Testprojekt 3

# Rahmenbedingungen

* **Abgabetermin:**

30.06.2024, 12:00 Uhr.

* **Umfang der Abgabe:**
* Meldung per E-Mail an [olivia.macolic@dsgf.de](mailto:olivia.macolic@dsgf.de), danach sind keine Anpassungen am Portfolio vorgesehen.
* 30 Min Präsentation der Ergebnisse mit Fragerunde
* **Rückfragen:** 
  + Rückfragen per E-Mail an [olivia.macolic@dsgf.de](mailto:olivia.macolic@dsgf.de) senden.
  + Je nach Bedarf wird ein gemeinsamer Termin eingestellt.

# Aufgabe

**Automation** und **Dokumentation** von **Einkaufprozessen im Online-Shop "Swag Labs"** mittels relevanter Werkzeuge. Diese Aufgabe führt durch alle für die Aufgabe relevanten Werkzeuge.

## **Schritt: Für die Aufgabe müssen folgende Vorbereitungen getroffen werden:**

* Installation von Python, Robotframework und Selenium (und anderer Bibliotheken falls nötig) **Ѵ**
* Installation von VSCode **Ѵ**
* Einrichtung von Geckodriver **Ѵ**
* Installation und Einrichtung von Docker oder Podman
* Aufsetzen einer SQL-Datenbank in Docker oder Podman inklusive Webinterface

## **Schritt: Erstelle mit Robotframework folgenden Test:**

* Besuche die Seite <https://www.saucedemo.com/> **Ѵ**
* Führe alle möglichen Login Kombinationen durch
* Führe mit jedem User einen Kauf durch
* Nach erfolgreichem Kauf sollte der Produktname, sowie der Preis des Produktes und der entsprechende User in der SQL-Datenbank gespeichert werden

## **Hinweise zur Umsetzung:**

* Sollte ein Einkauf nicht erfolgreich sein, sollte dies ebenfalls in der Tabelle ersichtlich sein
* Schreibe die entsprechenden Keywords um Daten in der SQL-Datenbank zu speichern selber in Python (Nicht die vorhandene Database Library dafür verwenden)
* Verwende für den Robotframework Test den Datadriven Ansatz
* Verwende zum interagieren mit dem Browser die SeleniumLibrary
* Dokumentiere deine Schritte in einem Git Repository

# Ergebnisse

* Robotframework Bericht und LOG im Git
* Sourcefiles im Git
* Dokumentation über Installation und Konfigurationsschritte der verwendeten Tools im Git
* Präsentation in Teams

**Robot Framework – Wichtige Grundlagen & Struktur**

**🐍 Python**

* Grundlage für Robot Framework – läuft komplett in Python.
* Installation über [python.org](https://python.org) und Paketmanager pip.

**🤖 Robot Framework**

* Das eigentliche Testframework für automatisierte Tests.
* Installation: pip install robotframework
* Arbeitet mit speziellen .robot-Dateien für Tests.

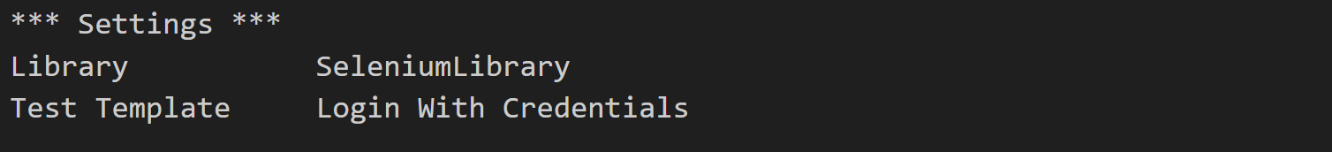
**🌐 SeleniumLibrary**

* Robot Framework Library, um echte Browser zu steuern (z. B. Firefox, Chrome).
* Ermöglicht Aktionen wie:
  + Elemente klicken
  + Texte eingeben
  + Seiteninhalte prüfen

**🔧 Geckodriver**

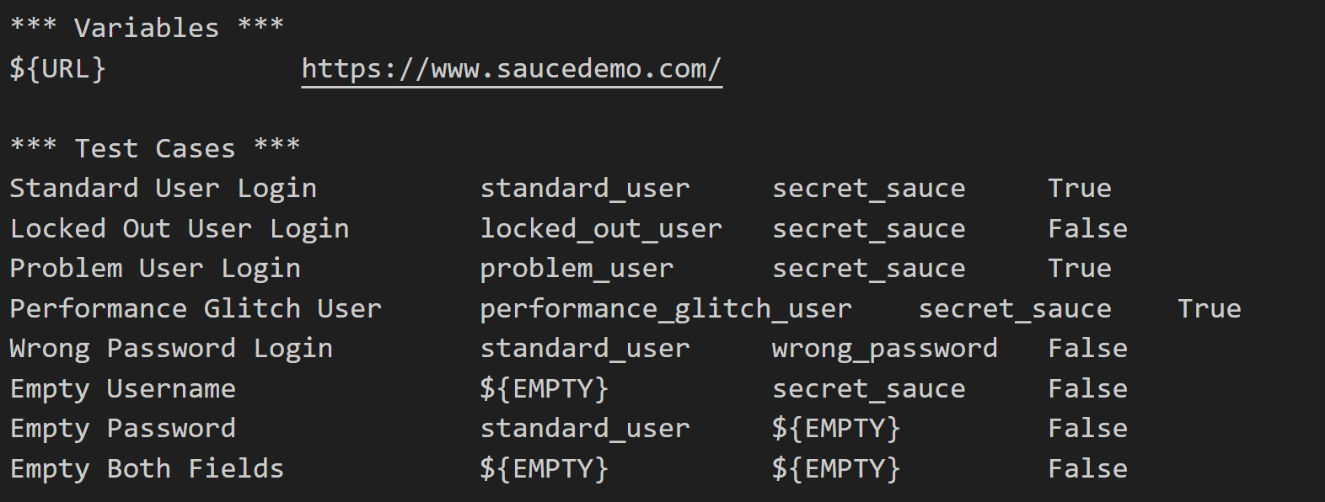
* „Brücke“ zwischen SeleniumLibrary und Firefox.
* Steuert Firefox über Selenium.
* Muss separat heruntergeladen und im Systempfad oder an bekannter Stelle liegen.

**Aufbau einer Robot Framework Datei (.robot)**

****

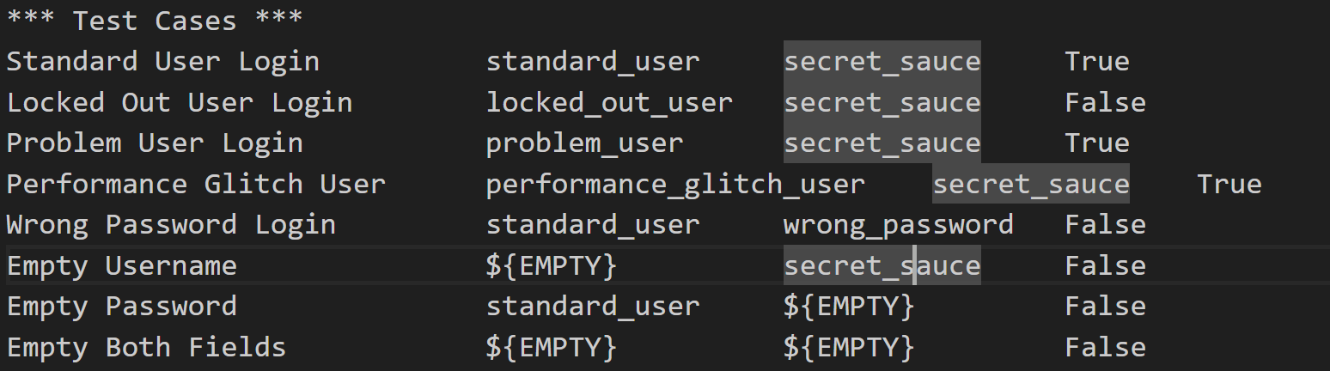
Lädt die benötigten Libraries.

Definiert „Test Templates“, damit Testfälle eine Vorlage (Keyword) gemeinsam nutzen.



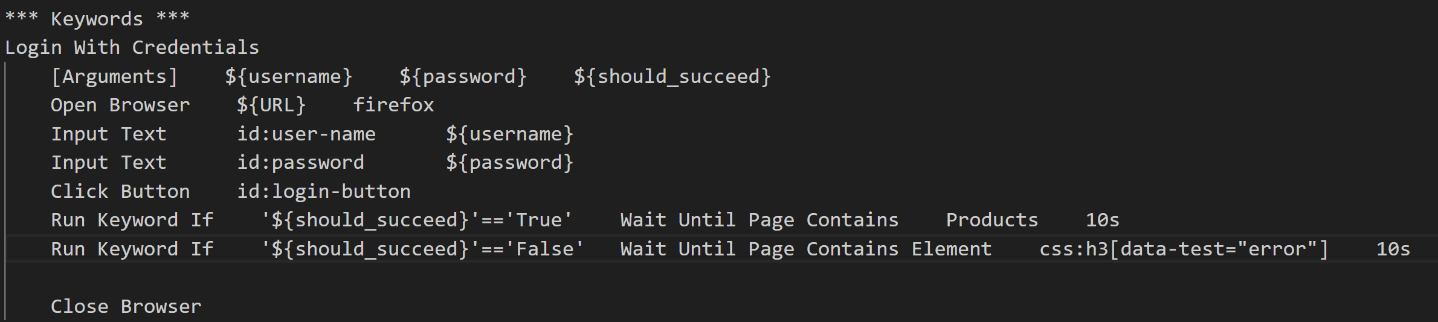
Definiert Variablen für wiederverwendbare Werte (z. B. URLs).

Sauberer, weil Änderungen zentral möglich sind.



Hier stehen die einzelnen Testfälle.

Jede Zeile ruft ein Keyword (Test Template) mit Parametern auf, z. B. Benutzername, Passwort, erwartetes Ergebnis.



Hier sind wiederverwendbare Befehle definiert (ähnlich wie Funktionen).

In diesem Beispiel:

Browser öffnen, Login-Daten eingeben, Login ausführen

Prüfen, ob Login erfolgreich war oder Fehlermeldung angezeigt wird

Browser schließen

**Ablauf bei Testausführung**

Robot Framework parst die .robot-Datei.

Erkennt alle Testfälle unter \*\*\* Test Cases \*\*\*.

Führt jeden Testfall nacheinander aus.

Nutzt die Anweisungen aus \*\*\* Keywords \*\*\*.

Erstellt am Ende folgende Reports:

output.xml (technisch, für CI/CD)

log.html (Detaillierte Ansicht jedes Schritts)

report.html (Übersicht der Testergebnisse)

Demo-Webseite & Zugangsdaten

https://www.saucedemo.com/ ist eine öffentliche Demo-Website für UI-Tests.

**Bekannte Benutzeraccounts:**

|  |  |  |
| --- | --- | --- |
| **Benutzername** | **Passwort** | **Verhalten** |
| standard\_user | secret\_sauce | Erfolgreicher Login |
| locked\_out\_user | secret\_sauce | Login schlägt fehl (gesperrt) |
| problem\_user | secret\_sauce | Login ok, UI zeigt Fehler (visuelle Bugs) |
| performance\_glitch\_user | secret\_sauce | Login ok, aber sehr langsam |
| error\_user | secret\_sauce | Ungewöhnliche Bild-Fehler |
| visual\_user | secret\_sauce | optional, manchmal deaktiviert |

**Robot Framework Output Dateien**

**1. output.xml**

**Was ist das?**

* Die **Haupt-Ausgabedatei** im XML-Format.
* Enthält alle Informationen über den Testlauf, inklusive aller Schritte, Status, Zeitstempel, Fehler, Logs, Metadaten.

**Inhalt**

* Jede Test-Suite, jeder Testfall, jedes Keyword (Schritt) wird hier als XML-Element mit Attributen (z. B. Status, Dauer) gespeichert.
* Detailinformationen zu Fehlern, Warnungen, Logs.
* Eignet sich perfekt zur automatischen Weiterverarbeitung.

**Wie liest du sie?**

* Am besten mit speziellen Tools oder Robot Framework eigenen Werkzeugen:
  + z. B. rebot-Tool:

rebot output.xml  
erzeugt log.html und report.html aus der XML-Datei.

* Du kannst auch XML-Viewer verwenden, aber das ist meist unpraktisch.

**Wofür wird sie genutzt?**

* **CI/CD-Pipelines** (Jenkins, GitHub Actions, GitLab CI, etc.) lesen output.xml ein, um automatisiert Testergebnisse auszuwerten.
* Andere Tools können daraus Berichte generieren oder in Dashboards integrieren.
* Grundlage für log.html und report.html.

**2. log.html**

**Was ist das?**

* Detaillierter **Testlauf-Report** in HTML.
* Sehr gut für manuelle Analyse.
* Zeigt alle Schritte, Testfälle, Status und Zeitdauer.

**Inhalt**

* Übersicht aller Tests mit Status (Pass, Fail, Skip).
* Für jeden Testfall:
  + Detaillierte Schritte (Keywords)
  + Ausgeführte Befehle (z. B. Klicks, Texteingaben)
  + Erfolg/Misserfolg einzelner Schritte
  + Zeitstempel
  + Fehlermeldungen
* (Optional) Screenshots bei Fehlern, wenn aktiviert.
* Filtermöglichkeiten (z. B. nur Fehler anzeigen).

**Wie liest du sie?**

* Einfach im Browser öffnen.
* Interaktive Navigation durch Testlauf.
* Kann zum Debuggen genutzt werden:
  + Wo genau ist ein Test fehlgeschlagen?
  + Welche Schritte wurden ausgeführt?
  + Welche Fehlermeldung gab es?

**Wofür wird sie genutzt?**

* Entwickler, Tester und QA-Teams verwenden die Datei, um Testdurchläufe zu analysieren.
* Wichtig für Fehlersuche und Verbesserung der Tests.
* Im Team teilen oder archivieren.

**3. report.html**

**Was ist das?**

* Kompakte **Übersicht** aller Testergebnisse.
* Fokus auf Gesamtstatus, keine Schritt-für-Schritt-Details.

**Inhalt**

* Gesamtanzahl der Tests
* Anzahl der bestandenen / fehlgeschlagenen / übersprungenen Tests
* Gesamtlaufzeit
* Zusammenfassung auf Suite- und Testfall-Ebene

**Wie liest du sie?**

* Öffne die Datei im Browser.
* Sie zeigt auf einen Blick, wie erfolgreich der Testlauf war.
* Du siehst schnell, ob z. B. alle Tests durchgelaufen sind.

**Wofür wird sie genutzt?**

* Schneller Überblick für Manager, Stakeholder oder automatisierte Systeme.
* Meist erstes Ergebnis nach einem Testlauf, um zu entscheiden, ob alles OK ist.
* Kombiniert mit log.html für detaillierte Analysen.

**Typischer Workflow im Einsatz**

1. Du führst Tests lokal oder in der CI/CD-Umgebung mit robot tests.robot aus.
2. Robot Framework erzeugt output.xml.
3. Aus output.xml werden automatisch log.html und report.html generiert.
4. Du öffnest report.html, um eine schnelle Übersicht zu bekommen.
5. Bei Problemen öffnest du log.html, um den Fehler und den genauen Schritt zu sehen.
6. Für automatisierte Abläufe (z. B. Jenkins) wird output.xml eingelesen und ausgewertet, um Build-Ergebnisse anzuzeigen oder Benachrichtigungen zu triggern.

**🛒 Ablauf eines Einkaufs auf saucedemo.com:**

1. Login
2. Produktübersicht → "Add to cart" klicken
3. Klick auf Warenkorb (oben rechts)
4. Klick auf "Checkout"
5. Eingabe von Vorname, Nachname, PLZ
6. Klick auf "Continue"
7. Klick auf "Finish"
8. Bestätigung sehen (z. B. „THANK YOU FOR YOUR ORDER“)