**Brief Explanation of the Test Cases**

1. **AcceptingCookiesTest.java**:  
   This test script verifies that the website correctly handles cookies and "Let Me Hack!" button can be clicked.
2. **BookRoomAdminTest.java**:  
   This test script focuses on the administrative functionalities related to booking rooms. It includes booking a room through an admin login. The name and last name are configured in the config.properties file as name = Sam and lastName = Peter.
3. **BookRoomTest.java**:  
   This test script allows an end-user to book a room. The customer can book a room using this script. The details related to the customer are configured in the config.properties file. The details taken from config.properties are name = Sam, lastName = Peter, email = Sampeters2025@gmail.com, and phone = 01234567890
4. **CheckingLinksTest.java**:  
   This test script ensures that some hyperlinks on the website are valid and lead to the correct destinations. It helps in identifying broken links that could negatively impact the user experience. For example, when clicking on "Read about feature," it opens a new page, but no information related to the feature is available. The gitHubFeatureLink will fail, causing this test script or the test suite to fail individually.
5. **ConactDetailsRetriveDBTest.java**:  
   This test script verifies that contact details sent from the home page are available for the admin user to check. After sending the contact details, the admin user logs in and checks the information messages. The customer details are configured in the config.properties file. The details taken are name = Sam, email = Sampeters2025@gmail.com, phone = 01234567890, subject = test information for booking room, and message = This is a test message and want to check contact details for the booking information.
6. **ConactDetailsSendTest.java**:  
   This test script checks the functionality of sending contact details through the contact section on the main home page. It ensures that the data is correctly transmitted. The customer details are configured in the config.properties file. The details taken are name = Sam, email = Sampeters2025@gmail.com, phone = 01234567890, subject = test information for booking room, and message = This is a test message and want to check contact details for the booking information.
7. **CreateRoomTest.java**:  
   This test script focuses on the functionality of creating new rooms within the system. It verifies that rooms can be added with the correct details and that they are properly stored and displayed using the admin user. The username and password are taken from the config.properties file: username = admin, password = password, room = 1001, and price = 200.
8. **HomePageTest.java**:  
   This test script checks the overall functionality of the main home page and verifies if the home page has been opened successfully.
9. **LoginTest.java**:  
   This test script verifies the login functionality, ensuring that users can log in with valid credentials. The username and password are taken from the config.properties file: username = admin and password = password.
10. **LoginTestDataDriven.java**:  
    This test script is a more advanced version of the login test, using multiple sets of credentials to test various scenarios. It helps in identifying edge cases and ensures robust handling of different input combinations. It is a data-driven test script where an Excel sheet is used for storing data, and that data is used for checking login functionality. Currently, the test script is failing because when invalid login credentials are entered, no error message is displayed, making it impossible to compare the results. The required username and password details are taken from the TestData.xlsx file.
11. **NFTConcurrentUserLoadTest.java**:  
    This test script checks how the system behaves under a load of concurrent users. It ensures that the application can handle multiple users simultaneously without performance degradation. The number of concurrent users that can log in to the website is configured in the config.properties file: concurtUser = 20.
12. **NFTHomePageStressTest.java**:  
    This test script subjects the home page to extreme conditions to identify its breaking point. It helps in understanding how the home page performs under stress and ensures it can handle high traffic volumes. The number of users is configured in the config.properties file: stresstestno = 100.
13. **NFTPerformanceHomePageTest.java**:  
    This test script measures the performance of the home page, focusing on metrics like load time and responsiveness. It ensures that the home page meets performance standards. The threshold for opening the home page is set to **5000ms**, which is configurable.
14. **NFTPerformanceLoginTest.java**:  
    Similar to the home page performance test, this script focuses on the login functionality, ensuring that it performs well. The threshold for opening the login page is set to **5000ms**.The username and password are taken from the config.properties file: username = admin and password = password.
15. **NFTResponsivenessLoginTest.java**:  
    This test checks the responsiveness of the login functionality, ensuring that it works correctly across different devices and screen sizes, such as mobile devices. The username and password are taken from the config.properties file: username = admin and password = password.
16. **NFTScalabilityLoginTest.java**:  
    This test evaluates how well the login functionality scales with an increasing number of users. It ensures that the system can accommodate growth without compromising performance. The username and password are taken from the config.properties file: username = admin and password = password.

**Configuration Details in config.properties File**

The test scripts take data from the config.properties file, which contains the following configurations:

1. **Browser Selection**: This refers to choosing which browser (e.g., Chrome, Firefox, Edge) will be used for running the test scripts.
2. **Headless Mode**: Setting headless to true allows the browser to run in the background without a graphical user interface (GUI). This is useful for running tests in environments where a display is not available, such as on a server or in a CI/CD pipeline.

*#browser = edge*

*#browser = chrome*

*browser = chromium*

*#browser = firefox*

*#browser = safari*

*url = https://automationintesting.online/*

*username = admin*

*password = password*

*headless = false*

*name = Sam*

*lastName = Peter*

*email = Sampeters2025@gmail.com*

*phone = 01234567890*

*subject = test information for booking room*

*message = This is a test message and want to check contact details for the booking information*

*room = 1001*

*price = 200*

*headless = false*