**Test Otomasyon Aracı Kurulum ve Kullanım Talimatları Dokümanı**

1. **Proje Tanımı**

Proje; test otomasyonu sağlanması için yazılan test senaryolarının kolay bir dille yazılmasını, test sonuçlarının açıkça raporlanması ve dinamik bir test geliştirme ortamı sağlanması amacıyla oluşturulmuştur. Maven ile oluşturulan Java projesinde Selenium, Cucumber, RestAssured gibi teknolojiler kullanılarak entegre edilmiştir.

Proje Visual Studio Code üzerinde geliştirilmiştir fakat farklı IDE’lerde de çalıştırılabilmesi ve test senaryolarının sorunsuz çalışması amaçlanmaktadır.

Kullanıcı(test developer) .features dosyaları içerisine yeni senaryolar ekleyerek RunTest classını çalıştırdığında Gherkin dilinde yazılan yeni test senaryoları projeye entegre olarak sorunsuz çalışmalıdır.

Şu aşamada framework yeterince dinamik olmadığı için eklenen senaryolar classlardaki mevcut methodlarca karşılanamayarak yeni methodlar eklemek gerekebilir. Detaylı bilgiler ve kurulum aşamaları dokümanın kalanında anlatılmıştır.

1. **Kurulumlar**
   1. **Java Kurulumu**

<https://www.oracle.com/java/technologies/downloads/> üzerinden Java JDK Installer indirilir ve adımlar takip edilerek Java JDK kurulur. Kurulumun tamamlandığından emin olmak için terminalden “java -version” komutu verilerek java versiyonu doğrulanır.

* 1. **Maven Kurulumu**

<https://maven.apache.org/download.cgi> üzerinden Maven indirilerek zipten çıkartılan klasör “C:\Program Files\Apache\maven” gibi bir dizine yerleştirilir. Klasörün bulunduğu dizin bilgisayarın ortam değişkenlerine eklenir. ../bin dizini de kopyalanarak ortam değişkenlerine Path olarak eklenir. “mvn -version” komutu verilerek Maven versiyonu doğrulanır.

* 1. **IDE Kurulumu**

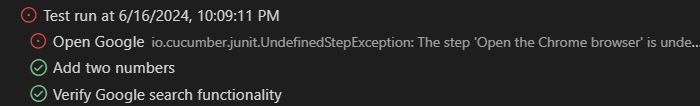
Projenin çalıştırılabilmesi için Java ile uyumlu bir IDE sağlanır. (Örneğin, VSCode, Intellij IDEA)

* 1. **Proje Bağımlılıkları**

Proje klasöründe bulunan pom.xml dosyası açılarak Selenium, Serenity, JUnit, Cucumber, RestAssured vb. bağımlılıkların bulunduğu doğrulanır. .xml dosyasına yeni bağımlılık eklenirse terminalde “mvn clean install” komutu çalıştırılır. “mvn test” komutu çalıştırılarak mevcuttaki (varsa) errorlar görülebilir.

1. **Projenin ve Testlerin Çalıştırılması**

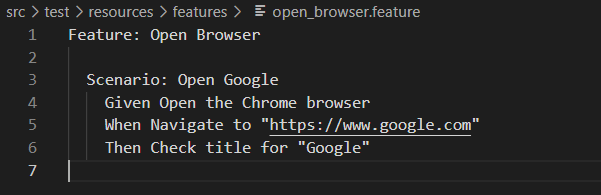
IDE’de proje klasörü açılarak src\main\java\frameworkproject altındaki MainClass.java çalıştırılır. Projede tanımlı test senaryolarının çalışması için ise src\test\java\frameworkproject altındaki RunTest.java üzerinde sağa tıklanır ve “RunTest”e tıklanır. Mevcut testlerin sonuçları(başarılı/başarısız) Terminal bölümünde “Test Results” altında görülür.



1. **Test Senaryolarının Oluşturulması**

**4.1. Gherkin İle Test Adımları Yazma**

Projeye yeni test senaryosu eklenmek istenirse src\test\resources\features dizini altına yeni bir .features dosyası eklemek gerekir. Aşağıdaki görsel üzerinden örnek vermek gerekirse “Feature” ve “Scenario” eklenerek yeni test senaryosuna isim verilmiş olur. ‘Given’, ‘When’, ‘Then’ keywordleri ise senaryoda uygulanması gereken adımları ve beklenen test sonuçlarını ifade eder. Örneğin burada verilen browser(Google) üzerinde google.com’a gidilerek başlığın Google olup olmadığı kontrol edilir.



Başka bir örnek vermek gerekirse:

*Feature: Scroll on Wikipedia Page*

*Scenario: Open Chrome browser and scroll on the Maldives Wikipedia page*

*Given a Chrome browser is opened*

*When the User navigates to "*[*https://en.wikipedia.org/wiki/Maldives*](https://en.wikipedia.org/wiki/Maldives)*"*

*And the User scrolls down the page*

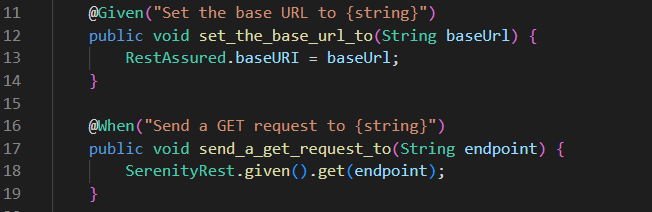
*Then the User does not close the browser*

şeklinde bir senaryo Gherkin dilinde yazılabilir.

Gherkin senaryolarının ‘Given’, ‘When’, ‘Then’ keywordleri ve projeye uygun cümle yapısı kullanılarak yazılması gerekmektedir. StepDefinition classlarında bulunan methodlar senaryo içerisinde yazılan cümleleri karşılamalıdır.

**4.2. StepDefinition Classlarının Genişletilmesi**

src\test\java\frameworkproject\stepdefinitions dizini altında bulunan classlar .feature dosyalarındaki senaryoların otomatik olarak koşulması için yazılmıştır. Gherkin ile yazılan her bir test adımı için ayrı bir method oluşturulabilir veya dinamik bir yapı kullanılarak birden fazla test adımı/senaryo için aynı methodların çağrılması sağlanabilir. Örneğin aşağıdaki görselde senaryodan iletilen Url parametresi farklı değerler içererek senaryoda belirtilen Url’lere göre farklı websitelerine yönlendirme sağlanmış olur.



Eklenebilecek farklı .feature dosyaları(veya dosya içerisindeki stepler) için eğer mevcut methodlar gerekli dinamizmi karşılamıyorsa stepdefinition classlarına uygun methodların eklenmesi gerekmektedir.