Python Logger Application

Purpose of this application is to understand how one Python service can communicate with other Python services using sync/async manner. Also how data can be stored and retrieved in ElasticSearch in different formats.

Prepare following services to develop the entire application,

1. Data tracker service - Python
    a. Create an API to track following fields in ElasticSearch document
        i. User message
        ii. User ID
        iii. Request count
        iv. Category
            1. Direct
            2. Retried
            3. Failed
        v. Created time
    b. Provide a facility to bulk insert data in ElasticSearch
    c. Create an API to fetch all user messages in which provided text is available
    d. Create an API to get number of messages by category and by date
    e. Validate token passed in header in every request
2. User authentication service - Python
    a. Create an API to create new user in system
        i. Capture username and password from request payload
        ii. Username should be between 5 to 15 characters long and only alphanumeric is allowed
        iii. Password should be between 6 to 12 characters long
        iv. If data is valid, then store data in Redis, this will be used during login request
        v. Also add random ID as user ID in above Redis object
    b. Create an API to validate username and password (for login purpose)
        i. Validate username and password with Redis data
        ii. Generate JWT token if credentials are valid
    c. Pass generated token in every API requests header
3. Data pusher service - Python
    a. Create an API to initiate data pushing process
        i. This API will be invoked from REST client (e.g. Postman)
        ii. JWT token should be verified in this service
            1. If token is not valid, then throw respected error

iii. User ID needs to be retrieved from token
iv. User message will be passed in request body
v. Pass following data to “Data validator” service using async approach
    1. User ID
    2. User message
    3. Random number from 1 to 60
vi. Maintain a request counter in Redis with respect to each user, and increment that counter in each request received from user (e.g. user ID #2 has made 4 requests, user ID #5 has made 10 requests)
    1. Also pass that counter in above data payload

4. Data validator service - Python
    a. Validate received data from “Data pusher” service using following logic,
        i. If (random number % 10) == 0
            1. Generate random number again and process data again after 4 seconds
            2. Set category as “Retried” for this data to perform again
            3. If this condition is matching for 2nd time as well, then we need to set category as “Failed” and we need pass data to “Data tracker” service
        ii. Else invoke “Data tracker” service API to track data in ElasticSearch with “Direct” category
        iii. Try to implement async approach to pass data from this service to “Data tracker” service

Note:

1. Define Swagger document for API definition
2. Add unit test cases in all services