The Smart Queue Management System

The Smart Queue Management System is an innovative solution designed to streamline queue management in service-driven environments such as banks, hospitals, and customer support centers. Its primary goal is to enhance efficiency and fairness by dynamically prioritizing individuals based on factors like urgency, waiting time, and service type. This system addresses common problems in traditional queue management, including excessive delays, unfair prioritization, and challenges with complex service dependencies, by offering a modern, adaptable approach. Each service category, such as VIP, Regular, or Emergency, has its own separate queue, and the system effectively manages these distinct queues. Urgency refers to the immediate need for service, often indicating the severity of a situation, while service type categorizes the nature of the service required, which might also imply a certain level of importance or specialized handling.

The system features several standout components. The **Modular Priority Engine** calculates priority scores using a customizable weighted formula, allowing administrators to adjust the balance of urgency, wait time, and service needs. **Dynamic Priority Updates** ensure fairness by periodically recalibrating scores to prevent individuals from waiting excessively long. With **Multiple Queues with Merging**, the system manages distinct queues for various service categories and can merge or redirect them as operational needs dictate. The **Fairness Monitor** tracks wait times, automatically elevating the priority of those who exceed set thresholds. For analysis, **Simulation Mode** generates random queue scenarios to test performance, while **Advanced Reporting and Sorting** provides detailed historical insights. The **Admin Console** provides full configurability of system parameters, tying all features together.

Contents

1. Modular Priority Engine

Description: The modular priority engine is the foundational component responsible for calculating and managing the priority of each individual in the queue. Administrators can define custom weights for priority factors such as urgency, waiting time, and service type. These weights are applied to an individual's attributes to compute a comprehensive priority score, ensuring that the individual with the highest score is served first.

Example (User Scenario): A new individual arrives at the system.

- **User Input:**
  - Individual ID: 101
  - Urgency Level: 5 (scale 1-5, 5 being most urgent)
  - Service Type: 'emergency'
- **System-Managed Data:**
  - Waiting Time: 0 minutes (initial)
- **Defined Weights (from Admin Console):**
  - urgency_weight = 0.5
  - waiting_time_weight = 0.3
  - service_type_weight = 0.2
- **Service Type Score (from Admin Console):**
  - 'emergency' = 10
- **System Processing:**
  - Retrieve weights from a hash table.
  - Calculate priority score using the formula:
    Priority Score=(Urgency Level×urgency_weight)+(Waiting Time×waiting_time_weight)+(Service Type Score×service_type_weight)
    Priority Score=(5×0.5)+(0×0.3)+(10×0.2)
    Priority Score=2.5+0+2.0Priority Score=4.5
  - Insert individual 101 into the priority queue with score 4.5.
- **System Output:** "Individual ID 101 with priority score 4.5 has been added to the main queue."

2. Dynamic Priority Updates

Description: Building on the modular priority engine, this feature prevents indefinite waiting ("starvation") by periodically updating priority scores based on increased waiting time. Updated scores reposition individuals in the queue, ensuring fairness.

Example (User Scenario): Individual 102's priority is re-evaluated after waiting.

- **Initial State:**
  - Individual ID: 102
  - Urgency Level (initial): 3 (assumed for example)
  - Service Type (initial): 'regular' (assumed for example)
  - Service Type Score (initial): 5 (for 'regular', from Admin Console)
  - Waiting Time (initial): 10 minutes
  - Initial Score: 3.0 (given in source, but we can recalculate for clarity based on example values: (3×0.5)+(10×0.3)+(5×0.2)=1.5+3.0+1.0=5.5)

- **System Processing (after 10 more minutes):**
  - New Waiting Time: 20 minutes
  - Recalculate score using the same priority score formula with the updated waiting time: New Priority Score=(Urgency Level×urgency_weight)+(New Waiting Time×waiting_time_weight)+(Service Type Score×service_type_weight)New Priority Score=(3×0.5)+(20×0.3)+(5×0.2)New Priority Score=1.5+6.0+1.0New Priority Score=8.5
  - Update position in the max-heap.
- **System Output:** "Priority for Individual ID 102 updated to 8.5 due to increased waiting time."

3. Multiple Queues with Merging

Description: Extending the single-queue system, this feature manages multiple priority queues (e.g., VIP, Regular, Emergency) and supports merging or load balancing when operational needs arise, such as an empty queue.
Example (User Scenario): VIP queue empties while the Regular queue is long.

- **Initial State:**
  - VIP Queue: [ID=201 (P=8.0), ID=202 (P=7.5)]
  - Regular Queue: [ID=301 (P=6.0), ID=302 (P=5.8), ID=303 (P=5.5)]
- **System Processing:**
  - Serve ID=201 and ID=202; VIP queue empties.
  - Merge Regular queue individuals into VIP queue. The individuals from the Regular queue are inserted into the (now empty) VIP queue, maintaining their priorities.
- **System Output:** "VIP queue is now empty. Redirecting individuals from regular queue to VIP service counter."

4. Fairness Monitor

Description: This feature ensures no individual waits excessively by tracking waiting times and flagging those exceeding a threshold, triggering priority boosts. When an individual's waiting time surpasses the defined max_wait_time threshold, their priority is explicitly increased based on how much extra time they have waited. This dynamic boost ensures that individuals who have waited significantly longer beyond the threshold receive a proportionally higher priority.
Equation for Priority Boost (within Fairness Monitor):
When an individual's Current Waiting Time exceeds the max_wait_time threshold, a Fairness Boost is calculated and added to their Current Priority Score.
First, calculate the Extra Waiting Time:

$$\text{Extra Waiting Time} = \text{Current Waiting Time} - \text{max\_wait\_time}$$

(This value is 0 if Current Waiting Time ≤ max_wait_time)

Then, calculate the Fairness Boost:

$$\text{Fairness Boost} = \text{Extra Waiting Time} \times \text{boost\_multiplier}$$

Finally, calculate the New Priority Score:

$$\text{New Priority Score} = \text{Current Priority Score} + \text{Fairness Boost}$$

Explanation:

This equation ensures that the longer an individual waits beyond the max_wait_time threshold, the greater their priority boost will be. The boost_multiplier is a configurable value (e.g., 0.5) that determines how much each minute of "extra" waiting time contributes to the priority score. This makes the system more responsive to prolonged waits, preventing extreme cases of starvation more effectively than a fixed boost.

Example (User Scenario): Individual 401 waits too long and receives a priority boost based on extra waiting time.

- **Initial State:**
  - Individual ID: 401
  - Entry Time: 10:00 AM
  - Current Time: 10:30 AM
  - Fairness Threshold (max_wait_time): 25 minutes
  - Boost Multiplier (boost_multiplier, from Admin Console): 0.5 (assumed for example)
  - Urgency Level (initial): 2 (assumed for example)
  - Service Type (initial): 'regular' (assumed for example)
  - Service Type Score (initial): 5 (for 'regular', from Admin Console)
  - Current Priority Score (calculated initially based on 30 min actual wait):
    $\text{Current Priority Score}=(2\times0.5)+(30\times0.3)+(5\times0.2)\text{Current Priority Score}=1.0+9.0+1.0\text{Current Priority Score}=11.0$
- **System Processing:**
  - Waiting Time: Current Time - Entry Time = 30 minutes.
  - Check: 30 minutes > 25 minutes (Fairness Threshold). **Condition Met!**
  - Calculate Extra Waiting Time:
    $\text{Extra Waiting Time}=30\text{ minutes}-25\text{ minutes}=5\text{ minutes}$
  - Calculate Fairness Boost:
    $\text{Fairness Boost}=5\text{ minutes}\times0.5=2.5$
  - Calculate New Priority Score using the dynamic boost:
    $\text{New Priority Score}=\text{Current Priority Score}+\text{Fairness Boost}$
    $\text{New Priority Score}=11.0+2.5\text{New Priority Score}=13.5$
    Update position in the max-heap with the new score of 13.5.
- **System Output:** "Warning: Individual ID 401 has waited 30 minutes (5 minutes over threshold). Priority boosted to 13.5 due to fairness rule."

5. Simulation Mode

Description: This testing tool simulates queue behavior by generating random individuals and displaying real-time queue states, relying on prior queue management features.

Example (User Scenario): Simulate a burst of arrivals.

- **Input:**
  - Duration: 60 minutes
  - Arrival Rate: 1 every 2 minutes (0.5 individuals per minute)
  - Counters: 3
- **System Output (Snapshot):**

```
--- Time: 2 minutes ---
New Arrival: ID=106 (P=5.0)
Regular Queue: [ID=106 (P=5.0), ID=101 (P=4.8)]
```

6. Advanced Reporting and Sorting

Description: Post-service analysis is enabled by storing historical data and sorting it (e.g., by waiting time) to identify performance trends.

Example (User Scenario): Report on regular queue, last hour.

- **Input:**
  - Filter: 'regular'
  - Sort: Waiting time (descending)
- **System Output:**
  - ID=302 | regular | 45 min | 12 min (Service time)
  - ID=304 | regular | 38 min | 10 min (Service time)

7. Admin Console

Description: The Admin Console allows configuration of system parameters and thresholds, controlling all features' behavior.

Parameters and Thresholds:

- **Priority Weights:**
  - urgency_weight: weight for urgency (e.g., 0.5)
  - waiting_time_weight: weight for waiting time (e.g., 0.3)
  - service_type_weight: weight for service type (e.g., 0.2)
- **Service Type Scores:**
  - E.g., 'emergency': 10, 'vip': 8, 'regular': 5
- **Fairness Threshold:**
  - max_wait_time: time before priority boost (e.g., 25 minutes)
  - boost_multiplier: value determining how much each unit of extra waiting time contributes to the priority boost (e.g., 0.5)
- **Simulation Parameters:**
  - duration: simulation length (e.g., 60 minutes)
  - arrival_rate: individuals per minute (e.g., 0.5)
  - counters: number of service counters (e.g., 3)

Example (User Scenario): Adjust weights.

- **Input:**
  - urgency_weight = 0.4
  - waiting_time_weight = 0.4
  - service_type_weight = 0.2
- **System Output:** "Priority weights updated successfully."

8. Grade Distribution and Deliverables

Analysis and Design **(6 points)**

Deliverable: A document (PDF) that includes:

1. The role(s) of each team member and assigned features.
2. Detailed description of each feature, supported by examples that cover all possible inputs and their outputs.
3. Justifications of the selection of data structures for each feature in terms of time and space complexities.
4. Implementation logic for each feature written in pseudocode with commentary and explanation.

Submission Deadline (Google Classroom): Wednesday July 23 11:59pm

Implementation **(14 points)**

- Modular Priority Engine: 2 points
- Dynamic Priority Updates: 2 points
- Multiple Queues with Merging: 3 points
- Fairness Monitor: 2 points
- Simulation Mode: 3 points
- Advanced Reporting and Sorting: 1 point
- Admin Console: 1 point

Deliverables:

1. Source Code: A complete implementation of the Smart Queue Management System incorporating all seven features described above.
2. Executable File: Runnable .exe file.
3. Testing **and Demonstration:** Test cases or demonstrations for each feature to confirm correct operation.

Submission Deadline (Google Classroom): Saturday August 2 11:59pm

Notes:

- Each team member is expected to submit on Google Classroom.
- Grading will be based on the roles chosen, the assigned features and performance during the final discussions for the two phases.
- Without having clear separation of roles, an entire team will be deducted for any mistakes.