Air Traffic Control

Objective: Using a language of your choice, develop an Air Traffic Control system (ATCS) that meets the requirements listed below. The ATCS will allow the queuing and dequeuing of aircraft (AC).

Layers

The ATCS should have the following layers,

1. REST layer
    a. This layer provides access to the ATCS.
    b. It should provide endpoints for all of the ATC methods.
    c. Appropriate error messages should be displayed / returned.
        i. Ex: A method was called before the system has booted.
    d. It must be a REST interface.
        i. Hint: Follow traditional REST best practices
2. Logic layer
    a. This layer provides the logical operations of the ATCS.
    b. The it should provide the following functions
        i. Boot – start the system
        ii. Enqueue – Add an AC to the queue
        iii. Dequeue- Remove an AC from the queue based on priority
        iv. List – Provide the current order of the AC in the queue
    c. It should access the queue through the data layer.
3. Data layer
    a. This layer holds the data structure for the queue.
    b. The queue must be accessed through this layer, not directly.

Objects

The ATCS should have the following objects with the listed properties,

1. Aircraft (AC)
    a. ID
        i. This field will store the AC id
    b. Type
        i. Emergency, VIP, Passenger, or Cargo
    c. Size
        i. Small or Large

Dequeue Logic

The ATCS should dequeue planes with the following logic

1. VIP are more important than other types, except Emergency.
2. Emergency AC always have the highest priority.
3. Passenger ACs have a higher priority than Cargo ACs.
4. Large ACs of a given type have priority over Small ACs of the same type.
5. Earlier enqueued ACs of a given type and size have precedence over later enqueued ACs of the same type and size.

Requirements

1. Implement the ATCS with the requirements specified above.
2. Assume multiple users will concurrently access they system. Multiple air traffic controllers will ask for the next plane to be dequeued.
3. To the greatest extent possible, show all of your code. Feel free to use standard libraries provided by your chosen implementation language.
4. Please send your **complete project** including artifacts with build and deploy instructions.
5. **Please include a README me file with instructions on how to run the project locally.**
6. **Optional Bonuses:**
    a. Use a non in-memory/persistent datastore in the data layer.
    b. Implement **a** UI to show the queue and add/remove ACs.