Test Answers:

Q1. Do you think the application satisfies the acceptance criteria?

\*Run the application to see the output, browse the implementation and write

a short summary of why you think the application does / doesn't meet the

acceptance criteria\*

Q1 Answer:

1.The variable ukDateTime was implemented using DateTime.Now, which captures the system’s local time. This approach assumes the local machine is set to UK time, which may not always be the case

2. The application does not successfully display the difference in time between UK and Canada - 'Must display the difference in time between the UK and Canada'

3.The code successfully meets the requirement to display the date in the specified format (dddd dd MMMM yyyy HH:mm:ss).

Q2. Refactor the code to be more testable.

Q2 Answer:

The project was the refactored to the classes and method below:

-- ITimeApiService Interface: ITimeApiService is defined as an abstraction for fetching the current time. The change will allow for easy mocking during tests.

-- WorldTimeApiService Implementation: This class implements ITimeApiService and uses HttpClient to retrieve time from the API.

-- TimeDifferenceEvaluator Class: This class handles the logic for formatting the date, displaying time, and calculating the time difference. By extracting this logic, we are able to test it independently from the API calls.

-- DisplayTimeDifferenceMessage Method: This method isolates the logic for displaying time difference messages, making it easy to test the method without having to rely on real date values.

-- An Interface should have been created for the Client. It was not implemented to avoid complicating the change.

Q3. What types of automated tests, and test cases would you write to test

this application?

Q3 Answer:

I implemented two types of automated tests: unit tests and integration tests. The unit tests focus on validating the logic within the TimeDifferenceEvaluator class, ensuring that its core functionality is quickly and efficiently tested. The integration test is designed specifically to verify connectivity to the Time API service, confirming that we can retrieve valid data from the service. By isolating connectivity in the integration test, I avoided duplicating scenarios already covered by the unit tests.

The following unit tests were written:

1. DisplayTimeDifferenceShouldReturnCorrectMessageWhenCanadaIsAheadOfUk

2. DisplayTimeDifferenceShouldReturnCorrectMessageWhenUkIsAheadOfCanada

3. FormatDateTimeShouldReturnFormattedDateTime

4. GetTimeDifferenceShouldReturnCorrect

Integration test:

VerifyThatTheUserCanGetTimeFromService