



# USING AI FOR CONTINUOUS TESTING

How AI Fits Into Your SDLC

# Why CI/CD?



## 208

**TIMES MORE**

frequent code deployments



## 106

**TIMES FASTER**

lead time from commit to deploy



## 2,604

**TIMES FASTER**

time to recover from incidents



## 7

**TIMES LOWER**

change failure rate  
(changes are 1/7 as likely to fail)

## Continuous Testing Challenges

- Not feasible to execute entire test suite
- Testing in production
- Need to build automation fast
- Need reliable, smarter automation
- Need more testing across the board

### Traditional Automation

- Expensive
- Lacks generality
- High maintenance
- Semi-automated
- Build model, generate tests. Generality?
- Model Maintenance

```
@Test
public void successLoginSampleTest() {
    product.login.userNameTextBox
        .set("user@test.com");

    product.login.passwordTextBox
        .set("testp@ssword");

    product.login.loginButton.click();
}

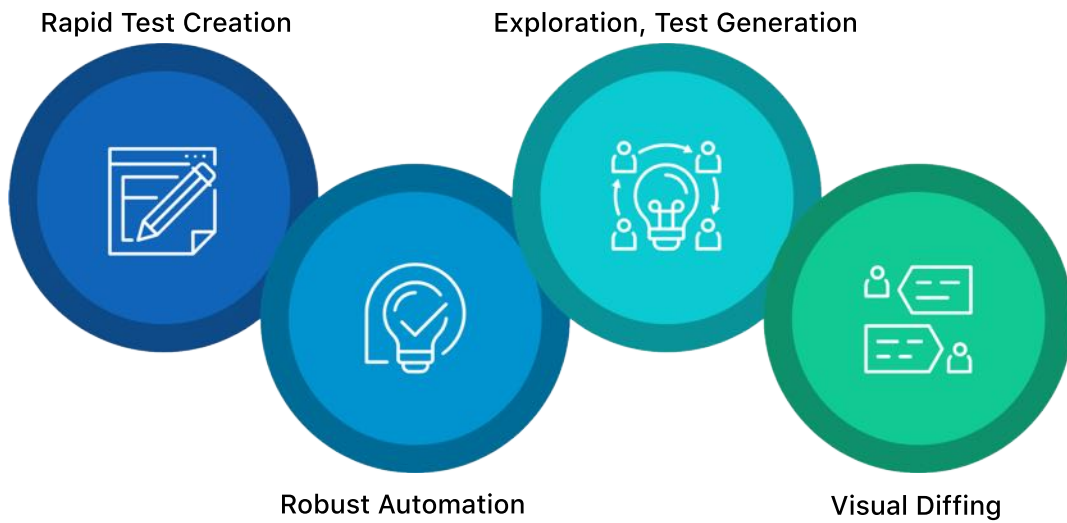
public class LoginPage extends Page {
    public TextBox userNameTextBox;
    public TextBox passwordTextBox;
    public Button loginButton;

    public LoginPage(AutomationInfo automationInfo) {
        userNameTextBox = new TextBox(automationInfo,
            By.cssSelector("input#email"));

        passwordTextBox = new TextBox(automationInfo,
            By.cssSelector("input#password"));

        loginButton = new Button(automationInfo,
            By.cssSelector("button#login"));
    }
}
```

# Benefits of AI-Powered Automation



## 1. Rapid Test Creation: Drag and Drop

This block compares manual test creation with AI-powered test creation. On the left, a screenshot shows a manual drag-and-drop interface for creating test steps, with a search bar and a list of elements. In the center, a screenshot shows a web application interface with a search bar and a list of products. On the right, a screenshot shows a test runner interface with a search bar and a list of elements. To the right of the central screenshot, the text 'VS.' is displayed. On the far right, two code snippets are shown. The top snippet is a manual test script in Java, and the bottom snippet is an AI-generated test script in Java.

```
@Test
public void successLoginSampleTest() {
    product.login.userNameTextBox
        .set("user@test.com");

    product.login.passwordTextBox
        .set("testpassword");

    product.login.loginButton.click();
}

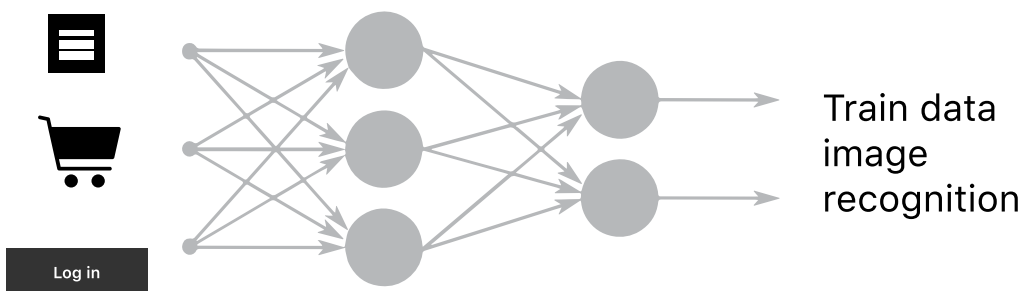
public class LoginPage extends Page {
    public TextBox userNameTextBox;
    public TextBox passwordTextBox;
    public Button loginButton;

    public LoginPage(AutomationInfo automationInfo) {
        userNameTextBox = new TextBox(automationInfo,
            By.cssSelector("input#email"));

        passwordTextBox = new TextBox(automationInfo,
            By.cssSelector("input#password"));

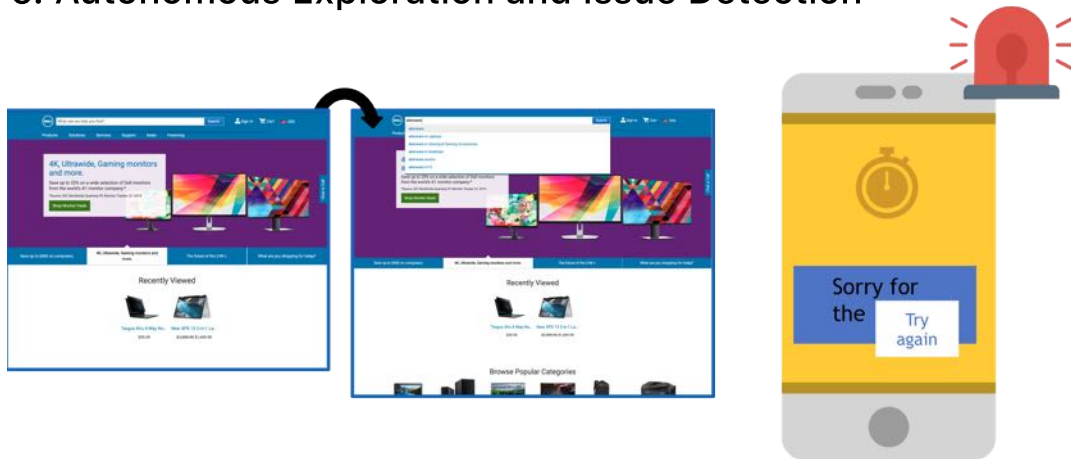
        loginButton = new Button(automationInfo,
            By.cssSelector("button#login"));
    }
}
```

## 2. Robust Test Automation: Self Healing AI Replaces Selectors



# Benefits of AI-Powered Automation

3. Self-healing test scripts that are resilient to styling changes to SUT
4. Test scripts are reusable across apps
5. Autonomous Exploration and Issue Detection



## 6. AI: Visual Diffing

- Continuous testing in production
- Automatic comparisons
- Difference reports
- Approve/Deny
- Reinforce AI

