- **Task1:**

Imagine the following situation. You need to establish a QA process in a cross-functional team.
The team builds a front-end application using REST APIs.

Q1. Where would you start? What would be your first steps?
A1.

- I will start by understanding product, going through documentation, training videos
- Will go through the API Specification/ Design Document for the features and Requirement Specification of the task/feature that will be assigned to me.
- Understanding existing test framework if any or come automation plan/ POC of framework for testing a new feature

Q2. Which process would you establish around testing new functionality? How would you want the features to be tested?
A2.

- A test plan document for each feature defining the scope / functional / non functional test to be targeted after undergoing Requirement Analysis and establishing Feature Benchmarks.
- Agile methodology in Test Case automation development, where the new test are added and executed while the feature is getting coded and in develop branch. Breaking down the automation task in stories for QA Engineer and defining what stories to be prioritised and completed first and time for each story.
- Acceptance test for feature / End 2 End test verification
- Release Testing

Q3. Which tools would you suggest using to help your team with a daily work?
A3.

- Test management tool like TestRail , JIRA for defect/issue/task tracking.
- POSTMAN for RESTFul , Chrome Developer tool for UI
- Pytest (or python based test framework), Selenium, Docker,
- Jenkins for CI
- Wikis for Release Communication

Q4. If you would do a test automation which techniques or best practices would you use the Application?
A4.

- I will try to follow the test pyramid closely.
- Try to have more than 80% automation, with min 30% focusing on negative flows
- BDD based test frameworks for acceptance test / Choose test tool with high number of users and active contributions.
- Continous Integration methodology of creating a feature TAG only if all the acceptance test passed using a Pipeline possibly with every checkin or Scheduled invocation

- Proper reporting capability with email notifications
- Benchmarking results to build verification

- **Task2 (RESTful API Automation Tests)**

  How would you test an HTTP API of some service? Choose some service from this list https://github.com/toddmotto/public-apis or any other of your choice. It is better to pick one without authorization as it is easier to test.

  Solution for Task2

  REST API chosen from https://github.com/toddmotto/public-apis is **Bhagavad Gita**

  API | Description | Auth | HTTPS | CORS
  Bhagavad Gita | Bhagavad Gita text | OAuth | Yes | Yes

  OAuth Authentication
  Client ID: KqGGPsGSe2RLTO0hwBnuSWZwIMj0HYM2WIf1L3EO
  Client Secret: isw5XWSSaZJOH4yQrZylpF7PWtvLfWBwrmrOszuthly5tDXqUP

  1.
  curl -X POST "https://bhagavadgita.io/auth/oauth/token" -H "accept: application/json" -H "content-type: application/x-www-form-urlencoded" -d "client_id=KqGGPsGSe2RLTO0hwBnuSWZwIMj0HYM2WIf1L3EO&client_secret=isw5XWSSaZJOH4yQrZylpF7PWtvLfWBwrmrOszuthly5tDXqUP&grant_type=client_credentials&scope=verse%20chapter"

  2.
  curl -X GET "https://bhagavadgita.io/api/v1/chapters" -H 'Authorization: Bearer TOhHtn4BF6HZCxSUA9zCjP1cR9MKr2'

  Q1. Document several test cases.

  A1.

Test Case No.	Step	Expected Result
Test 1	Prerequisite: Authenticate the API using oAuth 1. GET /chapters endpoint , verify there are total 18 chapters 2. Verify chapter 6 'name_meaning'	1. GET should return 200 OK , Total 18 chapters are returned 2. Chapter 6 'name_meaning' is 'Path of Meditation'
Test 2	Prerequisite: Authenticate the	1. Total number of verses in chapter 6 is 47

	API using oAuth 1. GET /chapters/6/verses, verify total number of verses are 47 2. Verify verse 1 of chapter 6 starting	2. Verse number : 1 of chapter 6 starts with 'The Blessed Lord said'

Q2. Implement one or two automated tests based on the test cases.

A2. The solution is coded in GitHub repo robot_framework_tests and the tests are present in test_api_bhagwadgita.robot

```
├──── geckodriver.exe
├──── Logs
│    ├──── api
│    │    └──── latest
│    └──── gui
│         └──── latest
├──── Modules
│    ├──── __init__.py
│    ├──── api
│    │    ├──── __init__.py
│    │    └──── bhagwadgita_api.py
│    ├──── api_util.py
│    ├──── gui
│    │    ├──── __init__.py
│    │    ├──── search_page.py
│    │    └──── search_result_page.py
│    └──── util.py
├──── ObjectRepo
│    ├──── obj_search_page.py
│    └──── obj_search_result_page.py
├──── README.md
├──── requirements.txt
├──── Resources
│    ├──── api
│    │    └──── bhagwadgita_resource.robot
│    └──── gui
│         ├──── search_resource.robot
│         └──── search_result_resource.robot
└──── Tests
     ├──── test_api_bhagwadgita.robot
     └──── test_search.robot
```

Task2 Alternative (Web Automation Tests)

Pick it instead of Task 2 if you think you are more proficient in frontend automation than in backend automation and this task will reveal you skills better.
How would you test search UI functionality of your favourite website (e.g. https://medium.com , https://www.google.de)? Choose your favourite website which has search functionality.

Solution for Task2 Alternative

Google.com is chosen for verifying search functionality.

Q1. Document several test cases.

A1.

Test Case No.	Step	Expected Result
Test 1	Input search term in the google search box and click Google Search button	1. Non zero number of results are displayed 2. Description of the each result containa searched text atleast once.
Test 2	Input Search term in Google search box and click one of the auto suggestion shown for the input text	1. Non zero number of results are displayed 2. Description of the each result containa searched text atleast once.
Test 3	Prerequisite: Note the first result URL for a input text 'abc' on clicking 'Google Search' button Input Search term in Google search box and click I am feeling lucky button	Redirected directly to the first result URL noted in the prerequisite step.

Q2. Implement one or two automated tests based on the test cases.

A2. The solution is coded in GitHub repo robot_framework_tests and the tests are present in test_search.robot

```
├──── geckodriver.exe
├──── Logs
│    ├──── api
│    │    └──── latest
│    └──── gui
│         └──── latest
├──── Modules
│    ├──── __init__.py
│    ├──── api
│    │    ├──── __init__.py
│    │    └──── bhagwadgita_api.py
│    ├──── api_util.py
│    ├──── gui
│    │    ├──── __init__.py
│    │    ├──── search_page.py
│    │    └──── search_result_page.py
│    └──── util.py
├──── ObjectRepo
│    ├──── obj_search_page.py
│    └──── obj_search_result_page.py
├──── README.md
├──── requirements.txt
├──── Resources
│    ├──── api
│    │    └──── bhagwadgita_resource.robot
│    └──── gui
│         ├──── search_resource.robot
│         └──── search_result_resource.robot
└──── Tests
     ├──── test_api_bhagwadgita.robot
     └──── test_search.robot
```

- **Bonus Project (Py.test and RESTful API) coded in past for Test Assignment from Plivo company**

I will also request the interview panel to go through one of the existing project in my GitHub repository name rupesharlekar/plivo.api

Project description:

Py.test based test automation framework for RESTful API functional/acceptance testing verification.

It was coded as a solution to one of the Test Assignment given to me by Plivo during there interview process. Please refer the README.md file of the project to get more idea.

```
├──── Message API.pdf
├──── README.md
├──── requirements.txt
└──── src
   └──── tests
      ├──── __init__.py
      ├──── conftest.py
      ├──── functional
      │     ├──── __init__.py
      │     └──── test_messaging.py
      └──── PlivoConfig.py
```