Relayr Task 1

Q Where would you start? What would be your first steps

Ans:
Considering we already have some QA Engineers in our team we will start with assigning different modules to each QA. Initially, QA will explore their respective modules and go through the module documents (if available) to get hands-on experience of the product. The team will start testing with REST API then move to End to End testing of Web UI. Initially, QA will start their activities with writing smoke and sanity test cases in testware and start automating them.

Few first steps to set up QA process:

1. Creating test plans & strategies by discussing timelines and constraints with stakeholders
2. Ask Developers to give documentation of REST API so that QA can refer them for writing test cases and automation script
3. Ask PM for module documentation, if it is not available QA will write test cases by doing exploratory testing
4. Involve Product Manager and feature Developer during test case review so that they can also understand cases and provide feedback which can be use to find edge cases
5. Defining work-flows for various tasks such as QA flow and Bug flow as given below:
   QA work-flow : Story-> QA task -> Identify Cases -> Test on Branch ->Automation -> Test on Release branch -> Released -> Done
   Bug work-flow: Bug -> Development -> Code-Review-> Test on Branch ->Automation -> Test on Release branch -> Released-> Done
6. Setting up daily stand-up meeting in QA team to discuss daily status and blockers
7. Encourage every QA to learn automation and start contributing in automation suite

Q Testing new functionality and how do you want features to be tested?

Ans:
We will start with creating test cases for features (if a feature is new) then we execute those test cases and other related test cases on developer's branch manually, then we create automation script based on test cases written earlier. If the feature is old we can simply add new test cases on existing automation regression suite and run it.

Ask Developer to merge feature branch to release branch where we will execute our automation regression suite to go final go-ahead before release.

Q Tools suggestion for team:

Ans:

1. JIRA for bugs tracking, releases and sprint planning
2. TestRail as a Testware where QA will write manual test cases
3. InteliJ IDEA as a Java IDE to create automation scripts in Java
4. Github for storing repositories on cloud
5. Postman to test REST API manually
6. Jenkins for continuous integration and development, also to define when, where, and how to run automated tests
7. Putty/ iTerm to ssh into staging servers for checking out branches, checking logs, accessing DB etc.
8. Postico/MySQL workbench for DB access
9. Slack or Skype for internal communication

Q Automation Framework & Techniques to be Used:

Ans:
In automation, we will start with API automation because API does not change frequently as compared to UI.
After completing API automation we will move to Web UI automation for this we can use Selenium with Java/ Any PL depending on resources.

Some guidelines or techniques for writing automation script:

- ·Know your product inside out before starting the automation
- ·Always decide what all cases to automate before start writing scripts
- ·Always provide logical and high-quality data in your scripts
- ·Always Take screenshots for failed test cases
- ·Use soft assertions if you need to make a list of related checks on the same web page
- ·Name your tests wisely
- ·Follow standard coding guidelines while writing scripts

Automation Framework:
Every automation framework has their own pros and cons, choosing automation framework is mostly depends on the type of application we are testing. For a general web application, we can use **POM framework** because of its simplicity and advantages are given below:

●・Operations and flows would be kept separately from verification. This concept makes our code cleaner and easy to understand.
●・It improves the code reusability it means we don't have to change different locators whenever there's a UI change.
●・Maintainability of code becomes easier in this framework.
●・Helps reduce duplication of code.