UNIVERSITY OF WESTMINSTER

INFORMATICS INSTITUTE OF TECHNOLOGY

Informatics Institute of Technology

University of Westminster

Object Oriented Programming

5COSC019C.1

Module Leader: - Mr. Poravi Guganathan

Date of Submission: - 12/12/2024

Assignment: - Coursework – Real-Time-Ticket-System

Assignment Type: - Individual

Dhanushi Dewmindi Panamulla Arachchi

UOW No – w2051668

IIT No – 20230443

Tutorial Group: - L5 CS-G16

Table of Contents

1. Table of generic test cases for the CLI

Test Case ID	Description	Expected Outcome	Actual Outcome
1.	Do you want to load the existing configuration? (yes/no)	“yes” System will start the configuration by retrieving the data from the already saved data from the Jason file.	Passed
		“no” System will override the Jason file with the new inputs from the user.	Passed
		The system will keep looping until the user inputs a valid input “Invalid input. Please enter 'yes' or 'no'.”	passed
2.	Enter ticket release rate (ms):	The rate of vendor release tickets (vendor thread sleep time). Inputs in between 10-10000 is allowed. No other inputs are allowed.	Passed
3.	Enter customer retrieval rate (ms):	The rate of customer ticket purchase (customer thread sleep time). Inputs in between 10-10000 is allowed. No other inputs are allowed.	Passed
4.	Enter maximum ticket capacity:	The maximum number of tickets that the ticket pool can hold. Only integers are accepted any other inputs (letters, negative numbers, symbols) are not allowed.	Passed
5.	Enter total number of tickets:	The maximum number of tickets that a vendor can release at once. Only integers are accepted any other inputs (letters, negative numbers, symbols) are not allowed.	Passed
6.	Commands: 'start' to begin, 'stop' to halt, 'exit' to quit.	“start” The system will start the threads and simulation with the config data.	Passed
		“stop” The system will hold the process until the user starts it again	Passed
		“exit” The system will save the provided config data to the Jason file and terminate the program	Passed
		The system will keep looping until the user inputs a valid input “The system will keep looping until the user inputs a valid input ”	Passed

2. Table of generic test cases for the Backend

Test Case ID	Description	Expected Outcome	Actual Outcome

1.	Adding a new customer to the system.
	
	(/customer/add-customer)
		{
		"customerId": 12,
		"name": "Cust_test",
		"vip": true
		}
		
		Ensure that the customerId is an integer, name is a non-empty string, and vip is a boolean (true/false). Any invalid input, such as incorrect data types or missing fields, will result in a **400 Bad Request** response. This guarantees that only properly structured requests are accepted by the system, ensuring data integrity and accurate processing.
			Passed

		{
		"code": 201,
		"message": "success",
		"data": {
		"customerId": 9,
		"name": "Cust_3",
		"vip": true
		}
		}
		
		If the inserted data is valid the new customer will be added to the system.
			Passed

2.	Deleting a customer from the database
	
	(/customer/delete-customer)
		{
		"code": 201,
		"message": "Success",
		"data": "Customer not found!"
		}
		
		Provide the customerId of the customer you want to delete. If the specified ID does not exist, an appropriate error message will be returned.
			Passed

		{ "code": 201, "message": "Success", "data": "Customer deleted successfully!"	Passed

		} If the inserted Id is available the relevant customer will be deleted. Any incorrect data types or missing fields will result in a **400 Bad Request** response.	
3.	View all the customers in the database (customer/get-all-customer)	All the available customers will be displayed with the relevant name, id and the status	Passed

4.	Get a customer detail by customer id.
	
	(customer/get-customer-by-id)
		{
		"code": 404,
		"message": "Customer not found",
		"data": null
		}
		
		If the id given is not available in the database response data will be null and pass a message error message.
		Any incorrect data types or missing fields will result in a **400 Bad Request** response.
			Passed

		{
		"code": 200,
		"message": "Success",
		"data": {
		"customerId": 71,
		"name": "Cust_3",
		"vip": true
		}
		}
		
		If the given Id is found the data of the relevant customer will be displayed. Any incorrect data types or missing fields will result in a **400 Bad Request** response.
			Passed

5.	Update a customer saved in the database.
	
	(customer/update-customer)
		{
		"customerId": 49,
		"name": "Updated_Cust",
		"vip": false
		}
		
		If the id given is not available in the database response data will be null and pass a message error message.
		Any incorrect data types or missing fields will result in a **400 Bad Request** response.
			Passed

		{	Passed

		"code": 201, "message": "Success", "data": "Customer has been updated successfully..." } If the id inserted is found the customer will be updated and pass a success message.	
6.	View all the tickets in the database (/ticket/get-all-tickets)	All the available and non-available tickets will be displayed.	Passed
7.	Get all the available tickets in the database (/ticket/get-available-tickets)	Only the available tickets will be displayed.	Passed

8.	Purchasing a ticket by the customer
	
	(/ticket/purchase-ticket)
		{
		"code": 201,
		"message": "Success",
		"data": "Ticket is not found or already sold...!"
		}
		
		If the customerId or the ticketId is not available this message will be passed. Any incorrect data types or missing fields will result in a **400 Bad Request** response.
			Passed

		{
		"code": 201,
		"message": "Success",
		"data": "Ticket purchased successfully by customer : Cust_3"
		}
		
		If both the inserted ids are found to be valid and available. The ticket will be purchased and the success message will be passed. And the purchase details will be saved to the purchase table.
			Passed

9.	Adding a new vendor to the system.
	
	(/vendor/add-vendor)
		{
		"code": 201,
		"message": "success",
		"data": "New vendor added...!"
		}
			Passed

		If all the data inserted is valid and acceptable the new vendor will be added to the system and the success message will be passed.	

		{
		"timestamp": "2024-12-10T19:17:52.734+00:00",
		"status": 400,
		"error": "Bad Request",
		"path": "/vendor/add-vendor"
		}
		
		Ensure that the vendorId is an integer, name is a non-empty string, and ticketReleaserate is a integer . Any invalid input, such as incorrect data types or missing fields, will result in a **400 Bad Request** response. This guarantees that only properly structured requests are accepted by the system, ensuring data integrity and accurate processing
			Passed

10.	Deleting a customer from the database
	
	(/customer/delete-customer)
		{
		"code": 201,
		"message": "Success",
		"data": "Vendor not found!"
		}
		
		Provide the vendorId of the vendor you want to delete. If the specified ID does not exist, an appropriate error message will be returned.
			Passed

		{
		"code": 201,
		"message": "Success",
		"data": "Vendor deleted successfully!"
		}
		
		If the inserted Id is available the relevant vendor will be deleted. Any incorrect data types or missing fields will result in a **400 Bad Request** response.
			Passed

11.	View al the available vendors (/vendor/get-all-vendors)	All the vendors saved in the database will be displayed here.	Passed

12.	Updating an existing vendor
	
	(/vendor/update-vendor)
		{
		"code": 200,
		"message": "success",
		"data": "Vendor not found!"
		}
			Passed

		If the id given is not available in the database response data will be null and pass a message error message. Any incorrect data types or missing fields will result in a **400 Bad Request** response.	

		{
		"code": 200,
		"message": "success",
		"data": "Vendor updated successfully!"
		}
		If the id inserted is found the vendor will be updated with the new data and passes a success message.
			Passed

13.	Starting a vendor thread by a vendor	{
		"price": 100,
		"ticketCount": 2,
		"ticketType": "CONCERT"
		}
		
		If all the field have valid inputs, an existing vendor can start their thread by giving their vendor Id. Any incorrect data types or missing fields will result in a **400 Bad Request** response.
			Passed

		{
		"code": 201,
		"message": "Success",
		"data": "Vendor with id 2 not found"
		}
		
		If the vendor Id is not matching with the data in the database the request will be denied.
			Passed

		{
		"code": 201,
		"message": "Success",
		"data": "Open a new Thread for vendor: vendor_10"
		}
		
		If the data required for the thread and the vendorId is valid and acceptable. The thread will be started while passing a success message.
			Passed

14.	Stopping a vendor thread by a vendor	{
		"code": 201,
		"message": "Success",
		"data": "vendor thread stopped..."
		}
			Passed

		If the vendor Id is not matching with the data in the database the request will be denied.	

		{
		"code": 201,
		"message": "Success",
		"data": "Vendor with id 2 not found"
		}
		
		If the vendor Id is not matching with the data in the database the request to stop the thread will be denied.
			Passed

3. Table of generic test cases for the Frontend

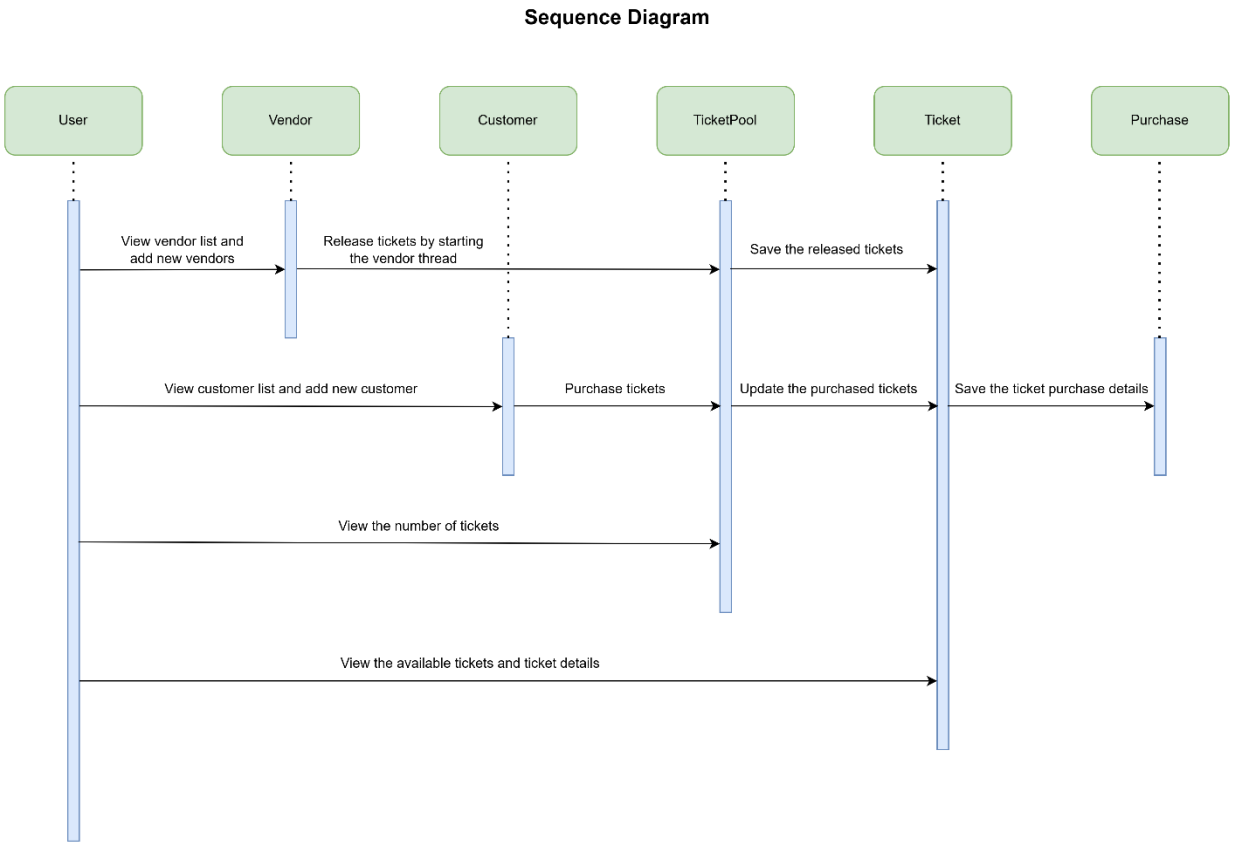
Test Case ID	Description	Expected Outcome	Actual Outcome
1.	Add customer form	All the required data should be in the correct data type (customerId-integer, Customer Full name-string) customer status can be selected from a drop-down menu. If the values do not match the requested type Customer will not be added an error message will displayed.	Passed
		If all the data types are valid new customers will be added to the table	Passed
2.	Update customer form	All the saved data will get fetched to the update form. If all the data inserted matches the requested data, the customer will be updated. Otherwise, an error message will be displayed.	Passed
3.	Delete customer	When the user presses the “Delete” button the relevant customer will be deleted. Before deleting the customer the user has to confirm the delete request just to be sure.	Passed
4.	Add vendor form	All the required data for the vendor must be in the correct data type (vendorId - integer, Vendor Name - string). The vendor status can be selected from a drop-down menu. If the values do not match the expected data types, the vendor will not be added, and an appropriate error message will be displayed.	Passed
		If all the data types are valid new customers will be added to the table	Passed

5.	Update vendor form	All the saved data will get fetched to the update form. If all the data inserted matches the requested data, the vendor will be updated. Otherwise, an error message will be displayed.	Passed
6.	Delete vendor	When the user presses the “Delete” button, the relevant vendor will be deleted. Before deleting the vendor, the user must confirm the delete request to ensure the action is intentional.	Passed
7.	Start vendor thread	All the required data for the vendor thread to start must be in the correct data type. If the values do not match the expected data types, the vendor will not be added, and an appropriate error message will be displayed. (Each vendor can activate their thread with the button in front of them)	Passed
		If all the data types are valid new vendor thread will be started, and the tickets will be added to the table.	Passed
8.	Stop vendor thread	By pressing the button corresponding to a specific vendor, the user can stop the thread associated with that vendor, effectively halting its operation.	Passed
9.	Purchasing a ticket by a registered customer	Customers must enter their **Customer ID** to access the ticket pool and purchase tickets. Once authenticated, they can select from the available tickets to complete their purchase.	Passed
		If the customer Id is not valid or all non of the tickets are available an error msg will be displayed.	Passed

4. Observations, issues, and solutions of this project

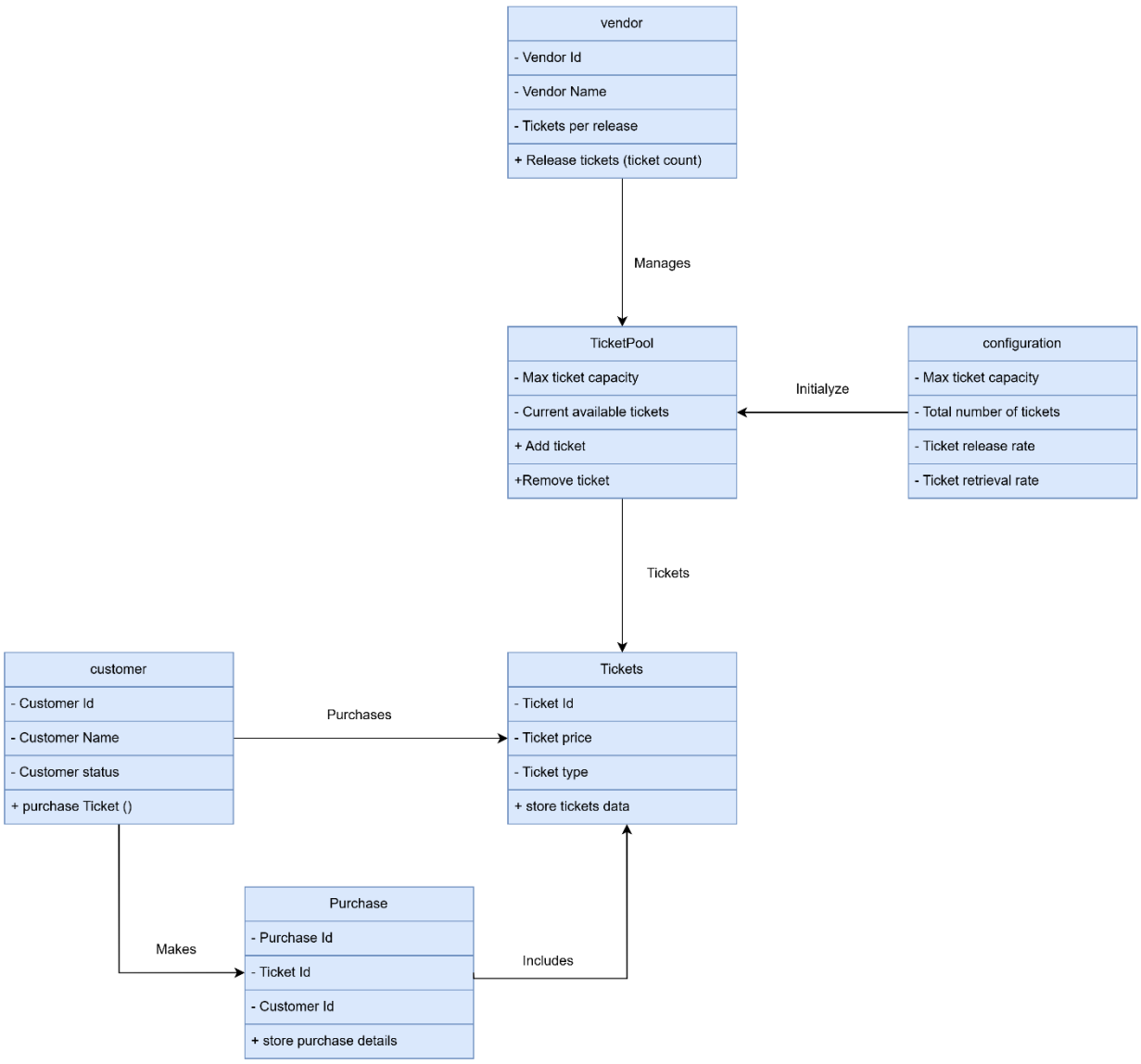
Id	Description of the Issue	Impact	Solution Implemented
1.	The "Do you want to load the existing configuration? (yes/no)" prompt was not functioning correctly.	Prevented proper loading of configuration.	Adjusted the input handling logic to correctly parse and handle user responses
2.	Vendors and customers occasionally clashed due to unsynchronized ticket handling.	Caused concurrency issues.	Added synchronized blocks and wait/notify mechanisms to ensure proper thread coordination.
3.	Threads were not stopping properly after all tickets were sold out.	The system remained active unnecessarily.	Used a stopTicketHandling method with executor.shutdownNow() and added appropriate termination logic to ensure proper thread closure.

5. Sequence Diagram



6. Class Diagram

Class Diagram



7. References

[1] "ChatGPT," 19 November 2024. [Online]. Available: https://chat.openai.com/.

[2] "W3 Schools - React, Bootstrap, Java," 5 12 2024. [Online]. Available: https://www.w3schools.com/.

[3] "Java Programming Exercises, Practice, Solution," 11 11 2024. [Online]. Available: https://www.w3resource.com/java-exercises/.

[4] J. Guides, "Spring Boot REST API Project Course | Build 2 Spring Boot REAL-TIME REST API Projects in 3 Hours," 08 2024. [Online]. Available: https://www.youtube.com/@JavaGuides.

[5] S. Smith, "Building Real-Time Web Apps with Spring Boot, webSockets," 10 08 2018. [Online]. Available: https://www.linkedin.com/learning/building-real-time-web-apps-with-spring-boot-and-websockets?u=76664938.

8. Appendices

Test case proof for the CLI

```
C:\Users\USER\.jdks\openjdk-21.0.2\bin\java.exe ...
Do you want to load the existing configuration? (yes/no)
yes
Configuration loaded successfully.
20:33:39.119 [main] INFO org.example.oop_cw.CLI.TicketingSystemCLI -- Starting command loop.
Commands: 'start' to begin, 'stop' to halt, 'exit' to quit.
Command: start
20:34:08.345 [main] INFO org.example.oop_cw.CLI.TicketingSystemCLI -- Starting ticket handling.
20:34:08.355 [pool-1-thread-1] INFO org.example.oop_cw.ticket.TicketPool -- Tickets added: 10, Total tickets: 10
20:34:08.357 [pool-1-thread-1] INFO org.example.oop_cw.ticket.Vendor -- Vendor 1 Added tickets. The ticket pool contains:10
20:34:08.361 [pool-1-thread-1] INFO org.example.oop_cw.ticket.Vendor -- Waiting for 1500
20:34:09.864 [pool-1-thread-1] INFO org.example.oop_cw.ticket.TicketPool -- Tickets added: 10, Total tickets: 20
20:34:09.865 [pool-1-thread-1] INFO org.example.oop_cw.ticket.Vendor -- Vendor 1 Added tickets. The ticket pool contains:20
20:34:09.865 [pool-1-thread-1] INFO org.example.oop_cw.ticket.Vendor -- Waiting for 1500
20:34:09.867 [pool-1-thread-2] INFO org.example.oop_cw.ticket.TicketPool -- Tickets added: 10, Total tickets: 30
20:34:09.867 [pool-1-thread-2] INFO org.example.oop_cw.ticket.Vendor -- Vendor 2 Added tickets. The ticket pool contains:30
20:34:09.867 [pool-1-thread-2] INFO org.example.oop_cw.ticket.Vendor -- Waiting for 1500
20:34:11.367 [pool-1-thread-1] INFO org.example.oop_cw.ticket.TicketPool -- Tickets added: 10, Total tickets: 40
```

```
C:\Users\USER\.jdks\openjdk-21.0.2\bin\java.exe ...
Do you want to load the existing configuration? (yes/no)
1234
Invalid input. Please enter 'yes' or 'no'.
no
Starting fresh configuration setup.
Enter ticket release rate (ms):
```

```
C:\Users\USER\.jdks\openjdk-21.0.2\bin\java.exe ...
Do you want to load the existing configuration? (yes/no)
no
Starting fresh configuration setup.
Enter ticket release rate (ms): 200000
20:41:11.484 [main] INFO org.example.oop_cw.CLI.TicketingSystemCLI -- Invalid input. Please enter an input
Please enter a value between 10 and 100000.
Enter ticket release rate (ms): 1500
Enter customer retrieval rate (ms): Abcd
20:41:29.223 [main] INFO org.example.oop_cw.CLI.TicketingSystemCLI -- Invalid input. Please enter valid input in the range 10 and 100000.
Invalid input. Please enter a valid number.
Enter customer retrieval rate (ms): 1000
Enter maximum ticket capacity: !@#$%
20:41:39.895 [main] INFO org.example.oop_cw.CLI.TicketingSystemCLI -- Invalid input. Please enter valid input in the range 1 and 2147483647.
Invalid input. Please enter a valid number.
Enter maximum ticket capacity: 120
Enter total number of tickets: 10
20:41:51.610 [main] INFO org.example.oop_cw.CLI.TicketingSystemCLI -- All the data is successfully loaded.
20:41:51.665 [main] INFO org.example.oop_cw.CLI.TicketingSystemCLI -- Configuration saved successfully.
Configuration saved successfully.
20:41:51.666 [main] INFO org.example.oop_cw.CLI.TicketingSystemCLI -- Starting command loop.
Commands: 'start' to begin, 'stop' to halt, 'exit' to quit.
Command:
```

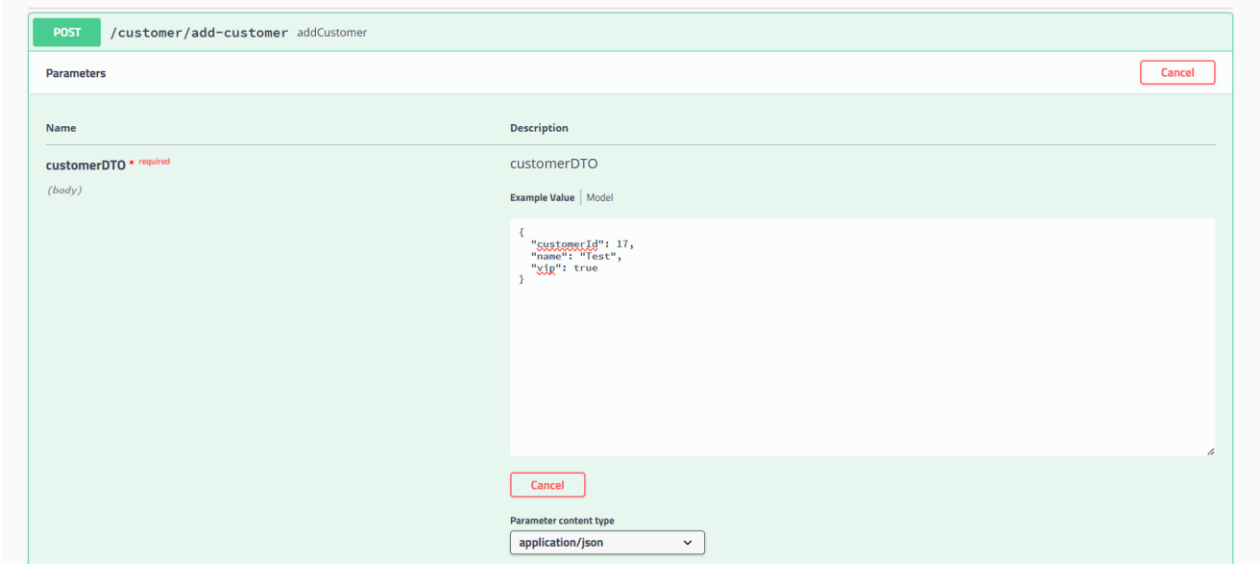
```
20:41:51.666 [main] INFO org.example.oop_cw.CLI.TicketingSystemCLI -- Starting command loop.
Commands: 'start' to begin, 'stop' to halt, 'exit' to quit.
Command: stop
System is not running.
Command: 1234
Unknown command. Please try 'start', 'stop', or 'exit'.
Command: car
Unknown command. Please try 'start', 'stop', or 'exit'.
Command: start
20:44:35.751 [main] INFO org.example.oop_cw.CLI.TicketingSystemCLI -- Starting ticket handling.
20:44:35.759 [pool-1-thread-1] INFO org.example.oop_cw.ticket.TicketPool -- Tickets added: 10, Total tickets: 10
20:44:35.761 [pool-1-thread-1] INFO org.example.oop_cw.ticket.Vendor -- Vendor 1 Added tickets. The ticket pool contains:10
20:44:35.764 [pool-1-thread-1] INFO org.example.oop_cw.ticket.Vendor -- Waiting for 1500
```

```
20:44:58.862 [pool-1-thread-1] INFO org.example.oop_cw.ticket.Customer -- Customer 5 has stopped.
20:44:58.877 [pool-1-thread-1] INFO org.example.oop_cw.ticket.Customer -- Customer 6 has stopped.
20:44:59.886 [pool-1-thread-1] INFO org.example.oop_cw.ticket.Customer -- Customer 7 has stopped.
20:45:00.898 [pool-1-thread-1] INFO org.example.oop_cw.ticket.Customer -- Customer 8 has stopped.
20:45:01.910 [pool-1-thread-1] INFO org.example.oop_cw.ticket.Customer -- Customer 9 has stopped.
20:45:02.921 [pool-1-thread-1] INFO org.example.oop_cw.ticket.Customer -- Customer 10 has stopped.
System has been stopped.
20:45:03.926 [main] INFO org.example.oop_cw.CLI.TicketingSystemCLI -- System has been stopped.
All tickets have been sold. System is stopping.
Command: exit
System has been stopped.
21:15:46.329 [main] INFO org.example.oop_cw.CLI.TicketingSystemCLI -- System has been stopped.
21:15:46.356 [main] INFO org.example.oop_cw.CLI.TicketingSystemCLI -- Configuration saved successfully.
Configuration saved successfully.

Process finished with exit code 0
```

Test cases for the Backend using Swagger API Documentation

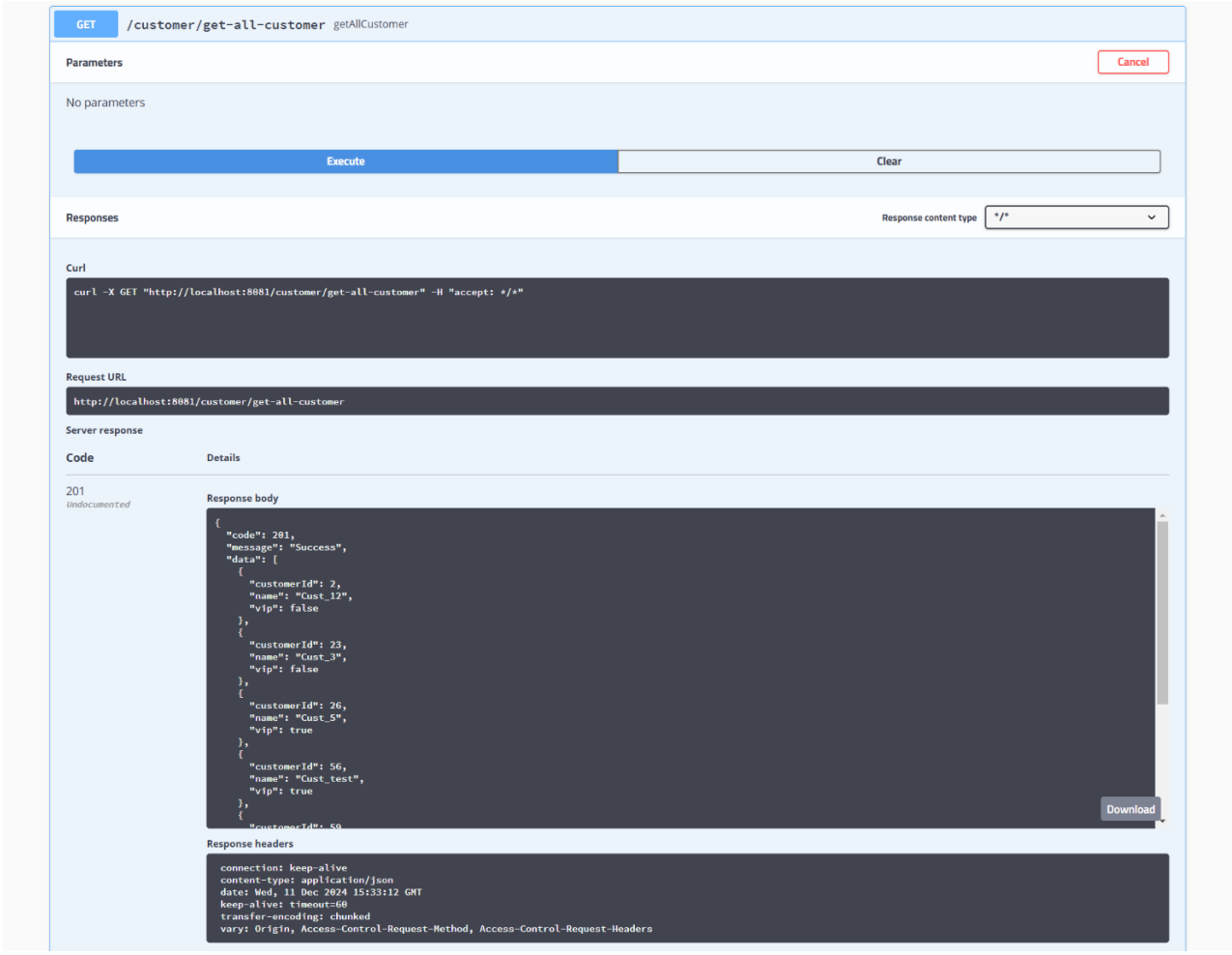
Customer functions

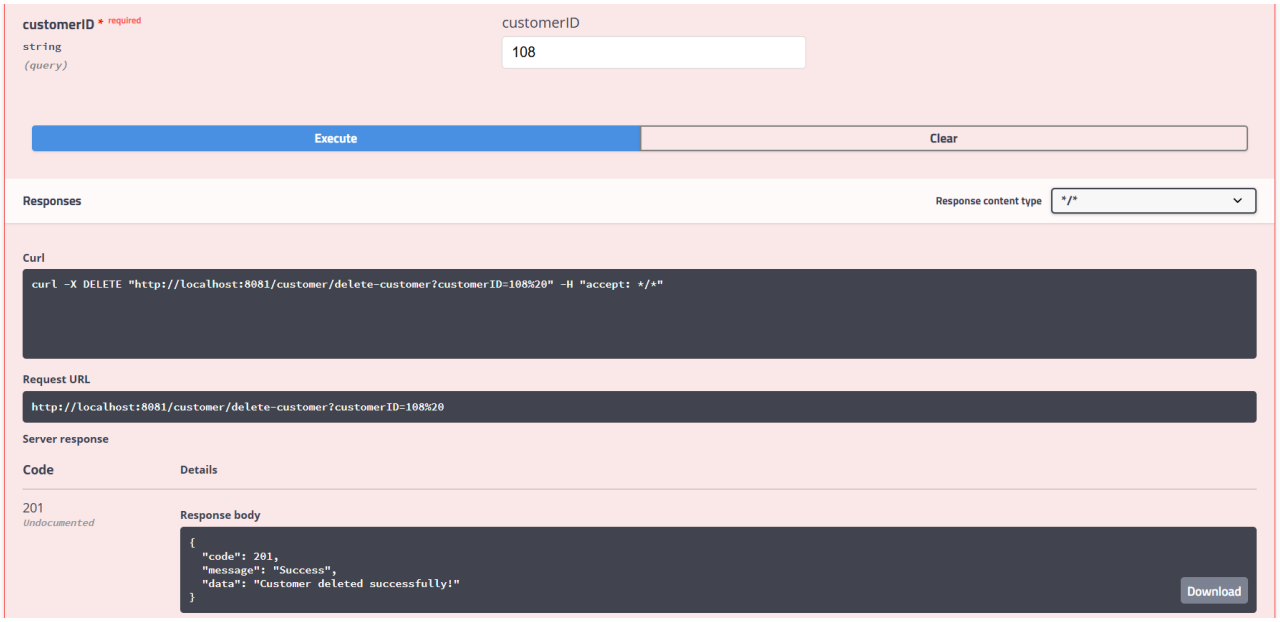


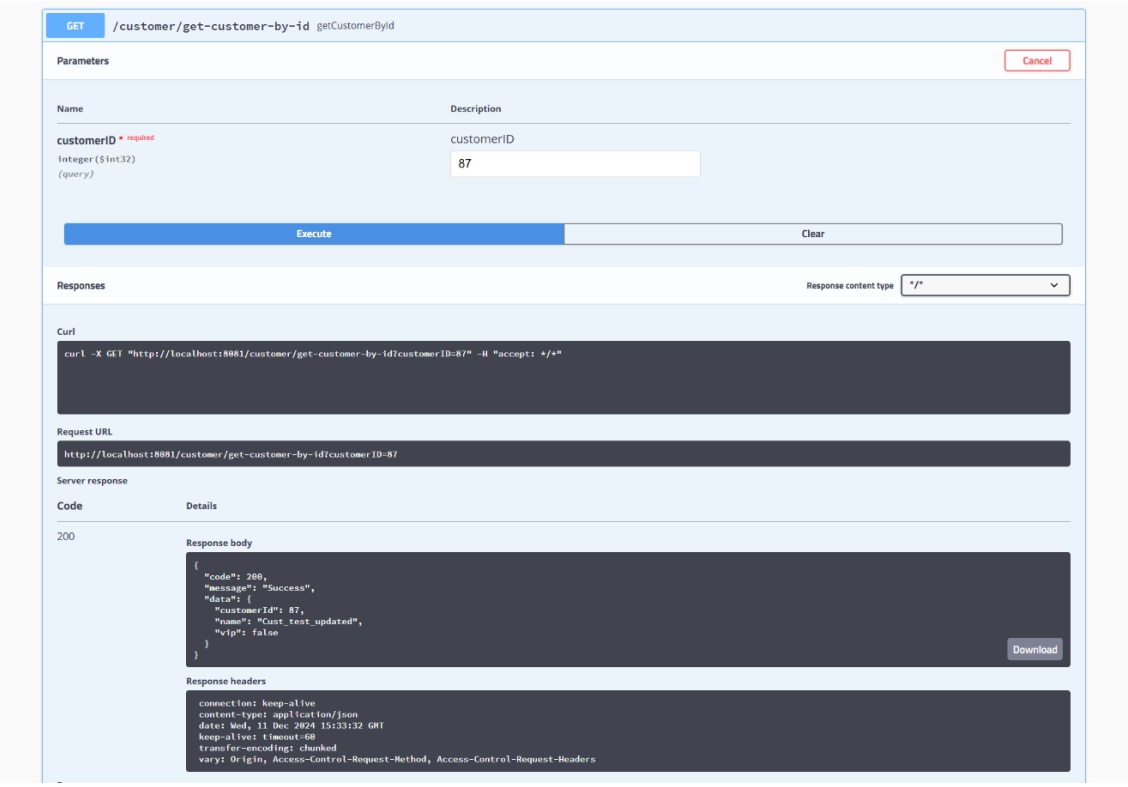
Response body

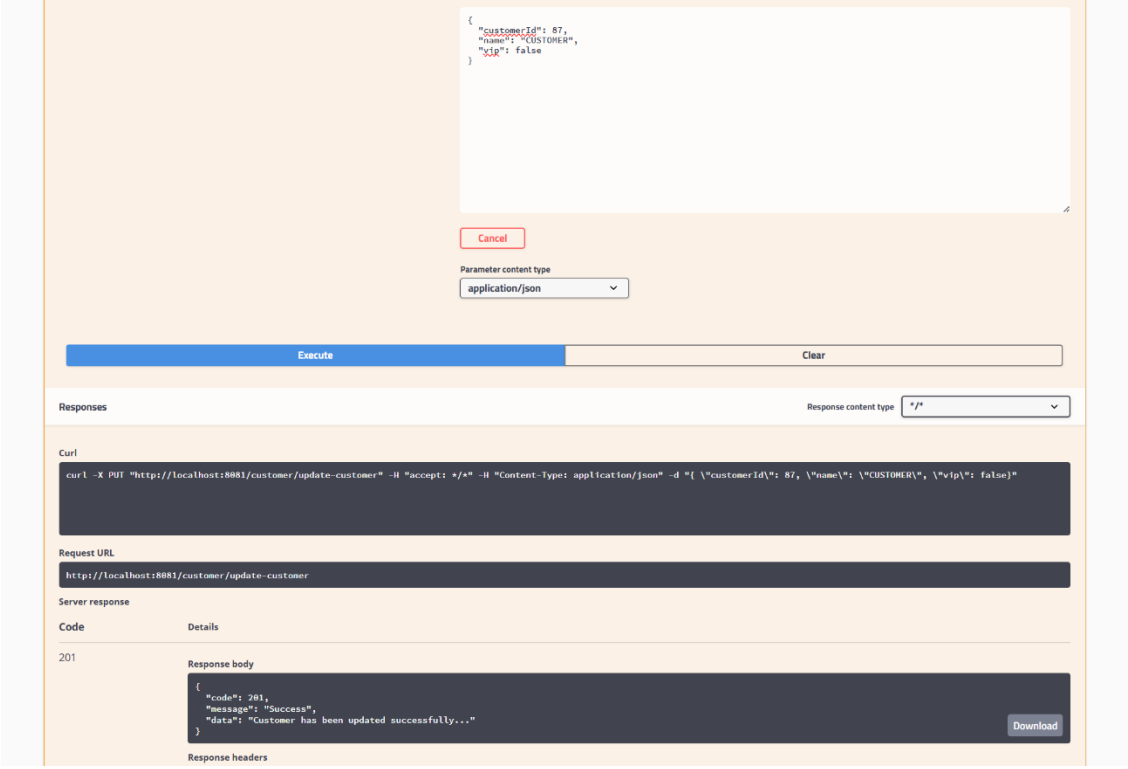

```
{
  "code": 201,
  "message": "success",
  "data": {
    "customerId": 17,
    "name": "Test",
    "vip": true
  }
}
```


Download

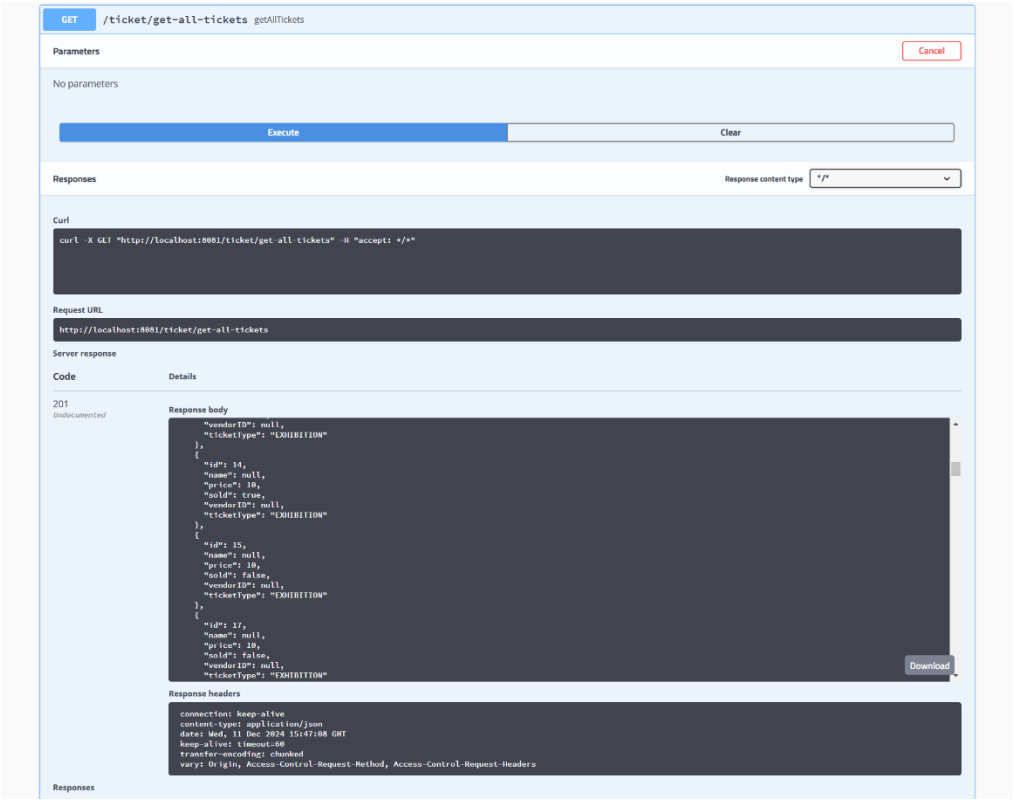


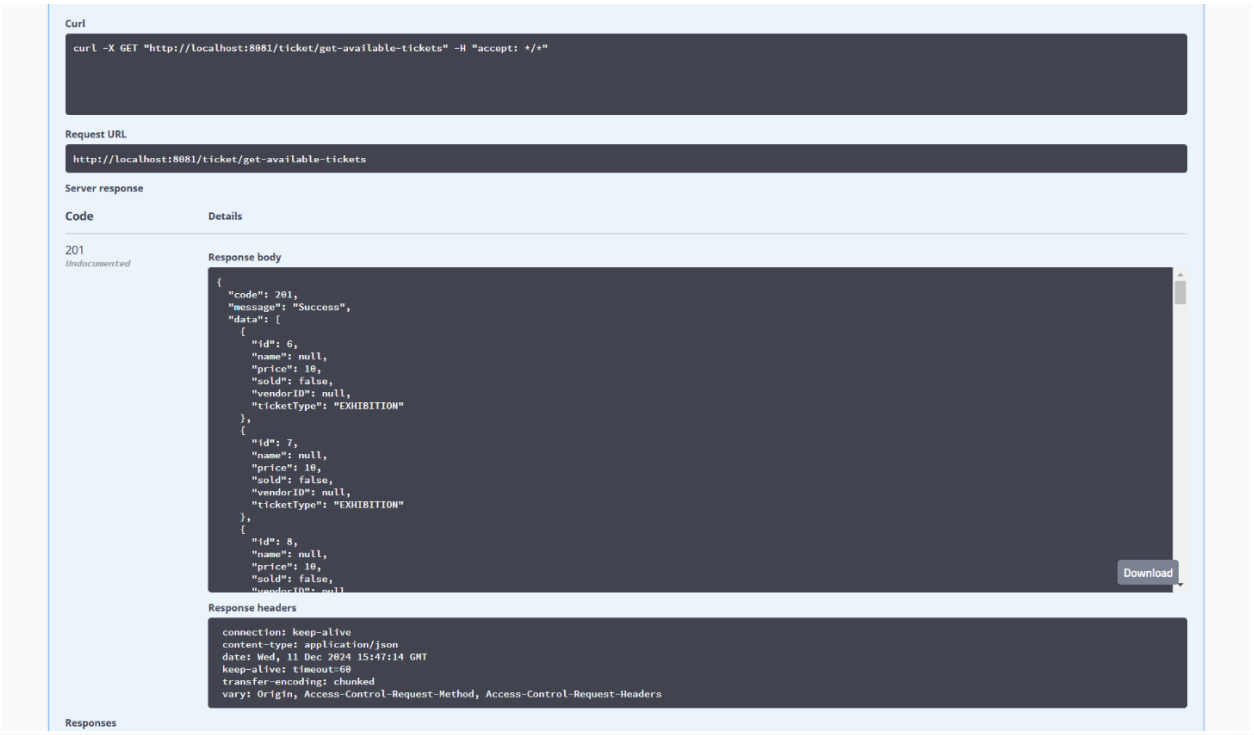

customerID * required
string
(query)
customerID
108
Execute
Clear
Responses
Response content type
/
Curl
curl -X DELETE "http://localhost:8081/customer/delete-customer?customerID=108%20" -H "accept: */*"
Request URL
http://localhost:8081/customer/delete-customer?customerID=108%20
Server response
Code
Details
201
Undocumented
Response body
{
"code": 201,
"message": "Success",
"data": "Customer deleted successfully!"
}
Download

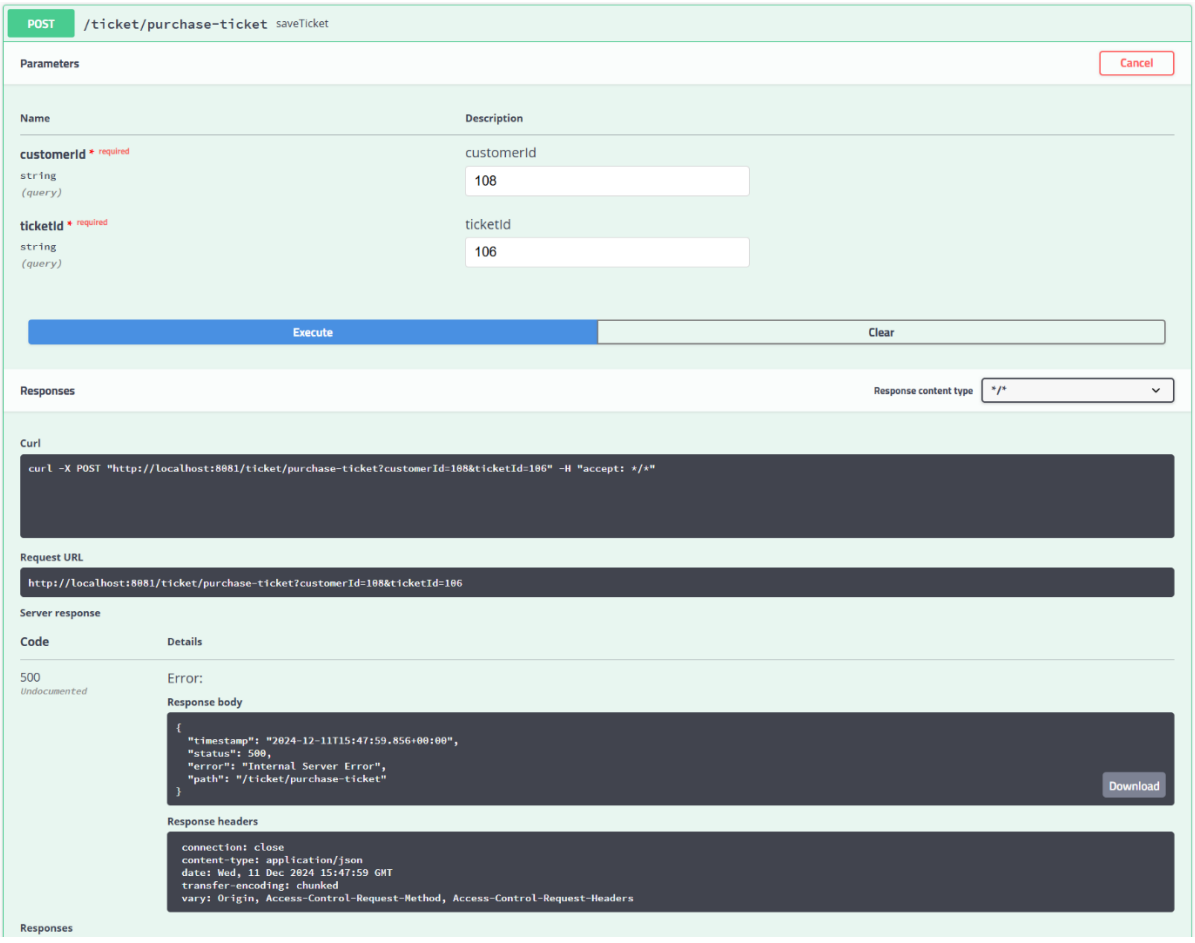

GET
/customer/get-customer-by-id
getCustomerById
Parameters
Cancel
Name
Description
customerID * required
integer($int32)
(query)
customerID
87
Execute
Clear
Responses
Response content type
/
Curl
curl -X GET "http://localhost:8081/customer/get-customer-by-id?customerID=87" -H "accept: */*"
Request URL
http://localhost:8081/customer/get-customer-by-id?customerID=87
Server response
Code
Details
200
Response body
{
"code": 200,
"message": "Success",
"data": {
"customerId": 87,
"name": "Cust_test_updated",
"vip": false
}
}
Download
Response headers
connection: keep-alive
content-type: application/json
date: Wed, 11 Dec 2024 15:33:32 GMT
keep-alive: timeout=60
transfer-encoding: chunked
vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers



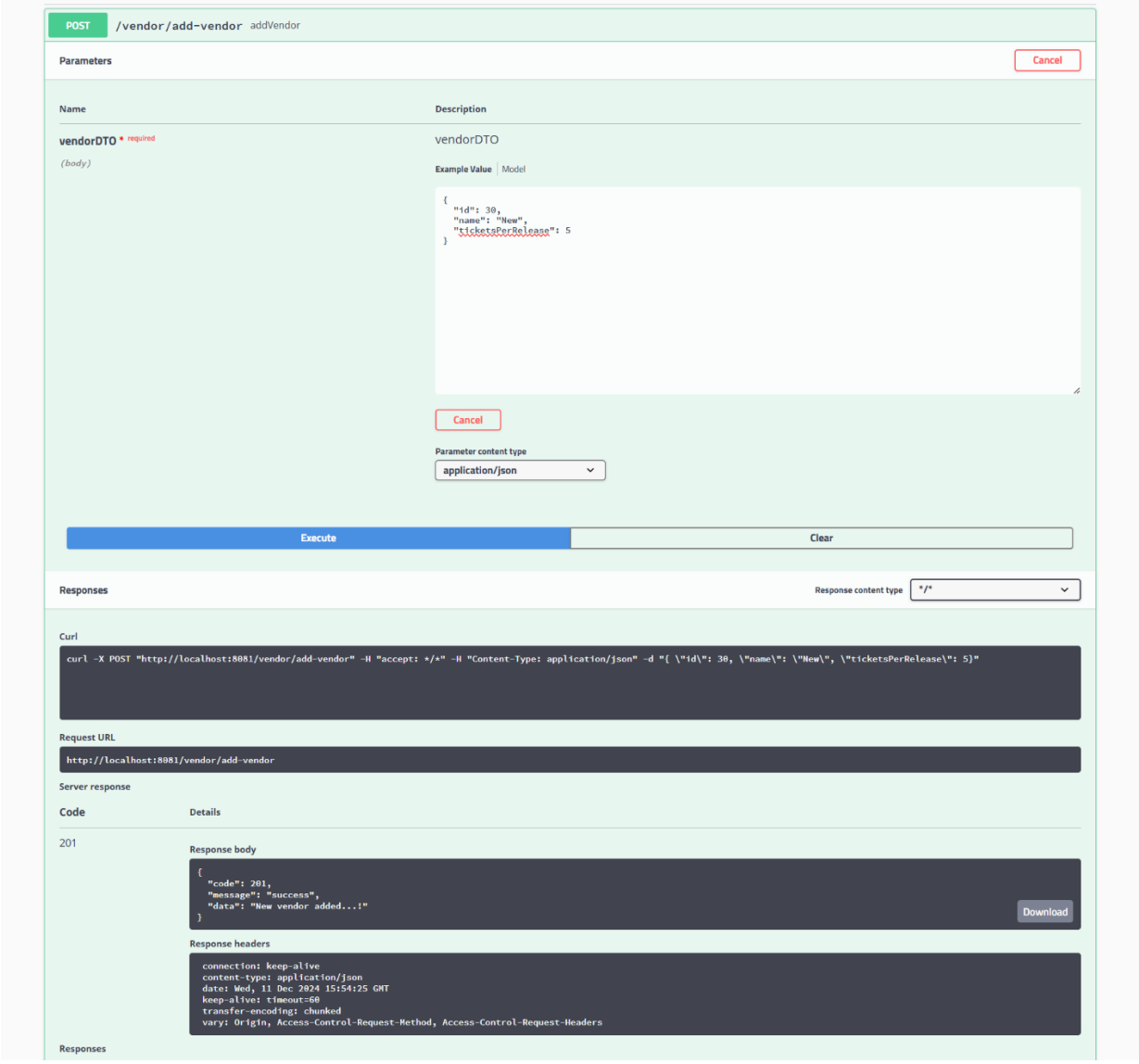
Ticket functions,

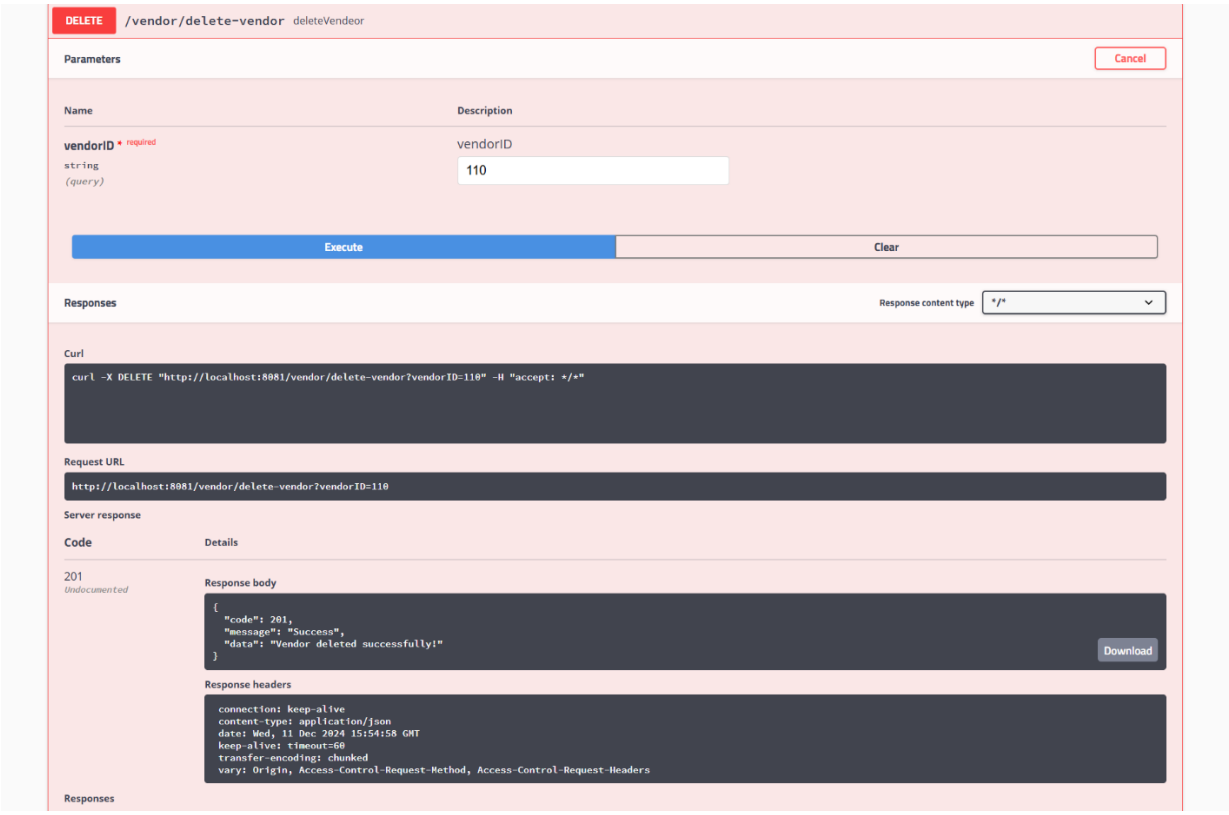


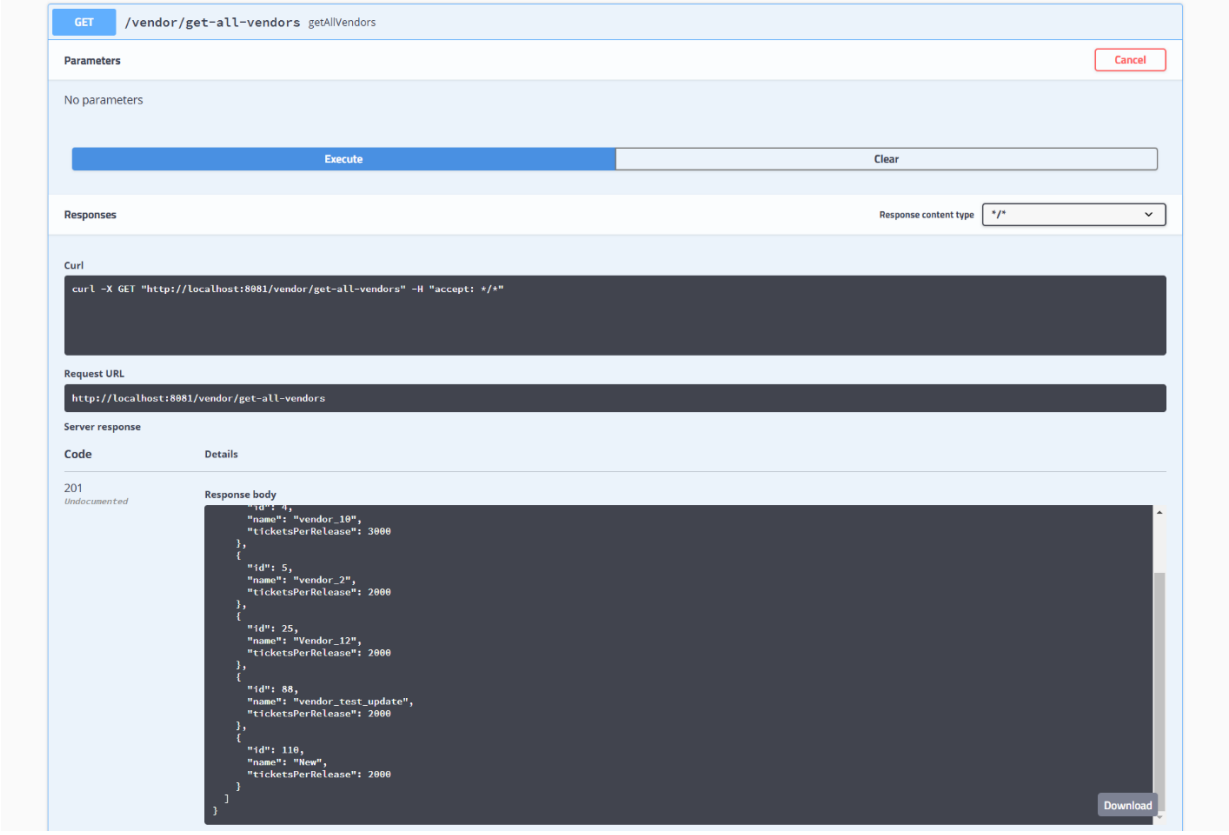

Curl
curl -X GET "http://localhost:8081/ticket/get-available-tickets" -H "accept: */*"
Request URL
http://localhost:8081/ticket/get-available-tickets
Server response
Code
Details
201
Undocumented
Response body
{
"code": 201,
"message": "Success",
"data": [
{
"id": 6,
"name": null,
"price": 10,
"sold": false,
"vendorID": null,
"ticketType": "EXHIBITION"
},
{
"id": 7,
"name": null,
"price": 10,
"sold": false,
"vendorID": null,
"ticketType": "EXHIBITION"
},
{
"id": 8,
"name": null,
"price": 10,
"sold": false,
Download
Response headers
connection: keep-alive
content-type: application/json
date: Wed, 11 Dec 2024 15:47:14 GMT
keep-alive: timeout=60
transfer-encoding: chunked
vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers
Responses

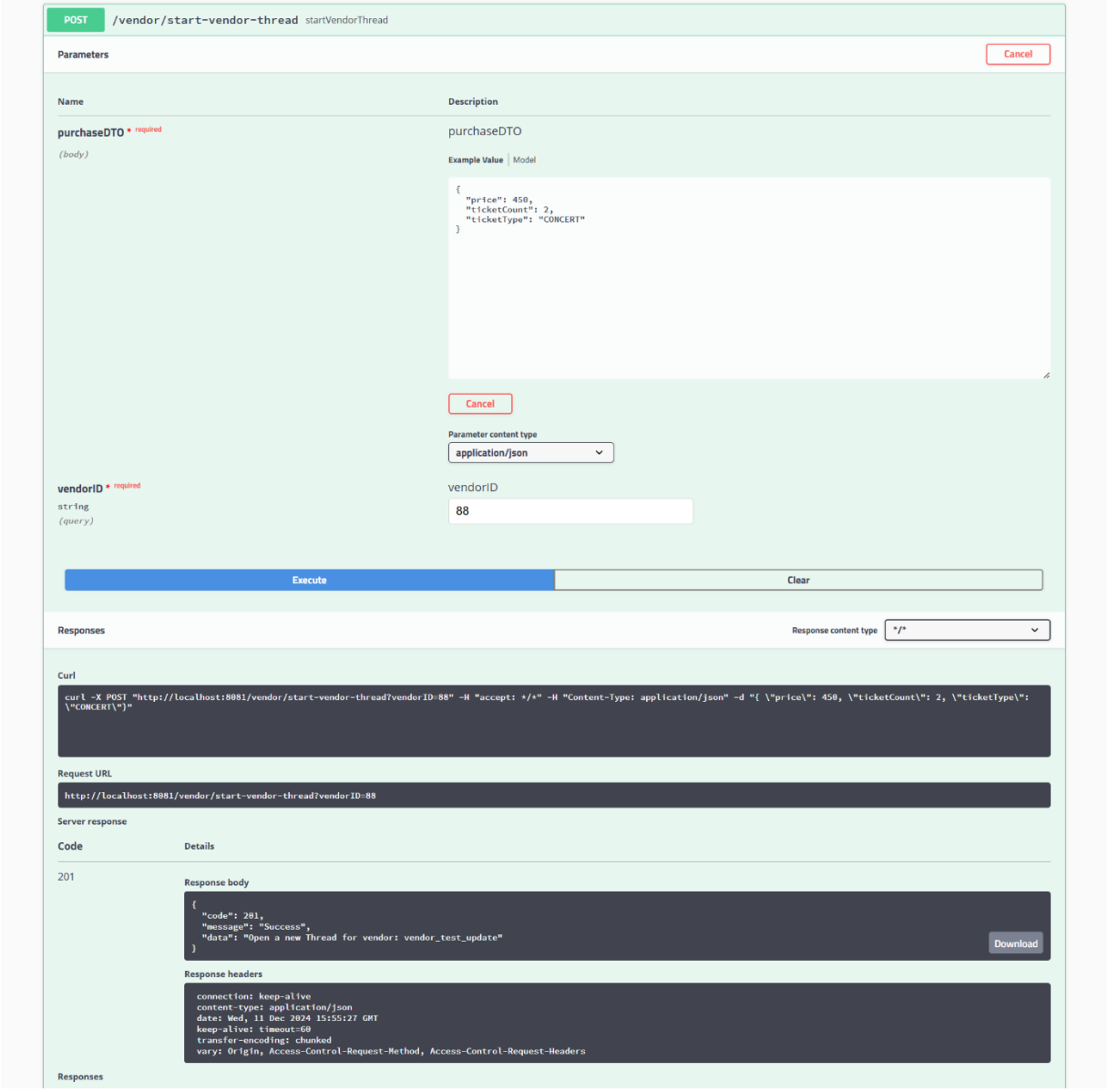

POST
/ticket/purchase-ticket
saveTicket
Parameters
Cancel
Name
Description
customerId * required
string
(query)
customerId
108
ticketId * required
string
(query)
ticketId
106
Execute
Clear
Responses
Response content type
/
Curl
curl -X POST "http://localhost:8081/ticket/purchase-ticket?customerId=108&ticketId=106" -H "accept: */*"
Request URL
http://localhost:8081/ticket/purchase-ticket?customerId=108&ticketId=106
Server response
Code
Details
500
Undocumented
Error:
Response body
{
"timestamp": "2024-12-11T15:47:59.856+00:00",
"status": 500,
"error": "Internal Server Error",
"path": "/ticket/purchase-ticket"
}
Download
Response headers
connection: close
content-type: application/json
date: Wed, 11 Dec 2024 15:47:59 GMT
transfer-encoding: chunked
vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers
Responses

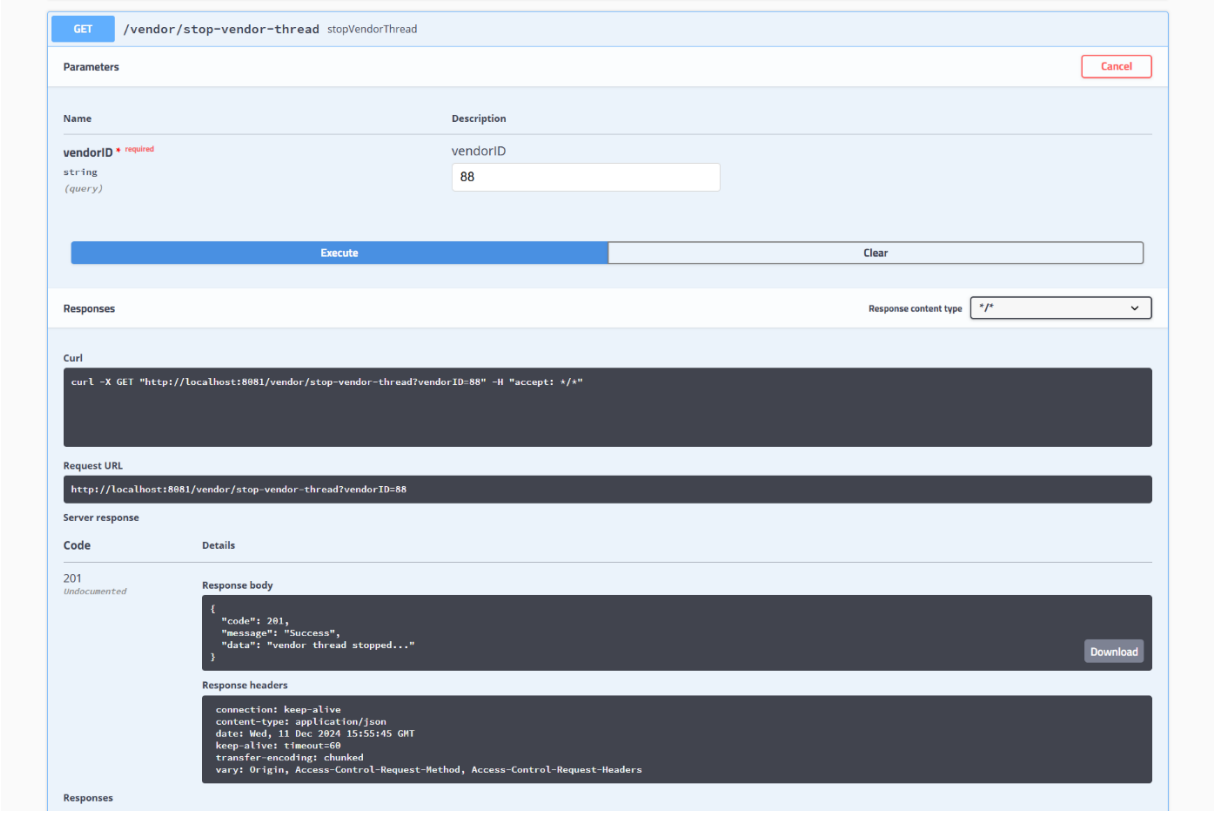
Vendor functions,

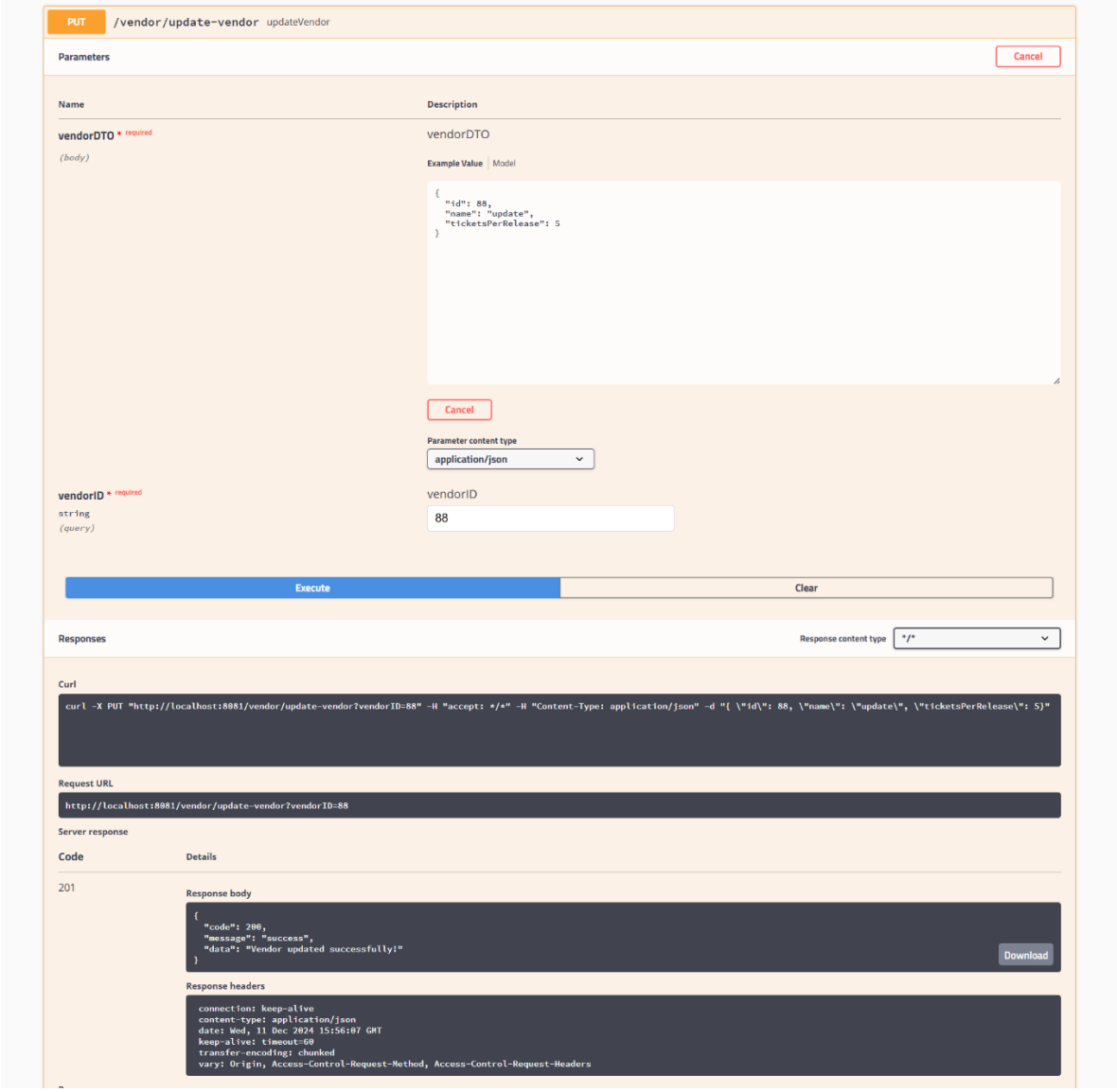



DELETE
/vendor/delete-vendor deleteVendeor
Parameters
Cancel
Name
Description
vendorID * required
string
(query)
vendorID
110
Execute
Clear
Responses
Response content type
/
Curl
curl -X DELETE "http://localhost:8081/vendor/delete-vendor?vendorID=110" -H "accept: */*"
Request URL
http://localhost:8081/vendor/delete-vendor?vendorID=110
Server response
Code
Details
201
Undocumented
Response body
{
"code": 201,
"message": "Success",
"data": "Vendor deleted successfully!"
}
Download
Response headers
connection: keep-alive
content-type: application/json
date: Wed, 11 Dec 2024 15:54:58 GMT
keep-alive: timeout=60
transfer-encoding: chunked
vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers
Responses


GET
/vendor/get-all-vendors getAllVendors
Parameters
Cancel
No parameters
Execute
Clear
Responses
Response content type
/
Curl
curl -X GET "http://localhost:8081/vendor/get-all-vendors" -H "accept: */*"
Request URL
http://localhost:8081/vendor/get-all-vendors
Server response
Code
Details
201
Undocumented
Response body
"name": "vendor_10",
"ticketsPerRelease": 3000
},
{
"id": 5,
"name": "vendor_2",
"ticketsPerRelease": 2000
},
{
"id": 25,
"name": "Vendor_12",
"ticketsPerRelease": 2000
},
{
"id": 88,
"name": "vendor_test_update",
"ticketsPerRelease": 2000
},
{
"id": 110,
"name": "New",
"ticketsPerRelease": 2000
}
]
}
Download

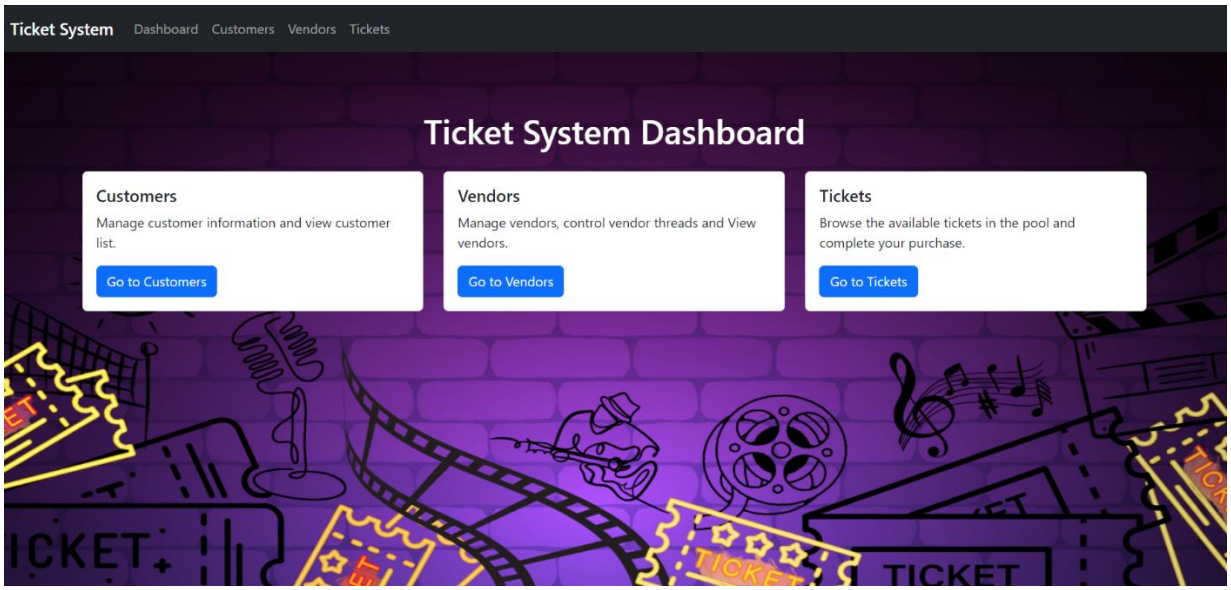

POST
/vendor/start-vendor-thread
startVendorThread
Parameters
Cancel
Name
Description
purchaseDTO * required
(body)
purchaseDTO
Example Value | Model
{
"price": 450,
"ticketCount": 2,
"ticketType": "CONCERT"
}
Cancel
Parameter content type
application/json
vendorID * required
string
(query)
vendorID
88
Execute
Clear
Responses
Response content type
/
Curl
curl -X POST "http://localhost:8081/vendor/start-vendor-thread?vendorID=88" -H "accept: */*" -H "Content-Type: application/json" -d "{ \"price\": 450, \"ticketCount\": 2, \"ticketType\": \"CONCERT\"}"
Request URL
http://localhost:8081/vendor/start-vendor-thread?vendorID=88
Server response
Code
Details
201
Response body
{
"code": 201,
"message": "Success",
"data": "Open a new Thread for vendor: vendor_test_update"
}
Download
Response headers
connection: keep-alive
content-type: application/json
date: Wed, 11 Dec 2024 15:55:27 GMT
keep-alive: timeout=60
transfer-encoding: chunked
vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers
Responses


GET
/vendor/stop-vendor-thread
stopVendorThread
Parameters
Cancel
Name
Description
vendorID
required
string
(query)
vendorID
88
Execute
Clear
Responses
Response content type
/
Curl
curl -X GET "http://localhost:8081/vendor/stop-vendor-thread?vendorID=88" -H "accept: */*"
Request URL
http://localhost:8081/vendor/stop-vendor-thread?vendorID=88
Server response
Code
Details
201
Undocumented
Response body
{
"code": 201,
"message": "Success",
"data": "vendor thread stopped..."
}
Download
Response headers
connection: keep-alive
content-type: application/json
date: Wed, 11 Dec 2024 15:55:45 GMT
keep-alive: timeout=60
transfer-encoding: chunked
vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers
Responses

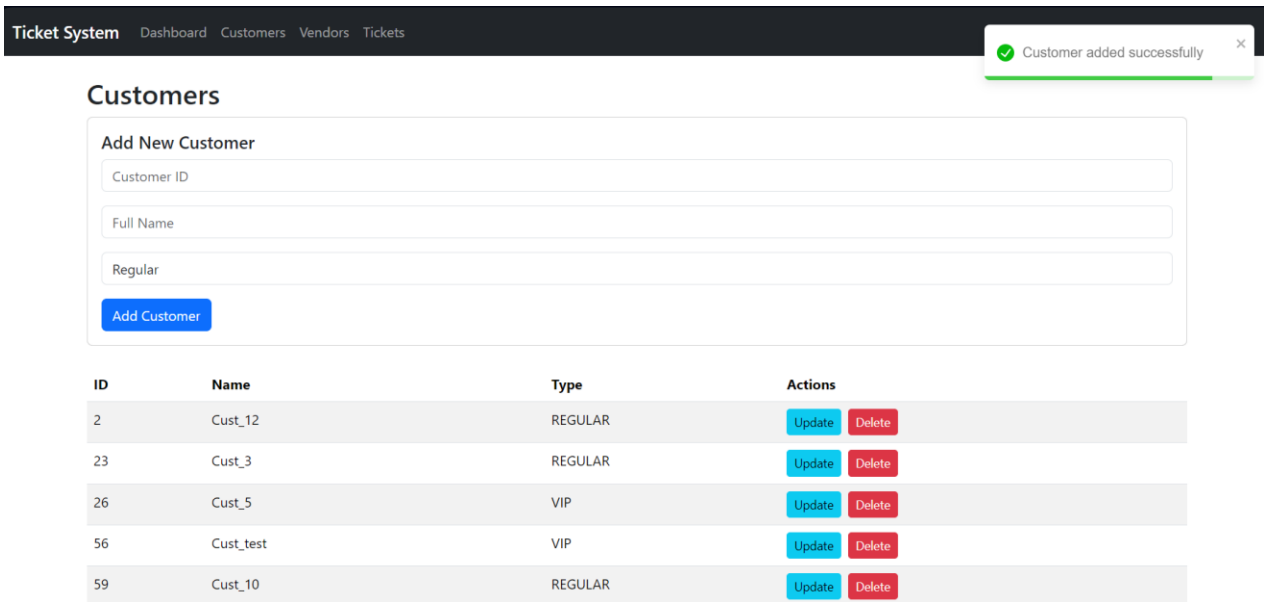

PUT
/vendor/update-vendor
updateVendor
Parameters
Cancel
Name
Description
vendorDTO * required
(body)
vendorDTO
Example Value
Model
{
"id": 88,
"name": "update",
"ticketsPerRelease": 5
}
Cancel
Parameter content type
application/json
vendorID * required
string
(query)
vendorID
88
Execute
Clear
Responses
Response content type
/
Curl
curl -X PUT "http://localhost:8081/vendor/update-vendor?vendorID=88" -H "accept: */*" -H "Content-Type: application/json" -d "{ \"id\": 88, \"name\": \"update\", \"ticketsPerRelease\": 5}"
Request URL
http://localhost:8081/vendor/update-vendor?vendorID=88
Server response
Code
Details
201
Response body
{
"code": 200,
"message": "success",
"data": "Vendor updated successfully!"
}
Download
Response headers
connection: keep-alive
content-type: application/json
date: Wed, 11 Dec 2024 15:56:07 GMT
keep-alive: timeout=60
transfer-encoding: chunked
vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers

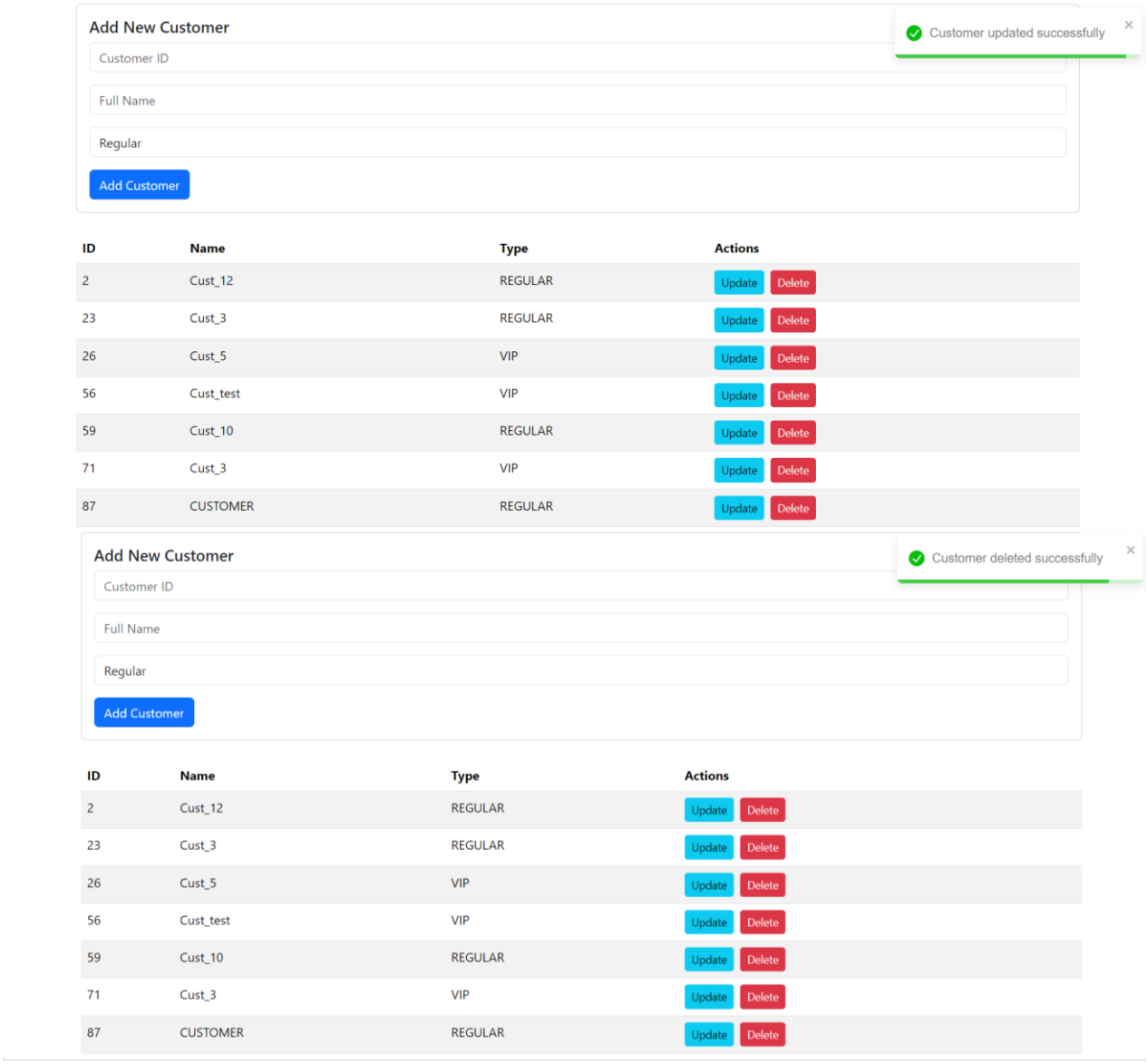
Test cases proof for the Frontend

Dashboard,

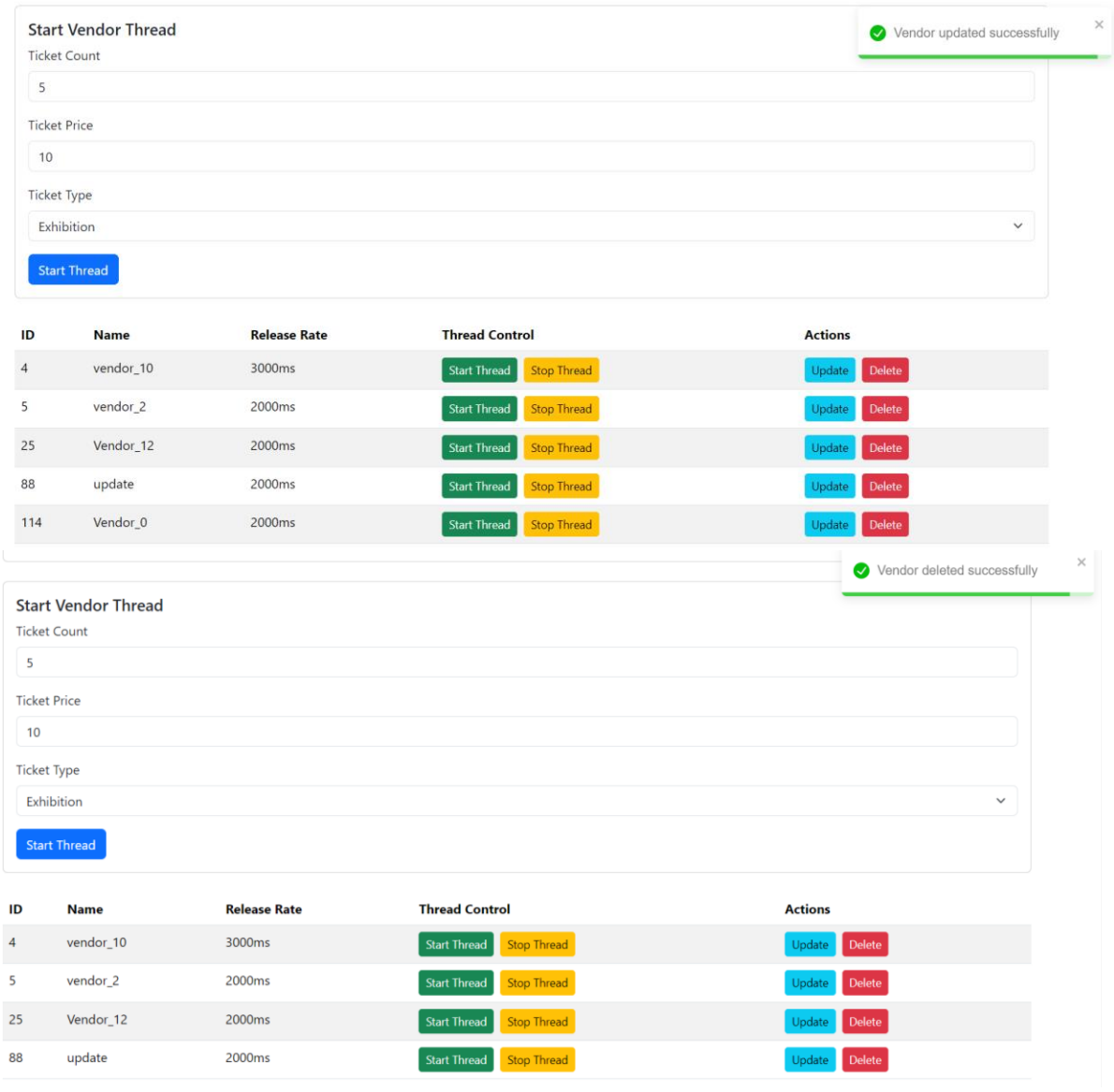


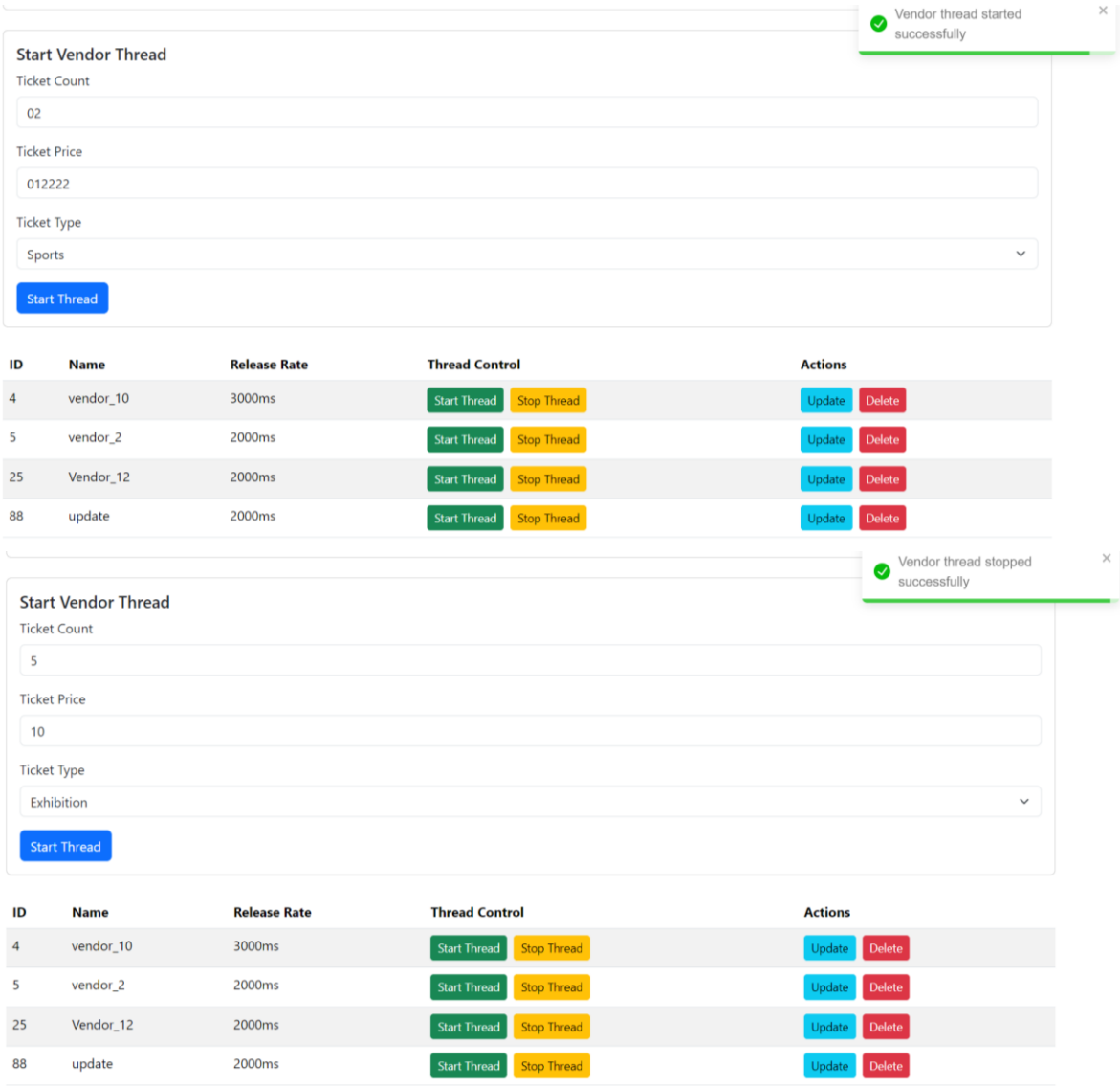
Customer features,



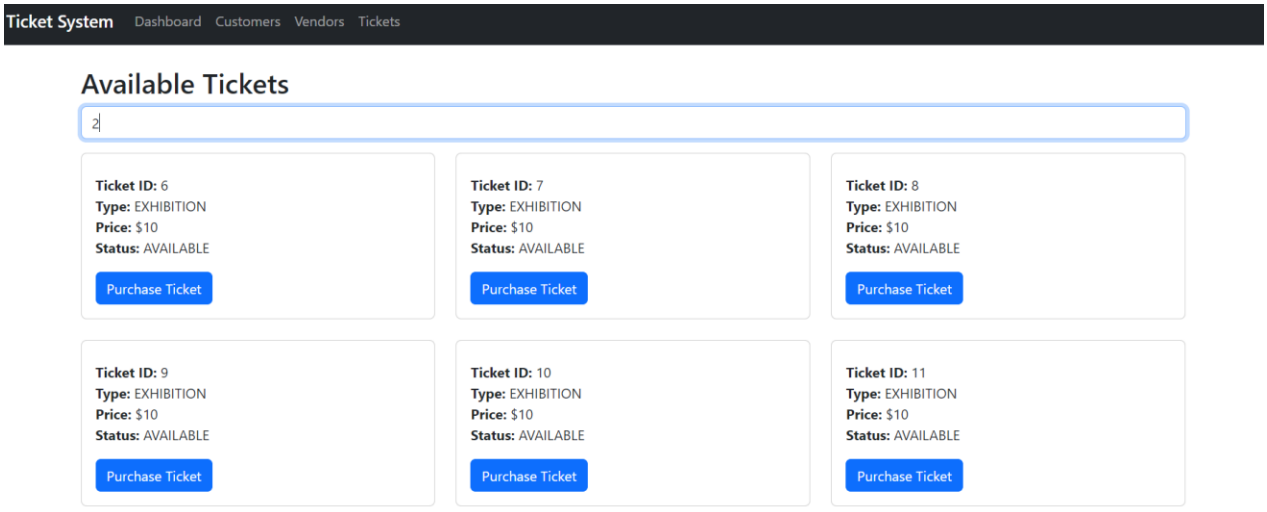

Add New Customer
Customer ID
Full Name
Regular
Add Customer
Customer updated successfully
ID
Name
Type
Actions
2
Cust_12
REGULAR
Update
Delete
23
Cust_3
REGULAR
Update
Delete
26
Cust_5
VIP
Update
Delete
56
Cust_test
VIP
Update
Delete
59
Cust_10
REGULAR
Update
Delete
71
Cust_3
VIP
Update
Delete
87
CUSTOMER
REGULAR
Update
Delete
Add New Customer
Customer ID
Full Name
Regular
Add Customer
Customer deleted successfully
ID
Name
Type
Actions
2
Cust_12
REGULAR
Update
Delete
23
Cust_3
REGULAR
Update
Delete
26
Cust_5
VIP
Update
Delete
56
Cust_test
VIP
Update
Delete
59
Cust_10
REGULAR
Update
Delete
71
Cust_3
VIP
Update
Delete
87
CUSTOMER
REGULAR
Update
Delete

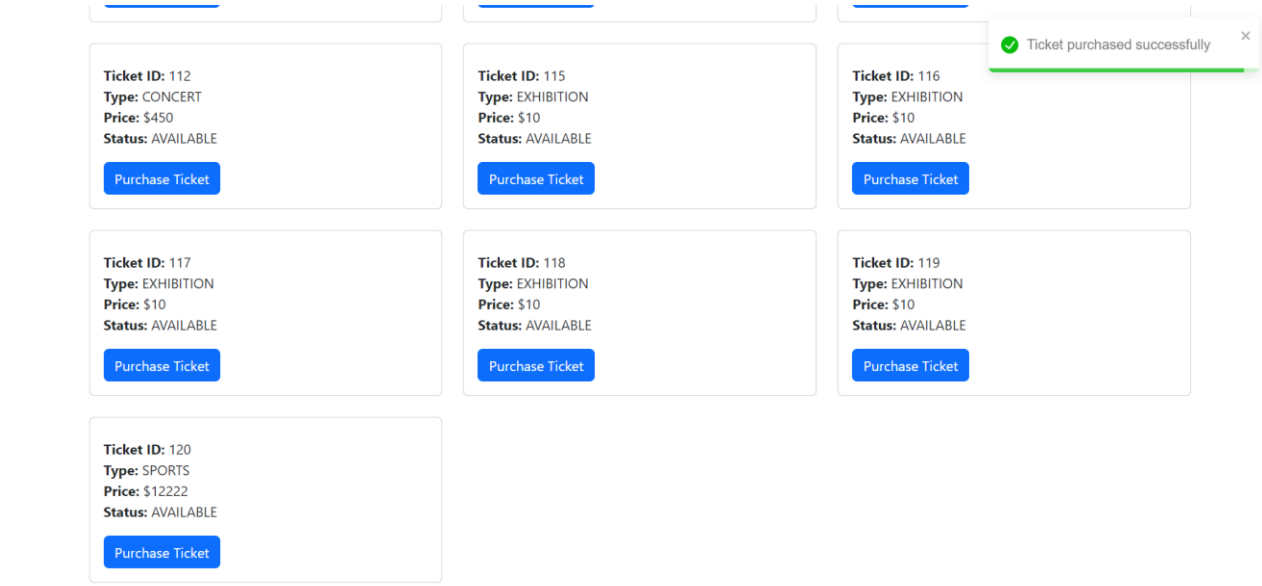
Vendor features,





Ticket displayer,





LinkedIn certificates: -

https://www.linkedin.com/learning/certificates/679eefc3bf5929b61487d9121e6faba8614e9514af15bf3e95e97eba490ba29e?trk=share_certificate

GitHub Link: -

https://github.com/danu2003-cy/RealTimeTicketSystem