**Additional Questions**

How do you review code?

I would consider the following steps in code review:

* Ensure coding standards are met. For example
  + Descriptive Test suite and Test case names.
  + Keywords are defined under Resource folder and proper folder structure to be followed.
  + Locators should always be assigned to variables and only variables are used in code. Div class locators are the preferred ones to be used and they should be start with uit-xx
  + Proper alignment of code.
  + Commented code lines do not exist in the file.
  + Main test suite file name should always end with Suite (MainTestSuite.robot)
  + Suite set up and Tear down to be added under settings.
* As a best practice, test data is always stored separately as json or csv files. Usage of test data directly in the code should be avoided.
* Code logic – Try to achieve the functionality in the best possible way.
* Common keywords are placed under Robot Library (created by us as a best practice) and organized under specific components so that they could be used by later by the team.
* Test run history – If a specific test fails multiple times at the same section in code due to timeouts before passing, try to refactor it.
* Usage of sleep should be avoided in the code. Instead Wait for keywords could be utilized.
* Good to include documentation for complex suites.

How do you enforce coding standards?

* We have a coding style guide available for Robot Framework, so I would make sure to follow the rules mentioned in the style guide during coding.
* I was part of brainstorming sessions on coding standards and adopted the best practices on standards proposed.
* Tests are available for Scala and Angular coding standards which checks for pre-defined standard that need to pass in Jenkins for all the pull requests.

How do you plan what kind of approach you take for test automation - what libraries to use, how does it work in couple of years, how to make it easy to maintain, etc? What are the main points to consider?

* The main approach that I would consider for test automation is to identify the test cases to be automated. Main functionalities need to be automated; inputs/outputs and complete system flow needs to be tested, but UI look/feel (colors, icon display) could be left only for UX review or manual check.
* Based on the feature to be automated, the libraries to be used are identified. For example, if the feature under development deals with dictionaries and dictionary operations, then good to include Collections dictionary.
* I will always separate Test data from code, in the sense no hard coding of data in the code. So for some code changes, only test data file could be modified and it is easy to maintain.
* I would always use variables for specifying the locators, so it is easy to modify later.
* I would organize the common keywords in the robot library under specific components, so it is easy for any new person to understand and pick it from there.
* I would periodically check and clear any tests related to old functionality that do not exist anymore. Also splitting of single complex functionality into multiple suites is a good option.

Code testability, how do you enforce it?

* I would ensure that there are no dependencies between suites and that every test suite could be executed independently. This would help to identify issues easily.
* I would write tests that are easy to understand and include documentation whenever necessary.
* I would use variables and pass parameters to functions/keywords so it can be tested easily.

How do you make sure that the product is testable?

* I would check to make sure that requirements are clear and that there are no open items associated with it.
* In the Test Design Development (TDD) meetings, I would go through each of the user stories to check if they are ready for development/test – which means that there is no ambiguity in the requirement.
* The product should be able to operate in specific environments and development for that product is complete.