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.

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

1. Product Understanding:

I would start with understanding the functionality of the overall product. I would spend some time to understand how the UI is utilizing the backend apis exposed by the rest server. I would check the request and response for each query through the "Developer Tools". Once I get the thorough knowledge of the apis, I will try the same with different test data through Postman.

2. Automation Framework:

I would try to understand the existing automation framework for backend and UI. If the framework is not in place I will start thinking about the design of overall framework design.

3. CI/CD pipelines:

I will try to understand the CI/CD pipelines established in the organisation. If these pipelines are not in place, I will initiate the conversation to build one.

Which process would you establish around testing new functionality?

1. Test Plan:

I would try to create a process where the Test Plan is created and reviewed at the very start of the new feature development. The whole team (QA+Developers) would contribute to create the Test Plan. Test cases should be written in a test management tool and are also marked with priority.

2. Automation First:

Before delivering any new feature or functionality I would ask the QA to identify the P0/P1 test cases. Once the TCs are identified I would make it mandatory to have 100% automation of these TCs before releasing the feature.

3. Test Pipelines:

Jenkins pipelines should be created to run the automation suite everyday against the master branch. This would help to catch the bug earlier.

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

1. **Design pattern:**

Based on the product architecture, I would think about the design pattern for the test framework.
for eg:
-> **UI Framework**: For UI automation I would think about the ***Page object model***.
Here, pages in the app are represented as Classes, and various UI elements of that pages are defined as variables.
-> **Backend Framework**: For Backend framework I would try to find out the optimal design framework that would work efficiently with the product framework. Some of the design patterns that I would consider are:
Factory design pattern
Facade pattern
Singleton pattern

2. Simulators:

In case there are a number of components in the product communicating with each other, I would create the simulators so that we can mock the components that are not getting tested in the current run.
For eg:
If there are three services A, B and C communicating with each other in our product. To test A I would create the simulators for B and C using Flask. This would help in the component testing and it will remove the dependency of having all the components up and running for a single component testing.

3. Flaky test identification:

I would try to establish a process that would help in identifying the flaky test cases. This would help in building the trust on automation test run results.

4. **Defect management and Project tracking tool integration with test framework:**

If suppose I am using the Testrail as the Defect management tool and JIRA as the project tracking tool. I would integrate the test framework with Testrail and JIRA using their API's. This would help to automatically log the defect and test status.