AI improves test coverage over manual testing in several key ways:

- **Smart Element Locators**:
Traditional Selenium scripts rely on fixed locators like IDs, XPaths, or class names. If the UI changes, these locators often break. AI-powered plugins or frameworks built on top of Selenium (e.g., Testim, Mabl, or Applitools) use smart, context-aware locators that adapt to changes in layout, making tests more robust and reducing maintenance.

- **Automatic Test Maintenance**:
AI can detect and fix broken selectors or suggest updates when the UI changes, something that would require manual debugging in plain Selenium. This helps maintain test coverage even as the application evolves.

- **Test Case Optimization**:
AI can analyze which parts of the application are most frequently used or recently changed and suggest or generate test cases accordingly. With plain Selenium, a tester must manually write and prioritize all tests.

- **Visual Testing Integration**:
Tools like Applitools integrate with Selenium and use AI to detect visual differences across browsers or versions that manual testers may miss or ignore.

- **Increased Speed and Scalability**:
While manual testing is slow and limited, AI combined with Selenium can scale test runs across multiple devices, browsers, and test scenarios simultaneously — ensuring broader and faster coverage.

Summary:

In this task, I used Selenium WebDriver to automate a login test case for the Power Learn Project platform. The test simulates a user attempting to log in with invalid credentials and checks whether the application correctly rejects the login attempt. The script fills in the login form, submits it, and verifies the appearance of an error message indicating failed authentication. A screenshot of the result is captured for evidence.

Compared to manual testing, AI-assisted tools such as Testim.io or Selenium enhanced with smart locators improve testing by adapting to dynamic UI changes, reducing script maintenance, and prioritizing test cases based on risk or change frequency. While my script used traditional element selectors, AI-based solutions can automatically update test steps when the interface changes, enabling continuous testing in fast-paced development environments. Automated testing improves efficiency, reduces human error, and ensures consistent validation of core functionalities like authentication.