval Rate  
4

Max Ticket Capacity  
20

Start

Inspector Console Debugger Network State Editor Performance Memory Storage Accessibility Application

POST localhost:8080 set-config java 411 B 41 B

Response Payload

1 Max capacity cannot exceed total tickets...

### 4.4. Ticket release rate should not exceed max capacity.

**Failed to save configuration!**

Total Tickets  
10

Ticket Release Rate  
25

Customer Retrieval Rate  
4

Max Ticket Capacity  
8

Start

Inspector Console Debugger Network State Editor Performance Memory Storage Accessibility Application

POST localhost:8080 set-config java 411 B 41 B

Response Payload

1 Ticket release rate cannot exceed max capacity

#### 4.5. Vendor cannot add tickets more than the Max Ticket Capacity, Total Tickets.

**Ticket Pool Status**  
Available Tickets: 40, Released Tickets: 290, Total Tickets: 300, Max Ticket Capacity: 10

Start Stop

Add Tickets Purchase Tickets

15 0

Add Tickets Purchase Tickets

Failed to add tickets.

Status	Method	Domain	File	Initiator	Type	Transferred	Size
200	GET	localhost:8080	status	localhost:8080/status	XHR	494 B	87 B
500	POST	localhost:8080	add-tickets	localhost:8080/add-tickets	XHR	430 B	60 B
200	GET	localhost:8080	get-config	localhost:8080/get-config	XHR	506 B	99 B
200	GET	localhost:8080	status	localhost:8080/status	XHR	494 B	87 B

#### 4.6. Customer cannot purchase tickets more than the available tickets in the pool.

**Ticket Pool Status**  
Available Tickets: 40, Released Tickets: 290, Total Tickets: 300, Max Ticket Capacity: 10

Start Stop

Add Tickets Purchase Tickets

Enter ticket count 50

Add Tickets Purchase Tickets

Failed to purchase tickets.

Status	Method	Domain	File	Initiator	Type	Transferred	Size
200	GET	localhost:8080	get-config	localhost:8080/get-config	XHR	506 B	99 B
200	GET	localhost:8080	status	localhost:8080/status	XHR	494 B	87 B
500	POST	localhost:8080	purchase-tickets	localhost:8080/purchase-tickets	XHR	412 B	42 B
200	GET	localhost:8080	purchase-tickets	localhost:8080/purchase-tickets	XHR	494 B	87 B

#### 4.7. Once event started, stop button will be enabled and start button will remain disabled, and when we click stop button the start button will be disabled.

**Ticket Pool Status**  
Available Tickets: 40, Released Tickets: 290, Total Tickets: 300, Max Ticket Capacity: 10

Start Stop

Add Tickets Purchase Tickets

Enter ticket count 50

Add Tickets Purchase Tickets

System has started. Running...

Go to Dashboard

**Ticket Pool Status**  
Available Tickets: 40, Released Tickets: 290, Total Tickets: 300, Max Ticket Capacity: 10

Start Stop

Add Tickets Purchase Tickets

Enter ticket count 50

Add Tickets Purchase Tickets

System has stopped. No transactions will be made.

Go to Dashboard

## 6. API Endpoints

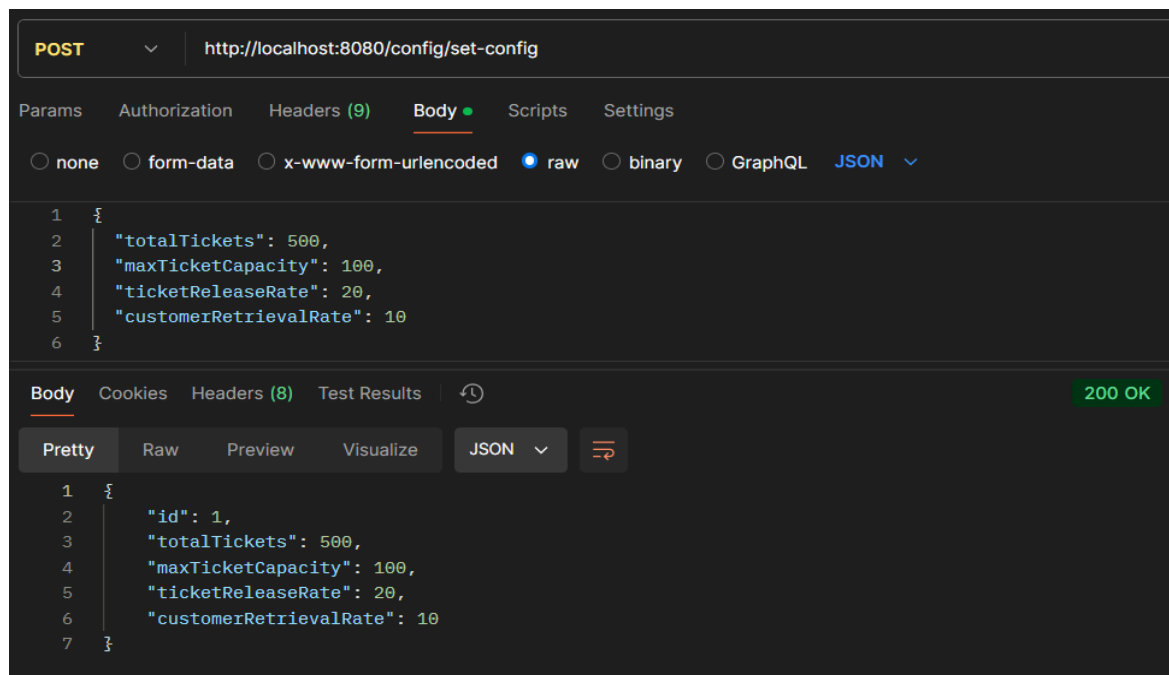
### 5.1 Configs

Method - POST set-config

Endpoint - <http://localhost:8080/config/set-config>

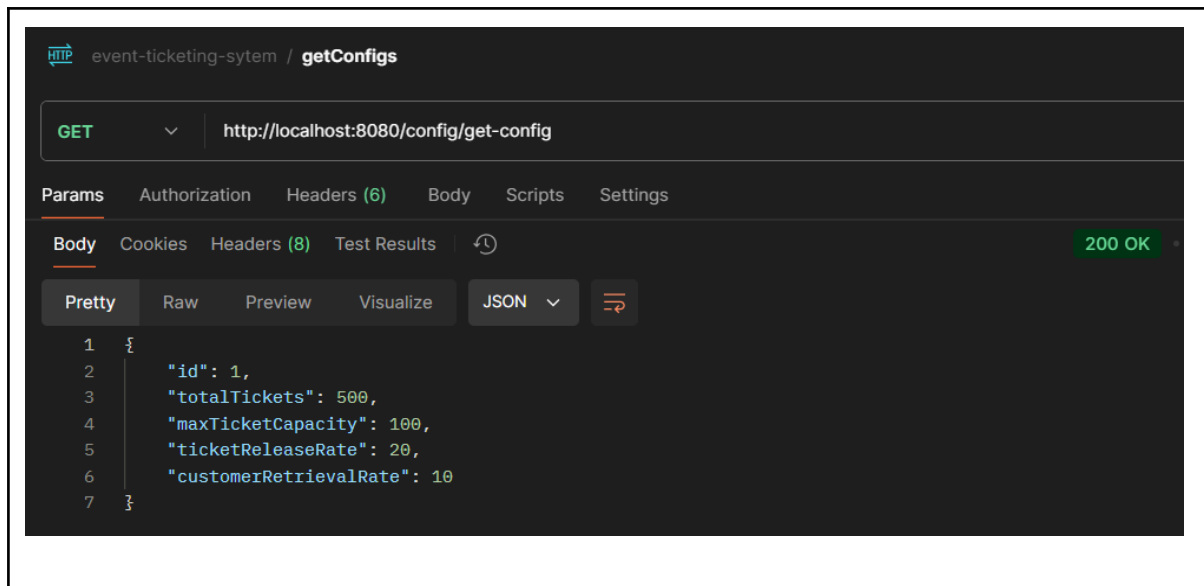
Request body:

```
{
  "totalTickets": 500,
  "maxTicketCapacity": 100,
  "ticketReleaseRate": 20,
  "customerRetrievalRate": 10
}
```



Method - GET get-config

Endpoint - <http://localhost:8080/config/get-config>



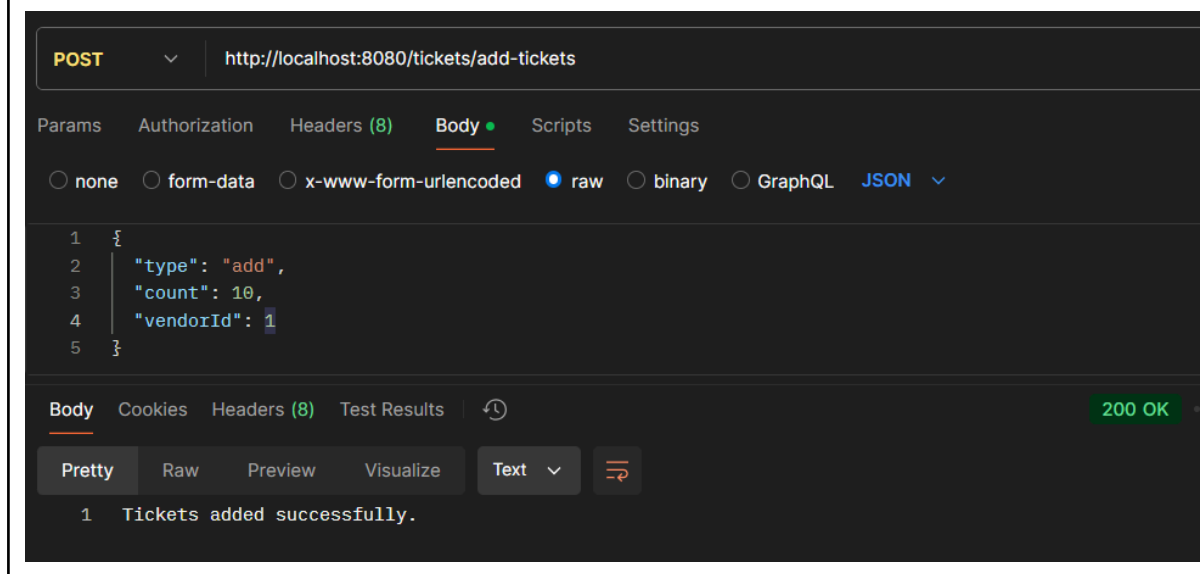
## 5.2 Tickets

Method - POST add-tickets

Endpoint - <http://localhost:8080/tickets/add-tickets>

Request body:

```
{
  "type": "add",
  "count": 10,
  "vendorId": 1
}
```





Method - POST purchase-tickets

Endpoint - <http://localhost:8080/tickets/purchase-tickets>

Request body:

```
{
  "type": "purchase",
  "count": 50,
  "customerId": 1
}
```

The screenshot shows a REST client interface with the following details:

- Method:** POST
- Endpoint:** <http://localhost:8080/tickets/purchase-tickets>
- Body Tab:** Selected, showing a JSON request body:

```
1 {
2   "type": "purchase",
3   "count": 50,
4   "customerId": 1
5 }
```
- Response:** 200 OK
- Body Tab (Pretty):** Shows the response text: `1 Tickets purchased successfully.`

Method - GET status

Endpoint - <http://localhost:8080/tickets/status>

The screenshot shows a REST client interface with the following details:

- Method:** GET
- Endpoint:** <http://localhost:8080/tickets/status>
- Body Tab:** Selected, showing a JSON response body:

```
1 {
2   "availableTickets": 10,
3   "releasedTickets": 310,
4   "totalTickets": 500,
5   "maxTicketCapacity": 100
6 }
```
- Response:** 200 OK

## 5.3 Login, Registration

Method - POST vendor-register

Endpoint - <http://localhost:8080/vendor/register>

Request body:

```
{
  "username": "judebevan",
  "email": "judebevan@gmail.com",
  "password": "jude@123",
  "mobileNo": 705689123,
  "isAdmin": true
}
```

POST <http://localhost:8080/vendor/register>

Params Authorization Headers (8) **Body** Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▾

```
1 {
2   "username": "judebevan",
3   "email": "judebevan@gmail.com",
4   "password": "jude@123",
5   "mobileNo": 705689123,
6   "isAdmin": true
7 }
```

Body Cookies Headers (8) Test Results **200 OK**

Pretty Raw Preview Visualize **JSON** ▾

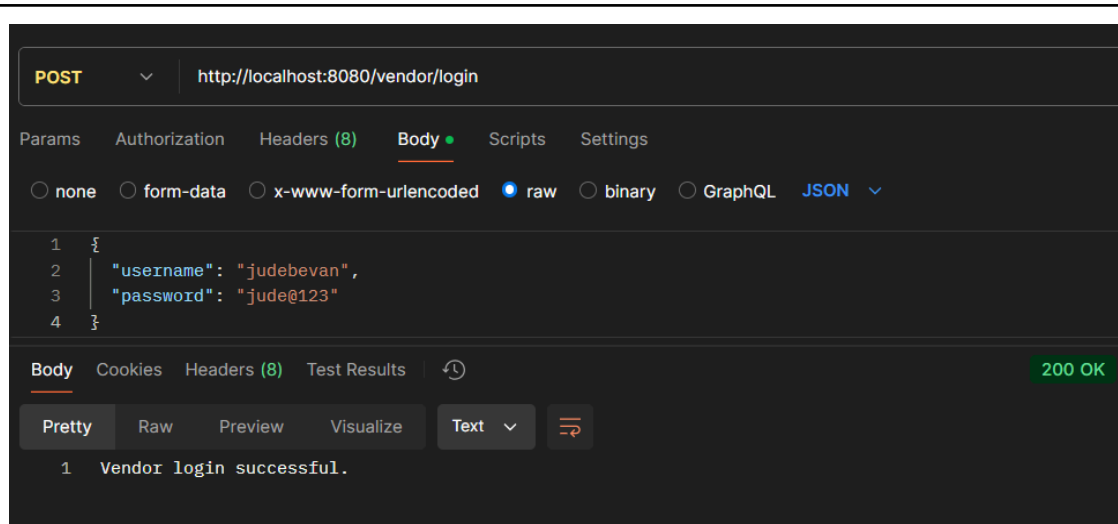
```
1 {
2   "id": 3,
3   "username": "judebevan",
4   "email": "judebevan@gmail.com",
5   "password": "jude@123",
6   "mobileNo": 705689123,
7   "isAdmin": true
8 }
```

Method - POST vendor-login

Endpoint - <http://localhost:8080/vendor/login>

Request body:

```
{
  "username": "judebevan",
  "password": "jude@123"
}
```

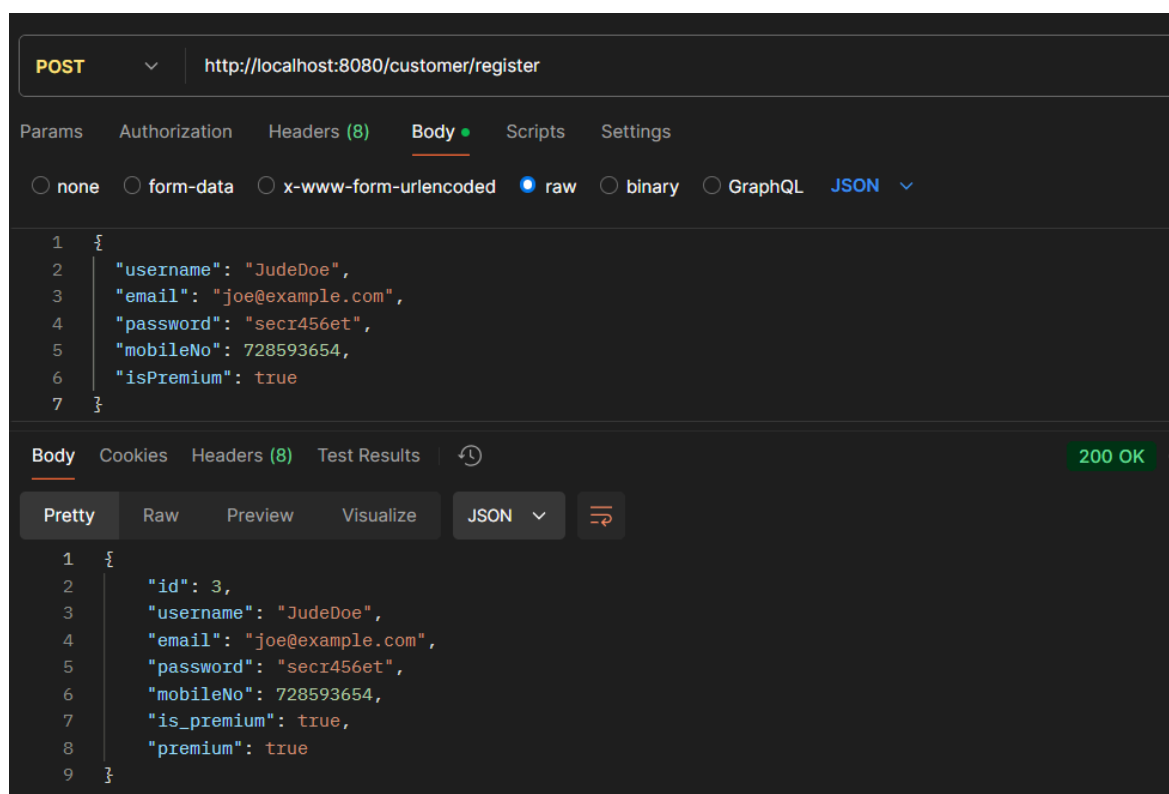


Method - POST customer-register

Endpoint - <http://localhost:8080/customer/register>

Request body:

```
{
  "username": "JudeDoe",
  "email": "joe@example.com",
  "password": "secr456et",
  "mobileNo": 728593654,
  "isPremium": true
}
```



Method - POST customer-login

Endpoint - <http://localhost:8080/customer/login>

Request body:

```
{  
  "username": "JudeDoe",  
  "password": "secr456et"  
}
```

The screenshot displays a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:8080/customer/login
- Body Type:** raw (selected)
- Body Content:**

```
1 {  
2   "username": "JudeDoe",  
3   "password": "secr456et"  
4 }
```
- Response Status:** 200 OK
- Response Body:**

```
1 Customer login successful.
```

## **7. Conclusion**

The Event Ticketing System successfully demonstrates its ability to streamline event management and ticket sales through efficient automation and robust handling of various scenarios. The system's key features, such as real-time ticket production and consumption, concurrent vendor-customer operations, and edge case handling, have been rigorously tested and validated.

By providing a user-friendly and scalable solution, the system eliminates traditional inefficiencies and enhances the overall event management experience. It is capable of handling high loads, ensuring data consistency, and meeting modern event ticketing requirements. This project highlights the importance of integrating technology to optimize event workflows, making it a valuable tool for organizers and attendees alike.