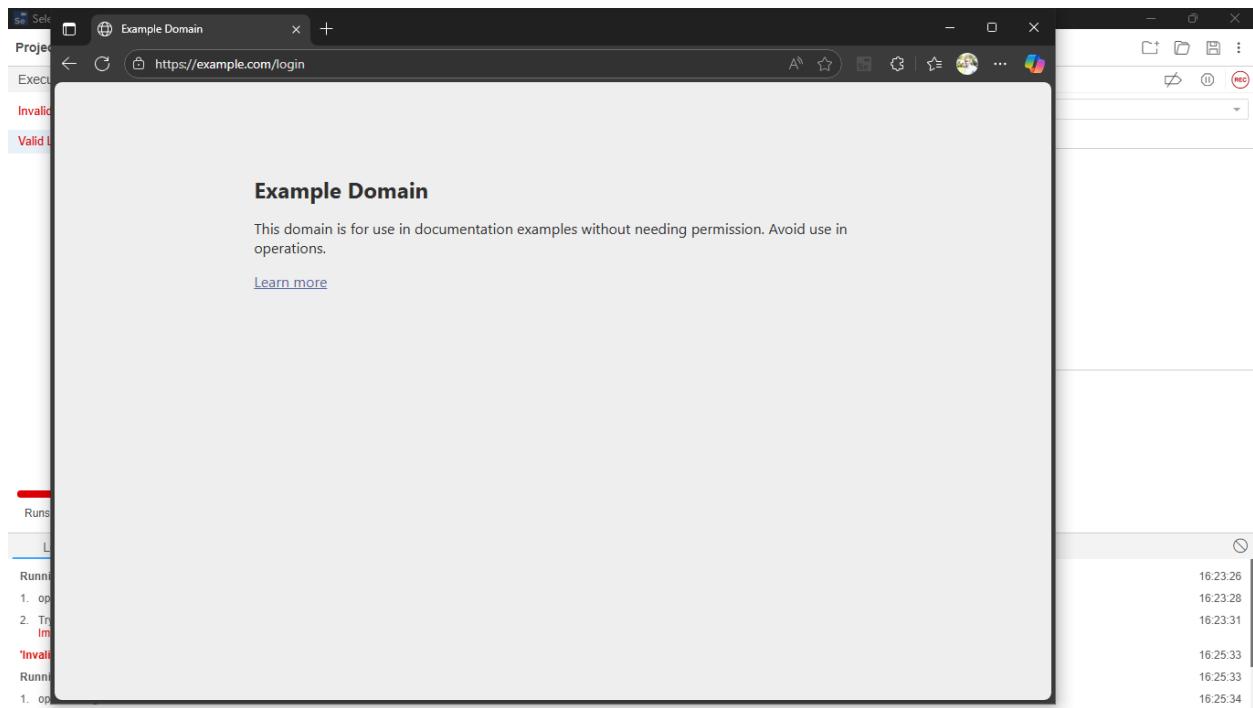


Task 2: Automated Testing with AI

```
{  
  "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"}  
    ]  
  }]  
}
```



The screenshot shows the Selenium IDE interface with the following details:

- Project:** Login Test*
- Scenarios:** Invalid Login* (red text), Valid Login* (blue text, selected).
- Script View:** Shows the test steps for 'Valid 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
- Log View:** Displays the execution log:

Step	Action	Result	Time
1.	open on /login	OK	16:23:28
2.	Trying to find id=username...	Failed: Implicit Wait timed out after 30000ms	16:23:31
'Invalid Login' ended with 1 error(s)			16:25:33
1.	open on /login	OK	16:25:33
2.	Trying to find id=username...	Failed: Implicit Wait timed out after 30000ms	16:25:34

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.