Report: Ticket Management System

**1. Introduction**

The Ticket Management System is designed to streamline the process of handling customer support tickets, agent assignments, and logging ticket resolutions. The system is structured using several core data structures, such as Dynamic Arrays, Stacks, Queues, and Linked Lists, to efficiently manage agents, tickets, and logs. The system ensures that tickets are assigned to agents based on their availability and priorities and provides functionality to track the resolution of tickets and maintain historical logs.

**2. Components of the System**

The system is divided into several modules:

1. **Ticket Class and List Management**
2. **Agent Management System**
3. **Dynamic Arrays**
4. **Queues**
5. **Stacks**
6. **Ticket System Class**

**3. Ticket Class and List Management**

The Ticket class represents a support ticket in the system. It includes fields for:

* ticketID: Unique identifier for the ticket.
* customerName: Name of the customer who submitted the ticket.
* priority: Priority of the ticket, determining its urgency.
* description: A brief description of the issue.
* creationTime: Timestamp when the ticket is created.
* status: Current status of the ticket (open/closed).
* closeTime: Time at which the ticket was resolved.

The ticketlist class is a linked list used to manage multiple tickets. It supports the following functionalities:

* Inserting tickets at the end, start, or in specific positions.
* Deleting tickets from the head or end.
* Searching tickets by ticketID.
* Sorting tickets by priority, creation time, or customer name.

**4. Agent Management System**

The Agent class manages agents who handle support tickets. It contains the following fields:

* agentID: Unique identifier for the agent.
* name: The agent's name.
* assignedTickets: A dynamic array storing the ticket IDs assigned to the agent.
* availability: Boolean flag to indicate whether the agent is available to take new tickets.
* status: Represents whether the agent is available or unavailable.

The AgentManager class manages a collection of agents. Key operations include:

* Adding new agents.
* Assigning tickets to the most suitable available agent.
* Displaying agent details and their current status.
* Handling ticket assignments based on agent availability.

**5. Dynamic Array**

The DynamicArray class provides a resizable array implementation to store agent ticket assignments and other data. Key features include:

* The ability to dynamically resize the array when capacity is reached.
* Accessing elements via the [] operator and checking the array size.
* Displaying the contents of the array.

**6. Queues**

The Queue class is used to manage pending tickets. Key operations provided by the class include:

* Enqueuing tickets into the queue.
* Dequeuing tickets for processing.
* Peeking the front ticket in the queue.
* Displaying all pending tickets.

**7. Stacks**

The Stack class is used to maintain a log of closed tickets. The stack works on a Last In, First Out (LIFO) basis, where the most recently closed ticket is the first to be accessed for viewing. Operations include:

* Pushing new logs into the stack.
* Popping logs from the stack for viewing.
* Displaying the entire log of closed tickets.

**8. Ticket System Class**

The TicketSystem class brings all components together and coordinates the overall ticketing process. It manages two main operations:

1. **Logging Closed Tickets**: Whenever a ticket is closed, a log entry is created and pushed to the resolution log stack.
2. **Adding Tickets to the Queue**: New tickets are added to the pending ticket queue, where they await assignment to available agents.

**9. Functionality Overview**

* **Agent Assignment**: The system assigns tickets to agents based on their availability. If an agent is available and has fewer than 5 tickets assigned, they are selected to handle the next ticket.
* **Ticket Prioritization**: Tickets can be prioritized, and the system supports sorting tickets based on priority or creation time, ensuring that the most urgent tickets are handled first.
* **Ticket History and Logs**: The resolution of each ticket is logged with timestamps, providing a history of actions taken on each ticket. These logs are stored in a stack for easy retrieval.
* **Efficient Handling of Tickets**: The system uses a queue to manage pending tickets and a dynamic array for handling assigned tickets, making the ticketing process more organized and efficient.

**10. Challenges and Considerations**

* **Agent Availability Management**: Ensuring that agents are only assigned tickets when they have the availability to take on more is a crucial feature. The system must dynamically update agent availability based on their current load.
* **Scalability**: As the number of tickets and agents grows, the system needs to be scalable to handle larger datasets efficiently. The use of dynamic arrays, linked lists, and queues helps ensure that the system can scale as needed.
* **Ticket Sorting**: Sorting tickets by various criteria such as priority and creation time ensures that urgent issues are addressed first. The system provides multiple sorting options for managing tickets.

**11. Conclusion**

The Ticket Management System provides an effective solution for handling customer support tickets. By integrating dynamic arrays, stacks, queues, and linked lists, the system ensures efficient management of tickets and agents, while also keeping a detailed log of all resolved tickets. The use of object-oriented principles makes the system modular and maintainable, allowing for easy updates and extensions.

This system is a practical implementation of various data structures to solve real-world problems in ticket management, and it can be further extended with features such as ticket reassignment, agent performance tracking, or integration with other support tools.