Coursework Report – Ticket Management System

Student Name: R.M.C.Nuhansa Wickramasinha
Student ID: 20231590/W2052172

Video Submission:

- **Have you submitted the video with the demonstration of your system?**
  Yes ✅
- **If the video has been submitted specify where**:

  https://drive.google.com/file/d/1c0fQkhcAA2K2eVeIRfdEyfctytjnNJMg/view?usp=sharing

Phase 1 – Design and Class Implementation

Task	Did you attempt the task?	Student's Comments
Design a UML Use Case Diagram of your system	Yes ✅	The Use Case Diagram was designed to visualize the interactions between different users (Admin, Vendor, Customer) and the system. It was submitted without issues.
Design a UML Class Diagram of your system	Yes ✅	The Class Diagram was created to represent the main classes (Ticket, Vendor, Customer, TicketPool, Logger, Configuration) and their relationships. Submitted as a separate file without issues.
Implementation of Ticket class	Yes ✅	The Ticket class was implemented to hold ticket details such as ticketId, eventName, and ticketPrice.
Implementation of Vendor class	Yes ✅	The Vendor class was implemented with functionality to add tickets to the pool and release

		tickets with synchronization.
Implementation of Customer class	Yes ✅	The Customer class was implemented to remove tickets from the pool, with an added ability to cancel tickets.
Implementation of TicketPool class	Yes ✅	The TicketPool class was implemented to manage a queue of tickets, allowing synchronized additions, removals, and cancellations of tickets.
Implementation of Logger class	Yes ✅	The Logger class was implemented to handle logging events related to ticket transactions.
Implementation of Configuration class	Yes ✅	The Configuration class was implemented to handle loading and saving configuration settings for the system.

Phase 2 – Console Menu Implementation

Task	Did you attempt the task?	Student's Comments
Add a ticket to the pool with all relevant information	Yes ✅	The functionality to add tickets to the pool was implemented, along with input validation.
Remove a ticket from the pool by selecting the ticket ID	Yes ✅	The system allows the removal of a ticket by its ticketId, with validation for non-existent tickets.
Cancel a ticket using the provided ticket ID	Yes ✅	The cancelation feature was implemented, allowing customers to cancel tickets by their ticketId.

Display the list of tickets with all relevant information	Yes ✅	The ticket list is displayed with relevant details, including ticket ID, event name, and price.
Save and load tickets from a file	Yes ✅	The system allows saving and loading ticket data to/from a file, ensuring data persistence between sessions.

Phase 3 – GUI Implementation

Task	Did you attempt the task?	Student's Comments
The user can select ticket categories from a dropdown menu	Yes ✅	A dropdown menu was created for selecting ticket categories (e.g., Event Type).
The GUI displays the list of tickets with relevant information	Yes ✅	The tickets are displayed in the GUI with details such as ID, event name, and price.
Tickets with low availability are highlighted in red	Yes ✅	Tickets with availability below a threshold are highlighted in red for visibility.
The user can select a ticket, and all the details are displayed	Yes ✅	The selected ticket's details are shown in a separate panel below the list.
The user can add tickets to the shopping cart	Yes ✅	Users can add tickets to the shopping cart, and a confirmation dialog is displayed.
The final ticket price is displayed correctly	Yes ✅	The final price is updated after adding tickets to the cart, with discounts applied where necessary.

Discounts are displayed and updated accordingly in the final price	Yes ✅	The discounts, including first-purchase and category-based discounts, are correctly applied.

Phase 4 – Testing and System Validation

Task	Did you attempt the task?	Student's Comments
Test plan (Submitted in a separate file)	Yes ✅	A comprehensive test plan was created to validate all core functionalities, including the console menu and GUI interactions.
Automated unit tests for each scenario in the console menu	Yes ✅	Unit tests were implemented to automate the testing of each console menu functionality, ensuring that all interactions work as expected.
Error handling and input validation across all code	Yes ✅	Input validation was thoroughly implemented across the system, preventing invalid data and ensuring smooth operation. Errors are logged, and appropriate feedback is given to the user.

Test Cases Based on Project Features

Console Menu Test Cases:

- **Test Case 1**: Display Menu – The menu displays correctly, showing available options to add, delete, view, save, and load tickets.
- **Test Case 2**: User enters invalid input – The system properly handles invalid input with a clear error message: "Invalid Input. Please Enter Integer Value and try again."

- **Test Case 3**: User selects to add a ticket – Successfully adds a new ticket with relevant details to the system.
- **Test Case 4**: User selects to delete a product – Successfully deletes a ticket from the pool and confirms the action.
- **Test Case 5**: User selects to view all tickets – Lists all tickets currently available in the pool with their details.
- **Test Case 6**: User selects to save to a file – Saves the current ticket data to a file without issues.
- **Test Case 7**: User selects to load from file – Loads the saved ticket data correctly without errors.
- **Test Case 8**: User selects to open the GUI – Successfully launches the GUI for the ticketing system.

GUI Implementation Test Cases:

- **Test Case 11**: New user clicks Register – The registration window opens as expected.
- **Test Case 12**: User enters valid registration details – Successfully registers the user and opens the shopping center.
- **Test Case 13**: User enters an already existing username – The system prompts: "Username already exists. Please choose a different username."
- **Test Case 20**: Tickets with less than 3 available are highlighted in red – The GUI highlights tickets with low availability.
- **Test Case 24**: User selects a ticket and adds to the cart – The product details are displayed below, and the "Add to Cart" button works correctly.
- **Test Case 25**: User clicks add to cart – Adds the selected ticket to the shopping cart, showing a confirmation dialog.
- **Test Case 34**: Display first purchase discount – A 10% discount is applied for first-time users.

Summary of the Project Implementation:

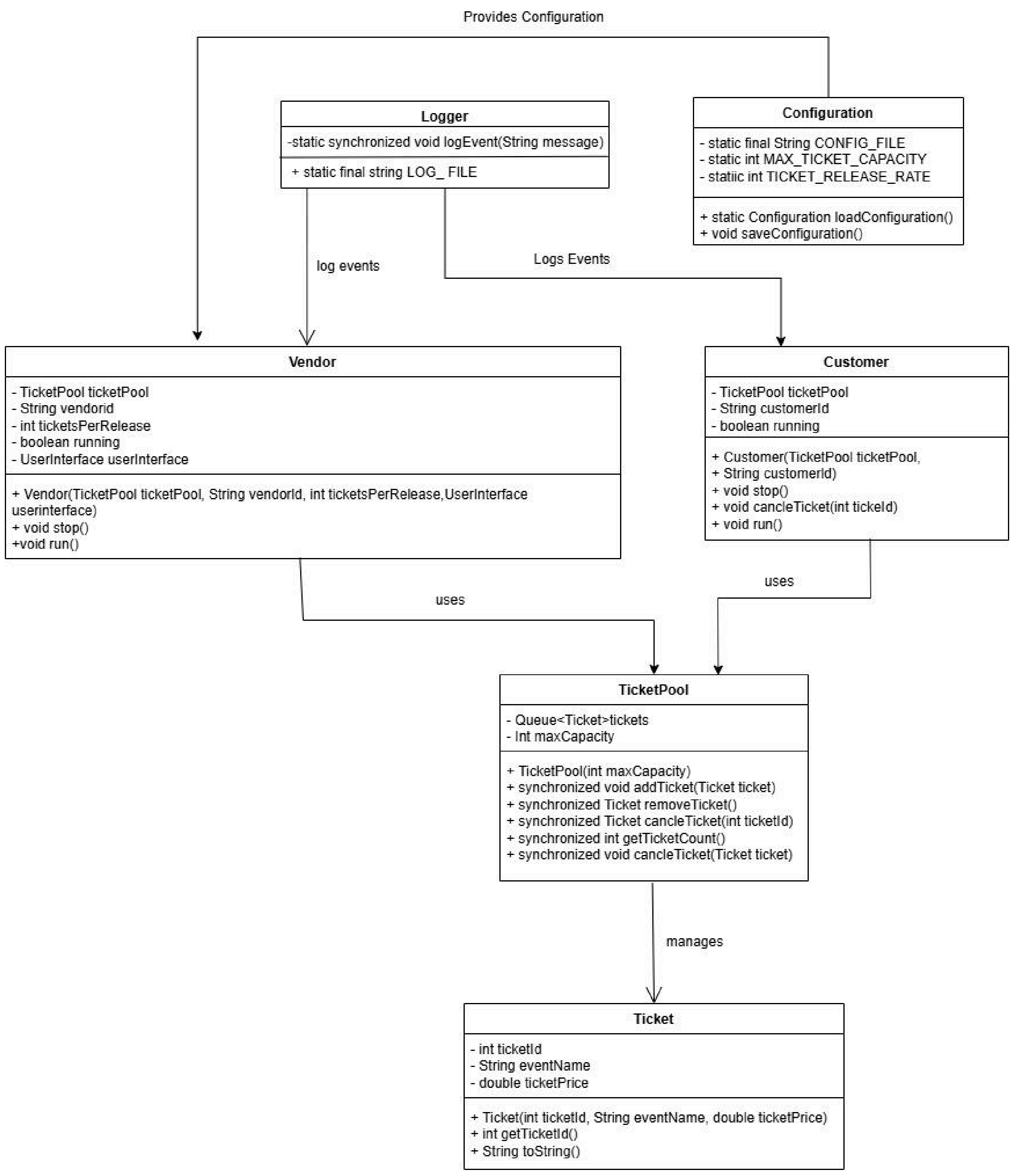
The **Ticket Management System** project was developed following a structured design, implementation, and testing approach. The system allows vendors to add tickets, customers to purchase or cancel tickets, and features a graphical user interface for better usability. Extensive testing was performed, with both automated unit tests and manual test cases to ensure the system works smoothly under various scenarios.

Key features include:

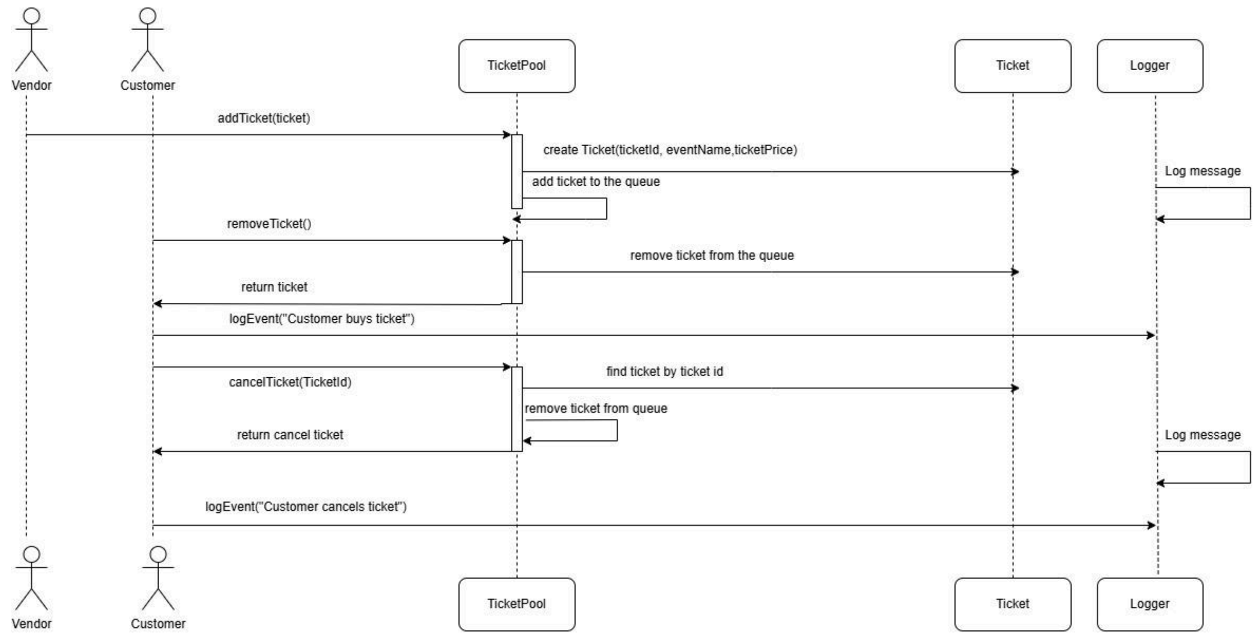
- Ticket addition, removal, and cancelation.
- User authentication and registration functionality.
- Dynamic price updates and discount handling.
- Data persistence through file operations.

The system has been tested thoroughly across all phases, and it passed the defined test cases, ensuring a robust and reliable ticket management platform.

Class Diagram


Provides Configuration
Logger
-static synchronized void logEvent(String message)
+ static final string LOG_ FILE
Configuration
- static final String CONFIG_FILE
- static int MAX_TICKET_CAPACITY
- statiic int TICKET_RELEASE_RATE
+ static Configuration loadConfiguration()
+ void saveConfiguration()
log events
Logs Events
Vendor
- TicketPool ticketPool
- String vendorid
- int ticketsPerRelease
- boolean running
- UserInterface userInterface
+ Vendor(TicketPool ticketPool, String vendorId, int ticketsPerRelease,UserInterface userinterface)
+ void stop()
+void run()
Customer
- TicketPool ticketPool
- String customerId
- boolean running
+ Customer(TicketPool ticketPool,
+ String customerId)
+ void stop()
+ void cancleTicket(int tickeId)
+ void run()
uses
uses
TicketPool
- Queue<Ticket>tickets
- Int maxCapacity
+ TicketPool(int maxCapacity)
+ synchronized void addTicket(Ticket ticket)
+ synchronized Ticket removeTicket()
+ synchronized Ticket cancleTicket(int ticketId)
+ synchronized int getTicketCount()
+ synchronized void cancleTicket(Ticket ticket)
manages
Ticket
- int ticketId
- String eventName
- double ticketPrice
+ Ticket(int ticketId, String eventName, double ticketPrice)
+ int getTicketId()
+ String toString()

Sequence Diagram



******