Automating the Star Health Application Using Selenium with Java

Introduction:

Automation testing plays a pivotal role in ensuring the quality and efficiency of web applications. Selenium, a widely used open-source tool, provides a robust framework for automating web browsers. In this write-up, we will guide you through the steps to automate the Star Health Application using Selenium with Java.

Prerequisites:

Java Development Kit (JDK): Ensure the JDK is installed on your machine to compile and execute Java code.

Integrated Development Environment (IDE): Use a Java IDE like Eclipse or IntelliJ IDEA for development.

Selenium WebDriver: Include Selenium WebDriver libraries in your Java project to interact with web elements.

Browser Driver: Download the appropriate browser driver (e.g., ChromeDriver) compatible with your browser version.

Setup:

1. Create a Java Project:

Initialize a new Java project in your preferred IDE.

Add Selenium WebDriver dependencies to your project, either manually or by configuring the build tool like Maven.

2. Configure Browser Driver:

Download the compatible browser driver and specify its path in your Java code using the System.setProperty() method.

java

```
System.setProperty("webdriver.chrome.driver", "path_to_chromedriver.exe");
```

Implementation:

3. Selenium Test Script:

Develop a Selenium test script to navigate to the Star Health Application homepage and perform various automation tasks.

java

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class StarHealthAutomation {

    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "path_to_chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.get("https://www.starhealth.in/");
        driver.quit();
    }
}
```

4. Automation Steps:

Implement specific automation steps like logging into the application, filling forms, clicking buttons, and validating elements.

java

```
driver.findElement(By.id("username")).sendKeys("your_username");
driver.findElement(By.id("admin")).sendKeys("password");
```

```
driver.findElement(By.id("loginButton")).click();
```

Best Practices:

Synchronization: Handle synchronization issues using appropriate waits like implicit, explicit, or fluent waits.

Page Object Model (POM): Organize your code using the POM design pattern for maintainability and readability.

Error Handling: Implement error handling and reporting mechanisms to capture and log failures during test execution.

Conclusion:

Automating the Star Health Application using Selenium with Java involves setting up the Java environment, configuring browser drivers, writing Selenium test scripts, and implementing automation steps. By following the outlined steps and best practices, you can efficiently automate the Star Health Application, ensuring its functionality and reliability.