The goal of this exercise is to demonstrate programming ability by developing a simple software subsystem. Work can be done in any language.

A software subsystem of an air-traffic control system is defined to manage a queue of aircraft (AC) in an airport. The aircraft queue is managed by a process that responds to three types of requests:

1. System boot used to start the system.
2. Enqueue aircraft used to insert a new AC into the system.
3. Dequeue aircraft used to remove an AC from the system.

AC’s have at least (but are not limited to having) the following properties:

1. AC type: Emergency, VIP, Passenger or Cargo
2. AC size: Small or Large

The process that manages the queue of AC’s satisfies the following:

1. There is no limit to the number of AC’s it can manage.
2. Dequeue aircraft requests result in selection of one AC for removal such that:
   1. VIP aircraft has precedence over all other ACs except Emergency. Emergency aircraft has highest priority.
   2. Passenger AC’s have removal precedence over Cargo AC’s.
   3. Large AC’s of a given type have removal precedence over Small AC’s of the same type.
   4. Earlier enqueued AC’s of a given type and size have precedence over later enqueued AC’s of the same type and size.

System Implementation Requirements

1. Develop one or more data structures to hold the state of an individual AC.
2. Develop one or more data structures to hold the state of the AC queue.
3. Define data structures and/or constants needed to represent requests.
4. Develop the code for three defined requests (queue, dequeue and list) and follows the above guidelines to implement an **aircraft queue manager**.
5. Assume multiple users of your dequeue implementation. Multiple air traffic controllers will ask for the next plane to be dequeued.
6. Expose the implementation as REST endpoints for integration with a 3rd party UI.
7. To the greatest extent possible, show all of your code. Feel free to use standard libraries provided by your chosen implementation language.
8. Please send your **complete project** including artifacts. If the project is built in Java, please use Maven, Gradle or Ivy/ANT for the build. **Please add README if needed to run the project locally.**
9. **Bonus:** Use a non in-memory/persistent datastore.
10. **Additional Bonus**: UI to show queue and add/remove planes