Explain your approach to the implementation?

I used a keyword-driven approach; The approach allows good script usability and re-usability, good test flow coverage and the ability to reproduce test results.

How do you review the code?

First, I figure out what the code is doing, why it's needed, test the code, check if there are any errors or warnings. Inspect the code for code styling, naming inconsistencies. Check the code for error handling.

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

When I get requirements, I write the test cases in the way shown below; after that, I think if I know how to automate the tests, if I know then I know what libraries might be helpful if I don't know, I google, watch YouTube, Udemy videos, and then I come to what libraries to use based on how other people solved a similar problem.

T*o make it easy to maintain test automation – we need to follow clean code principles and develop test cases that are easy to understand.*

Test Case Name:	
PRE-CONDITIONS	
Action	Expected Result
TEST STEPS	
Action	Expected Result
POST-CONDITIONS	
Action	Expected Result

Code testability, how do you enforce it?

- Use unit testing.
- Follow clean code principles.

How do you make sure that the product is testable?

First, during requirements review sessions, I make sure product requirements are testable; when that is done, I start creating test data, test cases. After development is done/end of development, I create automation tests based on test cases traced back to requirements. Test Product is testable if the requirements are testable.