NOTES

The presented test-plan represents my vision upon testing the given Mailbox API. Given the fact that I was required to have an MVP approach, there have been some things that I assumed as functional and some things that I have added just to ensure a better coverage of the API; all of the assumed/added things will be treated in a more detailed manner in this document.

APPROACH

First of all, I have started from the idea that tests should be independent and relatively simple and they should provide a good overview of the API's specifications.

Before creating an API test-plan, I take into account the following aspects:

- What does the API do?
- Who will use the API?
- What are the most relevant environments to test the API in?
- What are the top priorities in matter of testing?
- What are the positive scenarios that can be written for the API?
- What are the negative scenarios that can be written for the API?
- How does the API respond to abnormal user behavior?
- When is a test considered PASSED and when is it considered FAILED?
- What is the design of the output?
- Should the API interact with other components/APIs of the application?
- What resources are needed for performing the API testing? (hours of testing, persons designated, etc)

After deciding on the above mentioned aspects, there should be a discussion about what kind of tests will be performed on the given API. Depending on what the scope of the API is, I will mention a few types of testing that should take place: functionality testing, load testing, exploratory testing, negative testing, security testing, etc.

TESTS

An important aspect of the written tests lies in the preconditions that need to be fulfilled before actually performing a certain test. Given the fact that the testing and development teams work together, when performing a testing session the tester needs to make sure that he/she chooses the right environment and that he can manipulate it without interferences from other persons.

When writing the tests found in the test-plan one of the priorities has been to cover all the given endpoints and to combine positive and negative testing.

ASSUMPTIONS

1. I assumed by using POST /v1/messages we can import the records in the DB from the JSON file;
2. Status 200 is returned when REST call is correctly executed;
3. Status 404 is returned when the parameter value does not correspond to an existing resource;
4. The two REST parameters “read” and “archived” admit both “true” and “false” values for the “Mark as read/unread” and “Archive/Unarchive” functionalities;

PROPOSALS/REMARKS

1. Include “Archived” and “Read” fields in Messages resource;
2. I have noticed that the UID field in the JSON is string and not long as specified in the API documentation;
3. I think that there should be a design implemented for the pagination that is required in the task. I haven’t seen any rules for it, therefore I would consider adding them to the JSON (items/page, page count, message count, sorting criteria, etc.);
4. Since the given API is a simple mailbox and testing is not based on assumptions, I have limited the test-plan to the information that has been provided. The test plan can be extended based on further knowledge of the API/product and the components with which it will be integrated;
5. Added to the automated tests that I have provided, I would also consider performing load testing (i.e JMeter);
6. The API should not only use a basic authorization header for archiving, but for all the endpoints since the application is a mailbox and a user should not be able to access all messages stored in the database.