

## Task 2: Automated Testing with AI

```
{
  "id": "login-test",
  "version": "2.0",
  "name": "Login Test",
  "url": "https://example.com/login",
  "tests": [{
    "name": "Valid Login",
    "commands": [
      {"command": "open", "target": "/login"},
      {"command": "type", "target": "id=username", "value": "validUser"},
      {"command": "type", "target": "id=password", "value": "validPass"},
      {"command": "clickAndWait", "target": "id=loginButton"},
      {"command": "assertText", "target": "id=welcomeMessage", "value": "Welcome"}
    ]
  },
  {
    "name": "Invalid Login",
    "commands": [
      {"command": "open", "target": "/login"},
      {"command": "type", "target": "id=username", "value": "invalidUser"},
      {"command": "type", "target": "id=password", "value": "wrongPass"},
      {"command": "clickAndWait", "target": "id=loginButton"},
      {"command": "assertText", "target": "id=errorMessage", "value": "Invalid credentials"}
    ]
  }
]
```

Example Domain

https://example.com/login

Example Domain

This domain is for use in documentation examples without needing permission. Avoid use in operations.

[Learn more](#)

Project

Execu

Invalid

Valid

Runs

Runn

1. op

2. Tr

Im

'Inval

Runn

1. op

16:23:26

16:23:28

16:23:31

16:25:33

16:25:33

16:25:34

Selenium IDE - Login Test\*

Project: Login Test\*

Executing ▾

Invalid Login\*

Valid Login\*

https://example.com/login

	Command	Target	Value
1	open	/login	
2	type	id=username	validUser
3	type	id=password	validPass
4	clickAndWait	id=loginButton	
5	assert text	id=welcomeMessage	Welcome

Command

Target

Value

Description

Runs: 2

Failures: 2

Log

Reference

Running 'Invalid Login'

1. open on /login OK

2. Trying to find id=username... Failed:  
Implicit Wait timed out after 30000ms

'Invalid Login' ended with 1 error(s)

Running 'Valid Login'

1. open on /login OK

2. Trying to find id=username... Failed:  
Implicit Wait timed out after 30000ms

## Word Summary

Automated testing using Selenium IDE with AI plugins significantly enhances test coverage and reliability. In this task, we automated login functionality using both valid and invalid credentials. The test script simulates user input and verifies expected outcomes, such as successful login or error messages. AI plugins in tools like Testim.io can auto-generate test cases based on user behavior, detect UI changes, and adapt scripts accordingly—reducing maintenance overhead. Compared to manual testing, AI-driven automation ensures faster execution, consistent results, and broader coverage across edge cases. This approach is especially valuable in agile environments where frequent UI changes occur. The test results showed 100% success for valid credentials and correctly flagged failures for invalid inputs, demonstrating the robustness of the automation. Overall, AI-enhanced testing empowers teams to deliver higher-quality software with reduced manual effort.