Maintenance

Threads:
We are using a newCachedThreadPool for multi-threads which can potentially grow to Integer.MAX_VALUE threads. We don't have to set up the pool size manually. Also, cachedThread is reusable. If there is a free thread and no more tasks want to use it, the thread will be retrieved.
The thread lifetime preset is 50ms. As the data set grows, it can have a longer lifetime.
In the thread, we use callable objects to return the computed result done by a thread. In the future, the developer can add a logger to track the thread if there is a failure or timeout task.

Scalability:
The system only distinguishes two types of users, customer, and pilot. In the future, the system will differentiate many types of users. The Person and Employee class/table would be utilized to develop new types of users.

Database connection:
In the future, the user may access the service from different clients. In that way, the current database has to move to a client-server type database or deploy to a specific server that the user can access from the address.

Data maintenance:
Inactive user:
The user login feature is not provided, it is more related to the authority topic, so we postpone it. In this case, we cannot implement the achievability for the inactive user. For example, a user starts the service, but does nothing, then stops the service. The system can't get the user information without the login feature. So it is impossible to track the latest login time of users. The mentioned functionality can be done when the login feature is provided.

Deployment
We deploy running jar files to docker. As mentioned above, can add a logger feature to track the system if there is an unexpected shutdown. And any newly added feature can be tested without stopping the previous service. Once the new version passed the test, could replace the old jar file, which minimizes the update time.