1. Determine the Test Requirements

Identify how to use the application's features, such as user login, registration, product search, add to cart, checkout, and order history and break down these functionalities into test cases based on functional requirements, non-functional requirements.

2. Define Test Automation Architecture

Defining test automation architecture involves creating a clear structure for how automated tests will be organized and executed.

- Keyword-driven framework - To make the scripts reusable and readable, define keywords (such as "Login" and "Search Product") for each action.
- Modular-driven framework - To improve reusability, divide tests into modules (such as login, search, and add to cart).
- Behavior-driven framework - describes test scenarios in simple terms, making it simple for all stakeholders to understand and verify user interactions.
- Data-driven framework - In order to test different data sets without modifying the script, store test data in external files (such as Excel, CSV, or JSON).
- Hybrid framework - combines multiple testing approaches, such as data-driven and modular-based

3. Select Testing Framework and Tools

- Testing Framework - Select a dependable framework for automating tests that is compatible with the application's technological stack (e.g., Selenium WebDriver, Cypress for web-based apps).
- Programming Language - Select a language (such as Java, Python, or JavaScript) that the team is comfortable with and that works well with the framework.
- Build Tool - To handle dependencies and build processes, use Gradle or Maven.
- Integration of CI/CD: For test deployment and continuous integration, integrate with GitHub Actions, GitLab CI, or Jenkins.

4. Make Sure the Code Has the Correct Folder Structure

Test Cases, Utilities, Libraries, Logs, Test Data, Constants, Page Objects, and Locators are arranged into different directories inside the folder structure of a test automation framework. This configuration guarantees the distinct separation of various components, which improves the framework's manageability and makes updates and maintenance simpler.

5. Develop Test Data

Creating and maintaining a variety of input value sets is necessary to develop test data in order to fully test the operation of the application. To guarantee thorough coverage, this involves producing data for a range of scenarios, such as valid and invalid inputs.