Instructions

Task 1

Imagine the following situation: you are joining a cross-functional team which builds a front-end application using REST APIs. You are a first QA engineer and need to establish a QA process in the team.

1. What would you do in your first few days of work? Where would you start?

I will work with the team to define the scope of the Business functionality in other words the purpose of APIs. Then I would start understanding the applications workflow and how APIs perform/interacts for CRUD operations. Once the scoping is defined , would start with Test Planning and Test Strategy to ensure the success of testing process for manually testing API's.

Once the build is stable, then feasibility study for commences. Test Planning for automation includes the identification of test framework/test tools .Several factors like Time to market, tool age ,licensing of the software, frequency of code deployment contributes to the automation plan.

2. Which process would you establish around testing new functionality?

Test Planning/Epic planning:

- *Develop a test plan to ensure the testing scope is defined and agreed upon by different stakeholders.*
- *Scoping can list down the APIs under test and the description/link to API documentation*
- *Descoped functionalities are also clearly mentioned*
- *Testing types like performance testing, security testing which are agreed are also listed*
- *Test Framework which is identified for test and the test environmental set up required(includes tools and servers) are documented*
- *General test planning activities like risk mitigation, staffing, test schedules are documented in planning*

Test Design:

- *Test Scenario and Test cases templates are established to ensure the uniformity among the team*
- *I will apply various test design techniques like error guessing , boundary value, equivalence portioning to arrive at test scenarios*
- *Then test cases are derived capturing the details of API declaration*
- *Test data is prepared and documented in test case wherever appropriate*
- *Test cases are organized to group together the APIs which ideally groups the Business requirements*
- *Dependencies among the APIs are defined and the end to end API test case is designed*

- *Test Automation Framework Development*

    - *Test automation framework is developed based on the agreed Test Plan. We should make use of existing open source framework. Associated benefits include community for code support, cost etc.*
    - *Proper design pattern should be used. Popular patterns are Singleton, Page Object model. Type of framework is determined which best suits the requirements*
    - *Started to create test scripts for the identified/developed test scenarios in the established framework*
- *<u>Test Execution:</u>*
    - *Test execution of the identified test cases are performed*
    - *Test Reports are validated for errors (running manually) and identified if it's a bug or needs to be addressed*
    - *Test cases suites are modified if necessary, in case of changes in requirements/design*
    - *Test scripts are modified if the functionality is working as expected and the code needs a tweak.*
    - *If found as bug, the development team is communicated, and the defect triage is performed*
- *<u>Test Log/Reporting</u>*
    - *Test reports for each run is documented and shared as agreed upon*
- *<u>Test Closure:</u>*
    - *The progress of test automation is shared at the end of each Sprint or agreed duration. This gives understanding of the automation progress as well as the quality of code in continuous integration*

3. Which techniques or best practices in terms of code architecture and test design would you use in your automated tests?

- *Code architecture:*
    - *The benchmarked coding standard should be adhered. There should be no hard coding of variables . Almost everything should be parametrized. In case of unavoidable circumstances, proper explanation should be provided.*
    - *Modular code . Complex validations should be modularized as much possible.*
    - *Loosely coupled code should be achieved.*
    - *Logical grouping of functions should be established, and appropriate naming conventions should be used. For instance, reusable modules , setting priority for the tests.*
    - *Proper design pattern should be used . Popular among the established patterns are*
    - *Singleton design pattern or page object model.*
    - *We should try to make use of Open source automation framework. There are few advantages like the community , price.*
    - *Often, the user is confused with the default error message received from API. Hence property files can be used to customize the error messages.*
- *Validation Points:*
    - *Ensure that the basic validations are covered like Authentication, Authorization, status code, body, special headers if any*
    - *In addition to correct response, application state is tested*
- *Test Scenarios:*
    - *Traceability matrix is recommended as the bug identified could be tracked accurately*
    - *Ensure all happy path scenarios are covered with mandatory data. In addition, the API response is also tested optional data*
    - *Negative test scenarios are covered with invalid data formats and validate if the invalid values are handled*
    - *Negative test scenarios also focus to test with acceptable data format but violates the Business rule and validates if exceptions are handled*
    - *Test Scenarios are created to ensure appropriate error handling*
    - *Test scenarios should cover missing / duplicate functionalities and validate the system response*
- *Test Cases/Test Data:*
    - *Each test case can focus on testing one API*

- *Organize the APIS functionally to design End to End functional test cases where series of APIs are called*
- *Test Data playa a very vital role. Effort should be spent to determine data very close to real time , also not ignoring negative tests.*

- *<u>Non-Functional:</u>*
  - *Even if the performance / security is not in initial scope of the project, would ensure that the time processing of requests is normal, and no sensitive data is passed*

- *<u>API Documentation/Others:</u>*
  - *Test Cases are created to test the API documentation like endpoints naming convention, expected error messages and functionality*
  - *Wherever feasible, can test the web UI for one positive and negative data*
  - *Perform sanity and smoke testing and ensure the functionality works correctly*
  - *Apply exploratory test techniques to test the APIs with an attempt to fail the system*