Priority Queuing using REST APIs

Technologies used:

- Spring Boot
- Hibernate
- JUnit
- Java 8
- Maven build
- MySql database
- H2 database for testing

How to use:
To run using IDE:

- Download Spring tool suite/Eclipse IDE
- Download the project from this github repo to {location on PC}
- In the IDE, Import-> Existing Maven Project -> {location on PC}
- Install Mysql
- Run the sql statements in dbschema.sql file to create the required schema and tables.
- Provide the data source, username and password in src/main/resources/application.properties
- Right click on src/main/java and select run as as Spring boot application
- To run the unit tests, right click on src/test/java and select run as a Junit test

To run using CLI:

- Install maven
- On the commandline:
  - cd <to the project folder>
  - mvn spring-boot:run

DB Design:
I've created a single table which stores the :

- Id: This is the user's id and is the primary key.
- Timestamp: This will be the current time passed to the api while adding the user
- Priority naming convention : normal=4, high_priority=3, VIP=2, management_override=1
- Rank: Used to store the rank according to the ranking rules.

Troubleshooting:
In case of any connection errors/locks while running unit tests, delete the DB.db.mv.db file under the project folder

Priority Queue endpoints:

Note:

Assume the time is in the form of epoch time milliseconds and the current time passed in the parameters are no more than the current epoch time.

1.An endpoint for adding a ID to queue (enqueue). This endpoint should accept two parameters, the ID to enqueue and the time at which the ID was added to the queue:
/work/addUser/{id}/{time} - GET request

2. An endpoint for getting the top ID from the queue and removing it (de- queue). This endpoint should return the highest ranked ID and the time it was entered into the queue.
/work/getTop -DELETE request

3. An endpoint for getting the list of IDs in the queue. This endpoint should return a list of IDs sorted from highest ranked to lowest.
/work/getIds -GET Request

4. An endpoint for removing a specific ID from the queue. This endpoint should accept a single parameter, the ID to remove
/work/deleteUser/{id} -DELETE Request

5. An endpoint to get the position of a specific ID in the queue. This endpoint should accept one parameter, the ID to get the position of. It should return the position of the ID in the queue indexed from 0.
/work/getPosition/{id} -DELETE Request

6. An endpoint to get the average wait time. This endpoint should accept a single parameter, the current time, and should return the average (mean) number of seconds that each ID has been waiting in the queue.
/work/getAvgWait/{time} -GET Request