UNIVERSITY OF WESTMINSTER

INFORMATICS INSTITUTE OF TECHNOLOGY

Informatic Institute of Technology

Object Oriented Programming

5COSC019C

Coursework: Report

Dulith Senhas Mayakaduwa

UoW No: W2052084

IIT No: 20221107

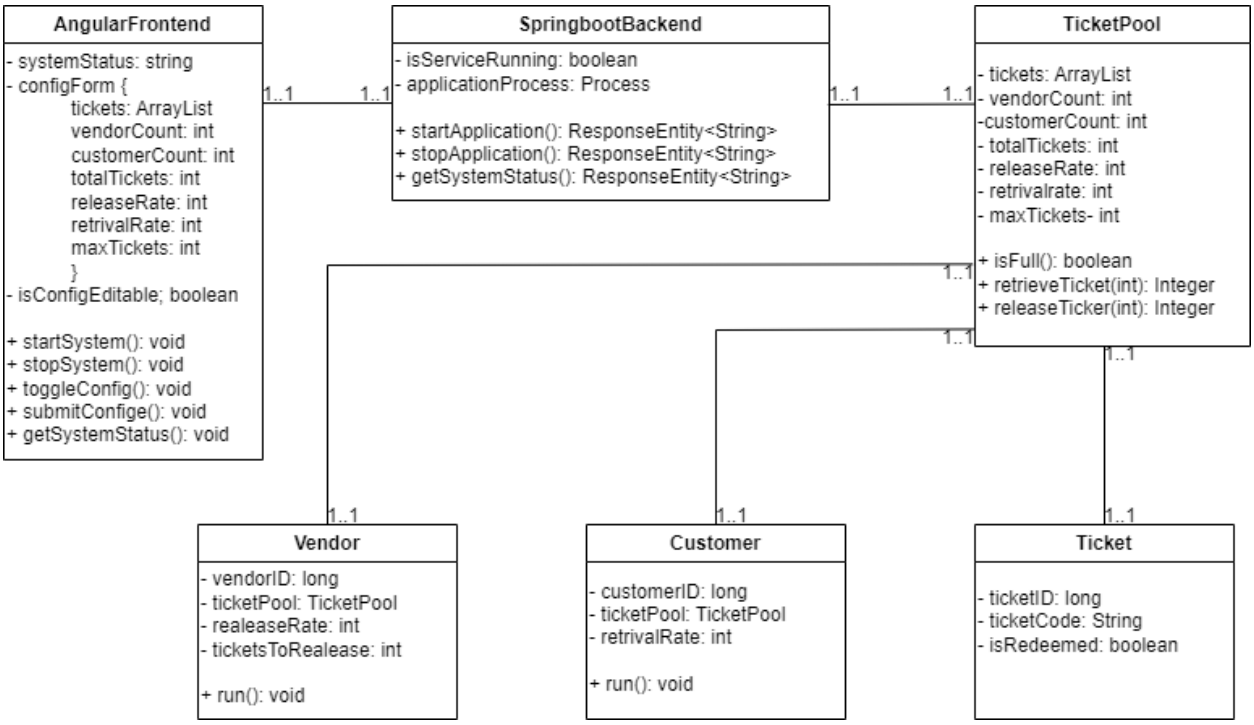
Tutorial Group: L5 CS-G16

Introduction

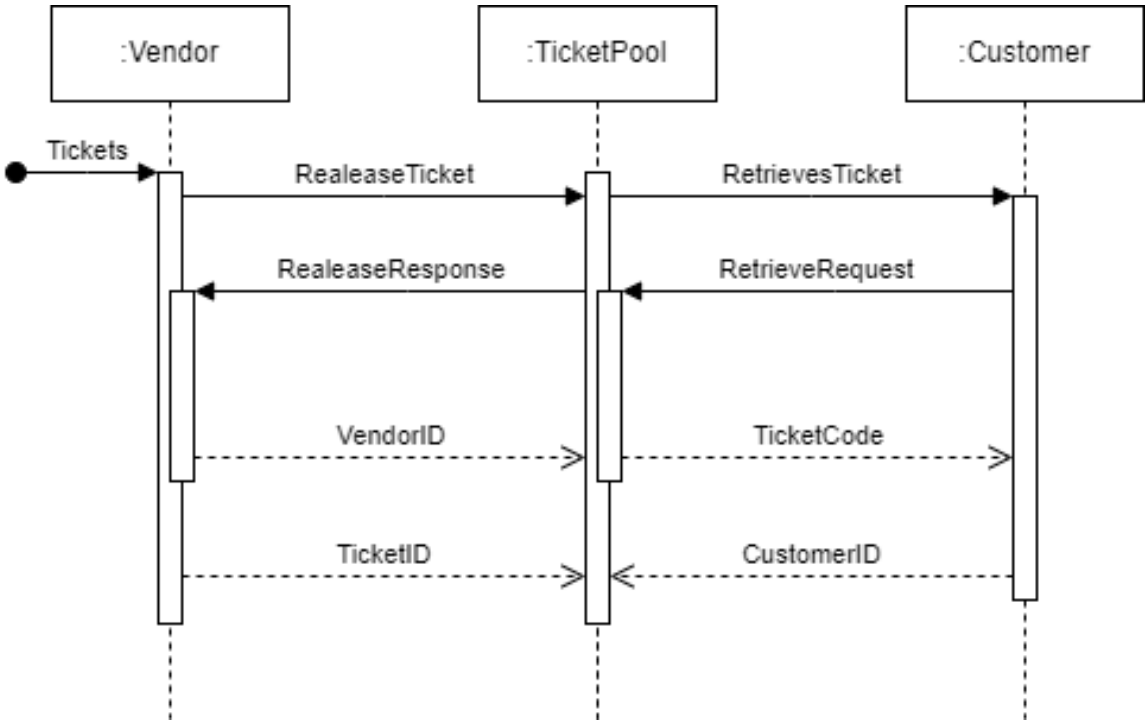
This report provides Diagrams for this project and detailed explanation of test cases to for each component of the Real-Time Event Ticketing System as per the coursework specification. The focus is on utilizing Object-Oriented Programming (OOP) principles and the Producer-Consumer pattern to simulate a dynamic ticketing environment.

Diagrams

1. Class Diagram



2. Sequence Diagram



Testing Report

1. Test Cases

Test Case ID	Description	Steps	Expected Results	Results
1.1	Fetch Initial System Status	1. Start the Angular app. 2. Verify displayed status. 3. Compare with /api/system/status response.	The displayed status matches the backend response.	Passed
1.2	Update System Status (Start)	1. Click **Start**. 2. Verify status updates to Running. 3. Verify /api/system/start request succeeds.	Status updates to Running, and the backend confirms success.	Passed
1.3	Update System Status (Stop)	1. Click **Stop**. 2. Verify status updates to Stopped. 3. Verify /api/system/stop request succeeds.	Status updates to Stopped, and the backend confirms success.	Passed
1.4	Handle Backend Failure for Status	1. Simulate failure in /api/system/status, /api/system/start, or /api/system/stop. 2. Observe error messages or fallback behavior.	Error message displayed, and the system defaults to Stopped.	Passed
2.1	Form Validation	1. Leave one or more fields empty. 2. Click **Save**. 3. Check for validation alerts.	Save button remains disabled until all fields are filled.	Passed
2.2	Submit Configuration	1. Fill all fields. 2. Click **Save**. 3. Verify /api/ticketpools request payload matches form. 4. Check backend for saved data.	Configuration is saved in the backend successfully.	Passed
2.3	Configuration Edge Cases	Test inputs: - Vendor count = 0 - Customer count = 0 - Negative release/retrieval rates - High total tickets/max capacity - Non-numeric values. Check validation.	The form does not accept invalid inputs. Validation errors are displayed.	Passed
3.1	Max Capacity Edge Case	1. Configure maxTickets. 2. Start vendors releasing tickets. 3. Observe when capacity is reached.	Vendors stop releasing tickets when capacity is full.	Passed

3.2	Multiple Customers	1. Configure multiple customers. 2. Start the system. 3. Observe ticket retrieval behavior.	Each ticket is retrieved once, and no synchronization issues occur.	Passed
3.3	Multiple Vendors	1. Configure multiple vendors. 2. Start the system. 3. Observe ticket release behavior.	Each ticket is released uniquely, with no duplication.	Passed
3.4	Empty Ticket Pool	1. Start customers without vendors. 2. Observe customer behavior (e.g., waiting or failing).	Customers handle empty pools gracefully without exceptions.	Passed
4.1	High Load of Vendors and Customers	1. Configure 50+ vendors and 100+ customers. 2. Start the system. 3. Monitor performance.	The system remains stable under high load with no synchronization issues.	Passed
4.2	Restart System with Active Processes	1. Start the system with active vendors and customers. 2. Stop and restart the system.	The system resets and restarts without errors or hanging.	Passed
5.1	Invalid API Responses	1. Simulate invalid responses from /api/system/status. 2. Observe frontend behavior.	Frontend handles errors gracefully and displays a meaningful message.	Passed
5.2	Backend Timeout	1. Simulate delayed responses. 2. Try starting/stopping the system or saving configuration. 3. Observe behavior.	Frontend displays a timeout error and does not hang.	Passed
6.1	Full System Workflow	1. Fetch system status. 2. Start system. 3. Configure system. 4. Run vendors/customers. 5. Stop the system.	The workflow completes without errors, and the system behaves as expected.	Passed
6.2	Cross-Origin Requests	1. Test Angular's HTTP requests to the backend. 2. Observe if CORS-related errors occur.	Requests succeed without any CORS-related issues.	Passed