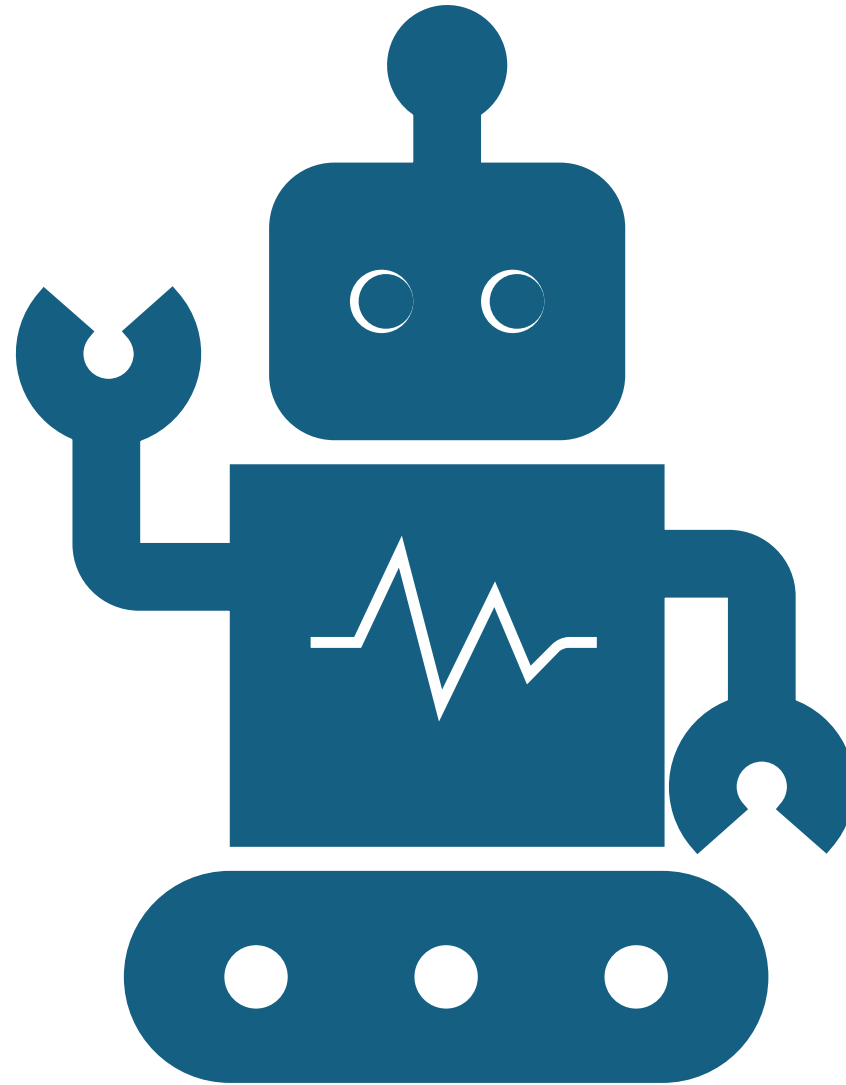


GitHub Copilot For Automation Scripting – Selenium Java

Author- Rahul Ritesh
Test Automation Lead



About Me

I am working as a Test Automation Lead with 9 years of experience in automation engineering across Banking, Payments, E-commerce, Salesforce CRM, and SAP ERP domains.





Agenda


- Introduction to GitHub Copilot
- How LLM Works
- Benefits of GitHub Copilot in Automation
- Setting Up GitHub Copilot in IDEs
- Utilizing GitHub Copilot for Automation Scripting
 - Creation Feature Files from Test Cases
 - Creation of Page Locators and Page Actions class
 - Creation of Step Definitions class
 - Creation of Test Runner Class
 - Creation of suite xml file
- Error Fixing and Code Optimization Features
- Limitations and Best Practices




GitHub Copilot setup in IDE

-  **Step 1: Sign in to GitHub**
- Go to <https://github.com>.
- If you already have a GitHub account, click **Sign In**.
- If not, click **Sign Up** and create a new account by following the prompts.

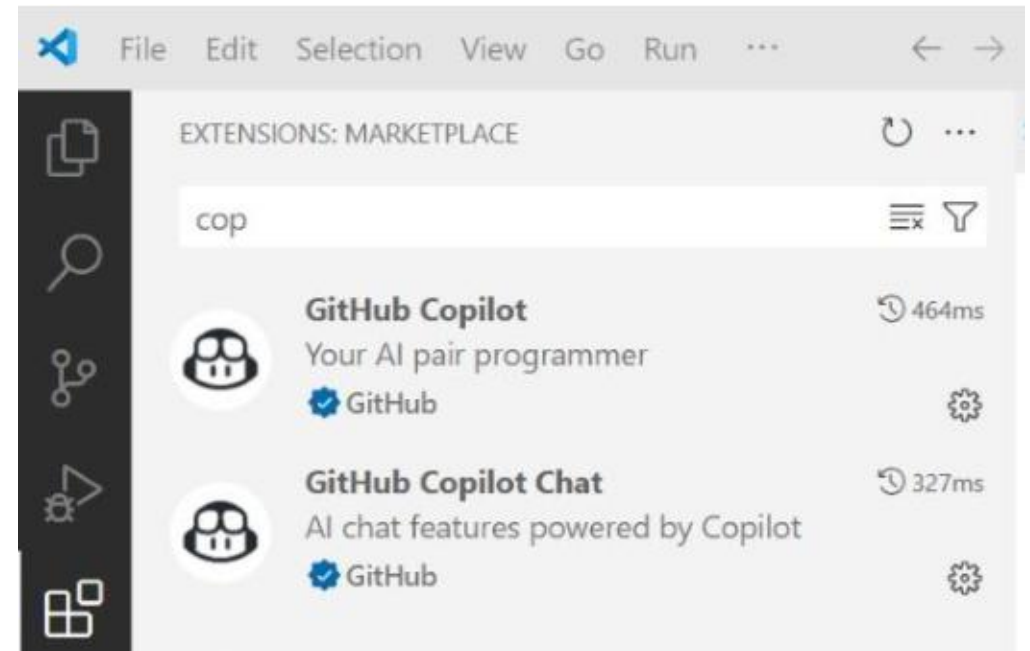
-  **Step 2: Navigate to GitHub Copilot**
- Once signed in, go to <https://github.com/features/copilot>.
- Click on the “**Start my free trial**” or “**Get Copilot**” button.

-  **Step 3: Choose a Plan**
- GitHub Copilot offers:
 - A **free trial** (usually 30 days).
 - A **monthly or yearly subscription** for individuals.
 - A **free plan for verified students and open-source contributors**.
- Select the plan that suits you and click **Continue**.

-  **Step 4: Confirm Billing Information**
- Enter your **payment details** (even for the free trial).
- Review and confirm your subscription.

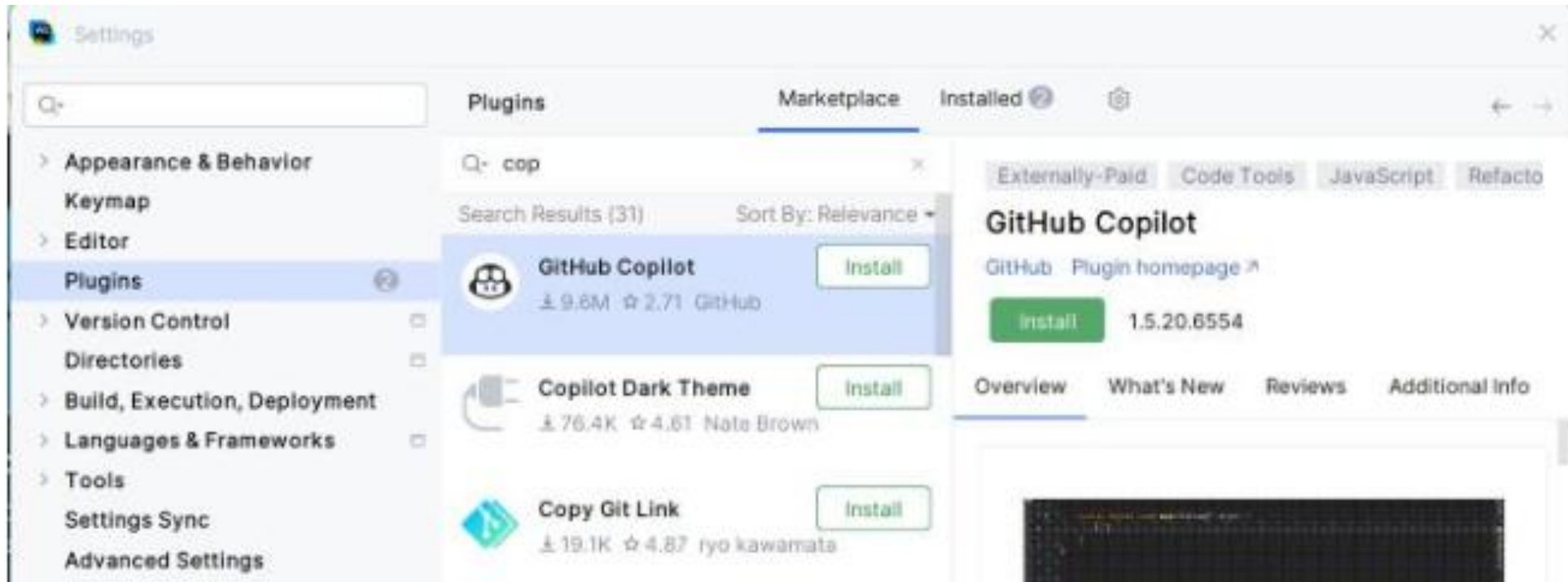
Adding GitHub Copilot in VS Code IDE

- Step 1: Sign Up For GitHub Account and subscribe for the license
- Step 2: Enable GitHub Copilot in IDE by adding extensions “GitHub Copilot” and “GitHub Copilot Chat” .



Setup in IntelliJ IDE

- You can set up GitHub Copilot in JetBrains IDE by installing the Copilot plugin from the JetBrains marketplace.
- Before that make sure that you have an active GitHub account with copilot subscription.



Foundation of GitHub Copilot

Version Control System

What is a Version Control System (VCS)?

Imagine you're writing a document or working on a project with your friends. You make changes, they make changes, and soon it becomes hard to keep track of who did what and when. A **Version Control System** is like a **smart time machine** for your files. It:

Keeps track of every change made to your files.

Lets you go back to an earlier version if something breaks.

Helps multiple people work on the same project without messing up each other's work.

GitHub Vs GIT

What is GitHub?

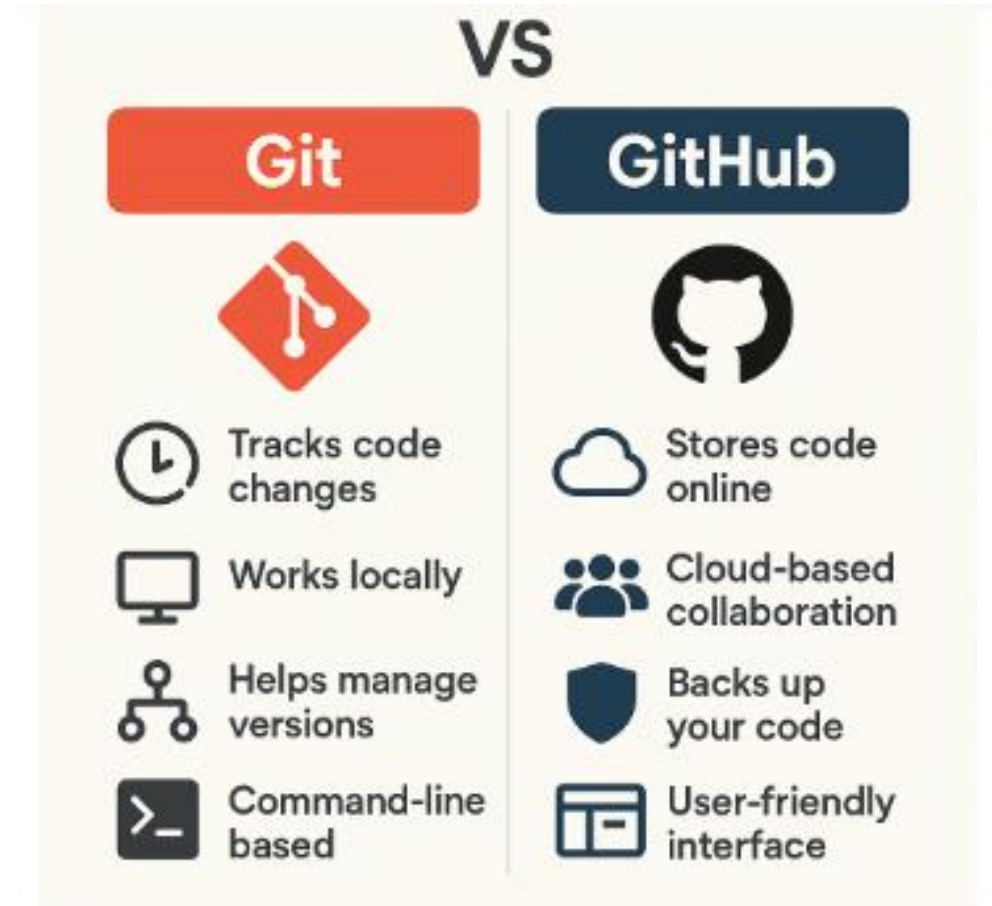
Think of **GitHub** as a **social media platform for code**.

- Just like you upload photos to Instagram or videos to YouTube, developers upload their **code projects** to GitHub.
- It's built on top of **Git** (the version control system we talked about earlier), but adds a **web-based interface** to make collaboration easier.

What is Git?

Git is one of the most popular version control systems. Think of it as:

- A **super-organized notebook** that remembers every change you make to your project.
- It lets you **save snapshots** of your work (called *commits*).
- You can **branch off** to try new ideas without affecting the main project.
- And when you're ready, you can **merge** your changes back in.
- Git is especially useful for developers working in teams, but even solo users benefit from its ability to track progress and undo mistakes.



Understanding GitHub Copilot

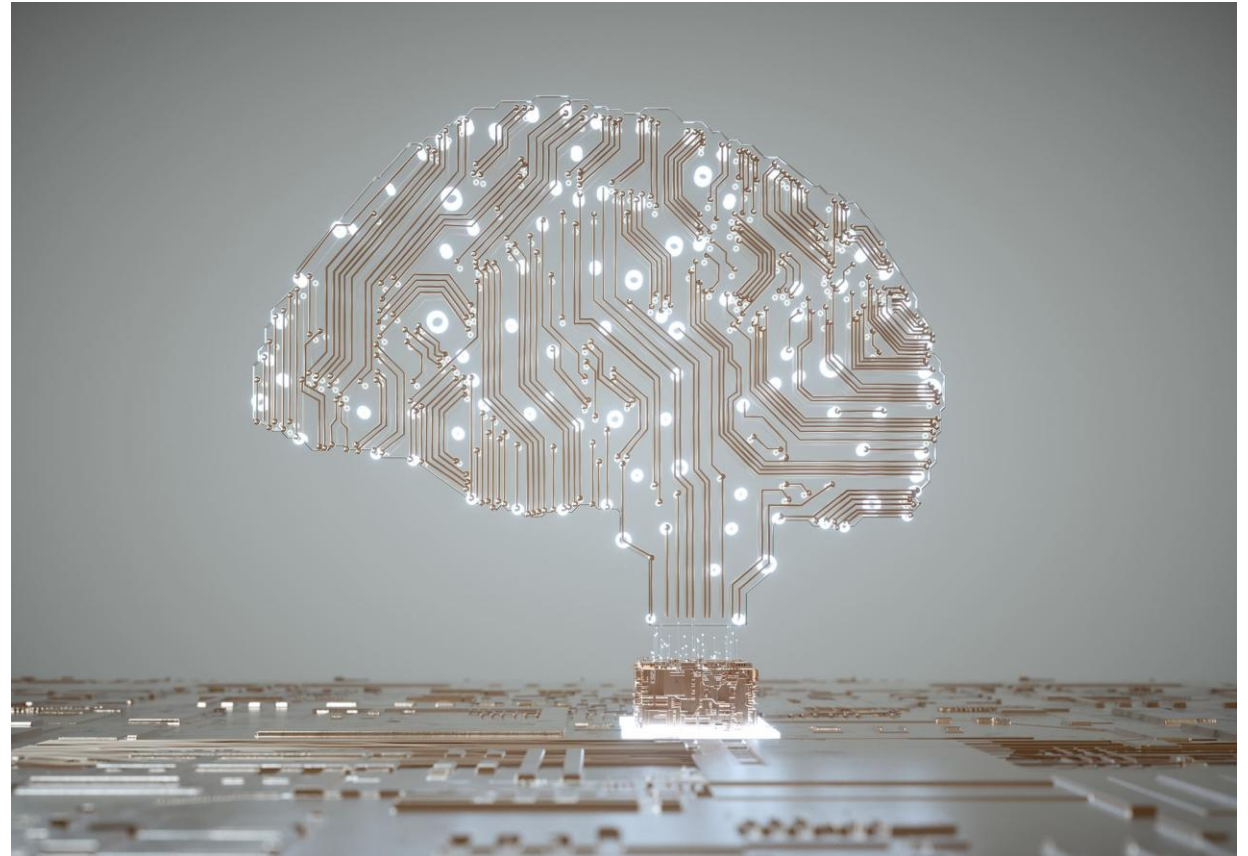
GitHub Platform: GitHub is an online service where developers can save, organize, and share their code. Think of it as a digital library for code.

Version Control with Git: GitHub uses a tool called Git to keep track of changes made to the code. This helps developers work together, go back to previous versions if needed, and manage their projects efficiently.

GitHub Copilot: GitHub Copilot is like a smart assistant for developers. It uses advanced AI technology to help write code in various programming languages found on GitHub.

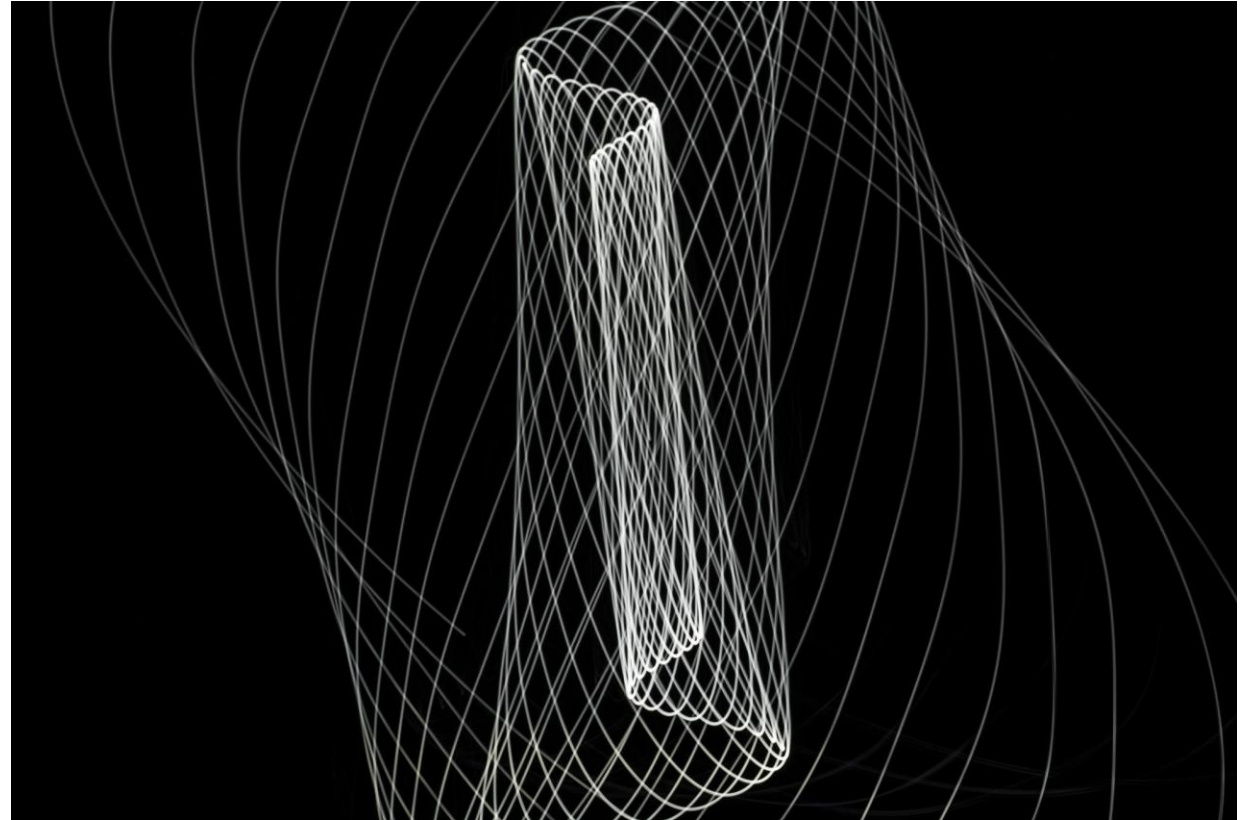
Understanding GitHub Copilot's Functionality

- GitHub Copilot is an AI-powered coding assistant that helps developers write code faster and smarter — right inside their code editor.
- It uses a large language model (LLM) to generate code suggestions.
- The model is trained on a diverse dataset of code from GitHub.
- Users can interact with Copilot in their IDE to receive real-time coding help.
- The latest version of the model enhances accuracy and code relevance.



Understanding LLM in GitHub Copilot

- It is trained on billions of lines of public code.
- Programming documentation enhances its learning.
- Natural language inputs improve code generation accuracy.
- LLMs help in understanding comments and instructions effectively.



How LLM Enhances Coding with GitHub Copilot

- Predicts and completes code as you start typing.
- Transforms comments into functional code effortlessly.
- Suggests complete implementations for requested functions.
- Provides fixes or alternatives when mistakes are made.



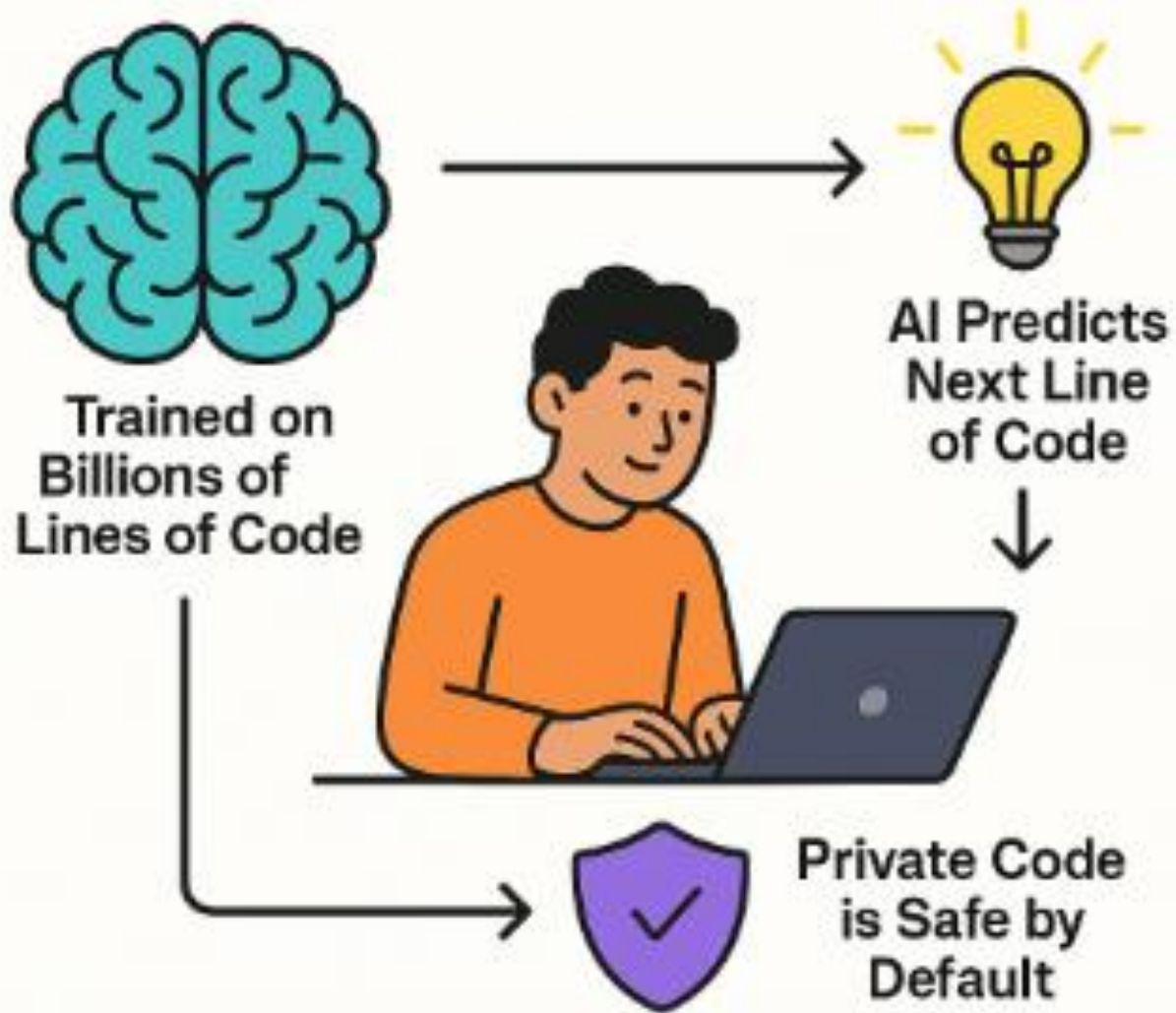
The Power of GitHub Copilot

- Understands both natural language and code effectively.
- Learns from the context present in your code files.
- Supports a wide range of programming languages.
- Saves developers time by reducing repetitive boilerplate code.



GitHub Copilot - Working

How GitHub Copilot Works



🧠 How Does GitHub Copilot Learn from Code?

GitHub Copilot is powered by **AI**, specifically a large language model developed by OpenAI. Here's how it learns:

📁 1. It Was Trained on Public Code

- Copilot was trained on **billions of lines of public code** from GitHub and other sources.
- It looked at how developers write code, solve problems, and structure projects.
- Just like how you learn a language by reading books, Copilot learned **coding patterns** by reading code.

2. It Recognizes Patterns

- It doesn't memorize exact code like a copy-paste tool.
- Instead, it **learns patterns**—like how loops, functions, and conditions are usually written.
- So when you start typing, it predicts what you're likely to write next based on those patterns.

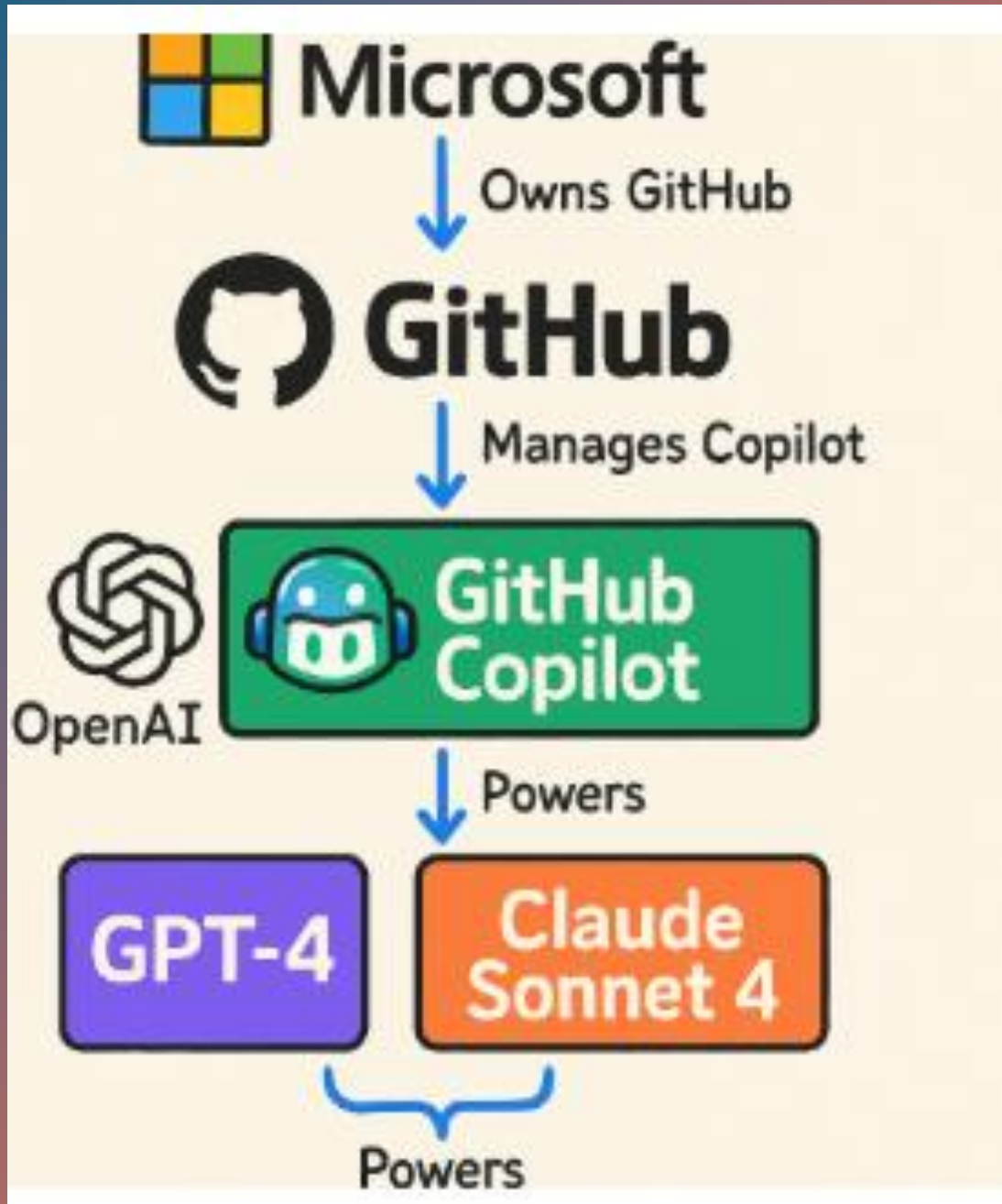
3. It Works Like Smart Autocomplete

- As you type, Copilot looks at your code and tries to **guess what you need next**.
- It uses its training to suggest code that fits your context—like a smart coding buddy who's read millions of examples.

4. It Doesn't Learn From Your Code in Real-Time

- Copilot doesn't learn from your private code unless you explicitly allow it.
- It uses what it already learned during training to help you write better code.

Interesting Facts about GitHub Copilot



As of 2025, **GitHub Copilot** is owned and operated by **GitHub**, which is a **subsidiary of Microsoft**

. The AI behind Copilot was developed in collaboration with **OpenAI**, but GitHub manages the product, its integration into development tools, and its commercial offerings. So in simple terms:

- **Microsoft** owns **GitHub**.
- **GitHub** owns and runs **Copilot**.
- **OpenAI** helped build the AI brains behind it.

As of 2025, GitHub Copilot uses multiple large language models (LLMs) to power its features. These include:

Claude Sonnet 4 and Claude Opus 4 by Anthropic – These are advanced models known for their strong reasoning and coding capabilities. Claude Sonnet 4 is optimized for coding workflows, while Opus 4 is more powerful and used in enterprise-level Copilot plans 1.

OpenAI models – GitHub Copilot originally launched using OpenAI's Codex model and continues to support newer OpenAI models like gpt-4 and gpt-4-turbo in some versions 2.

Gemini 1.5 Pro by Google DeepMind – Also supported in GitHub Copilot's multi-model setup 2.

This multi-model approach allows GitHub Copilot to offer more flexibility and better performance depending on the task and user plan.

Analogy:

- Think of Copilot like a chef who's read thousands of recipes. When you start cooking, they can suggest the next ingredient or step—not because they memorized your recipe, but because they've seen so many similar ones before.

Value Add by GitHub Copilot

Benefits

Generates code snippets quickly, saving time and effort in writing repetitive or boilerplate code.

Assists in solving complex coding problems by suggesting solutions or alternative approaches.

Provides code explanation, generate docs, code suggestions, helps in code fixing and debugging

Suggests optimized and clean code, helping developers write better and more efficient code.

Helps catch mistakes early by suggesting corrections or improvements in real-time.

Can be easily integrated with IDEs like VS code, IntelliJ etc.

GITHUB COPILOT



Saves Time



Boosts Productivity



Great for Learning



Reduces Mental Load



Works with Many



Helps with Documenta-

Effort Savings with GitHub Copilot



Developers can save up to 30% of coding time using GitHub Copilot.



Average code writing speed increases from 20 lines/hour to 40 lines/hour.



Eliminating repetitive tasks reduces burnout and increases productivity.



Real-time suggestions enhance coding accuracy, lowering debugging time by 25%.



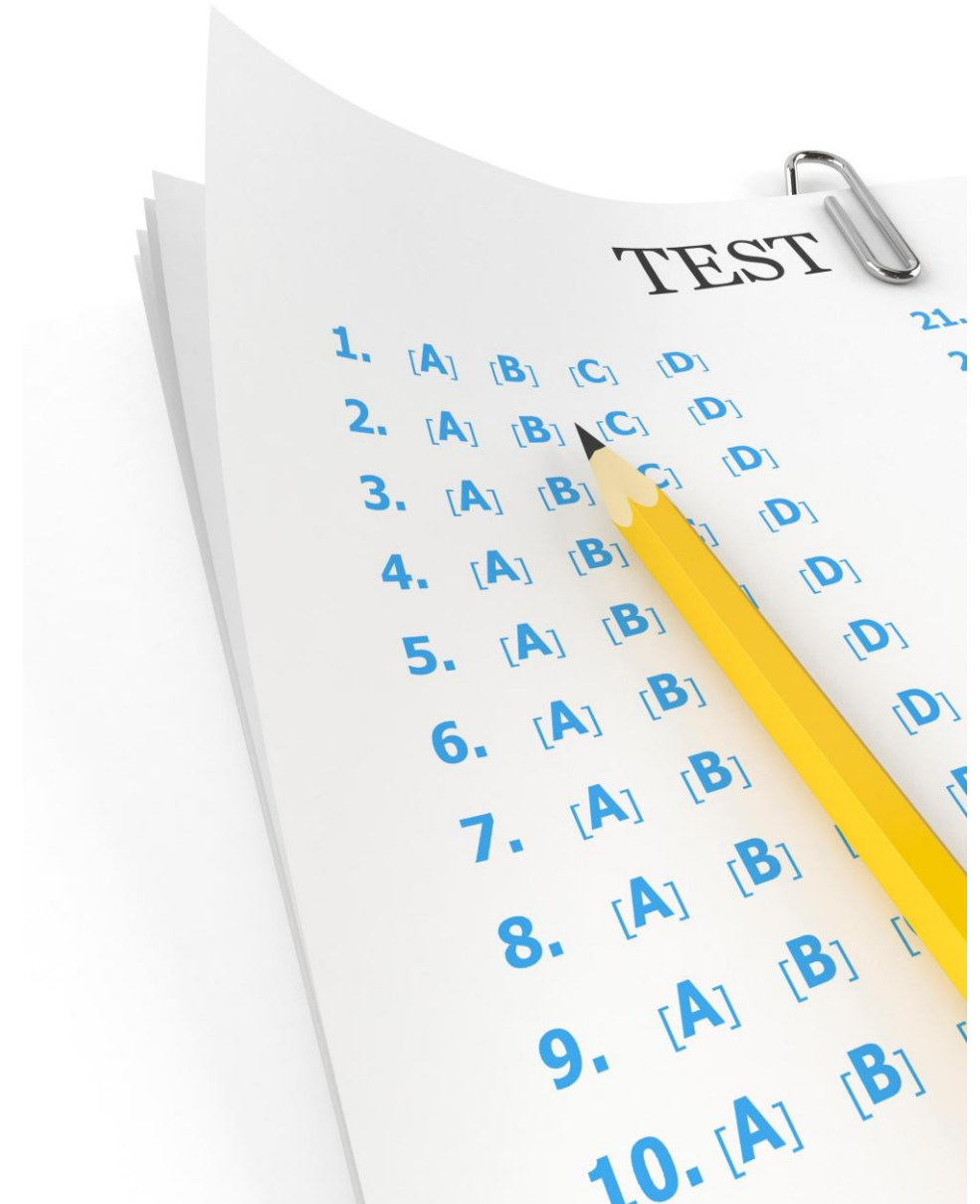
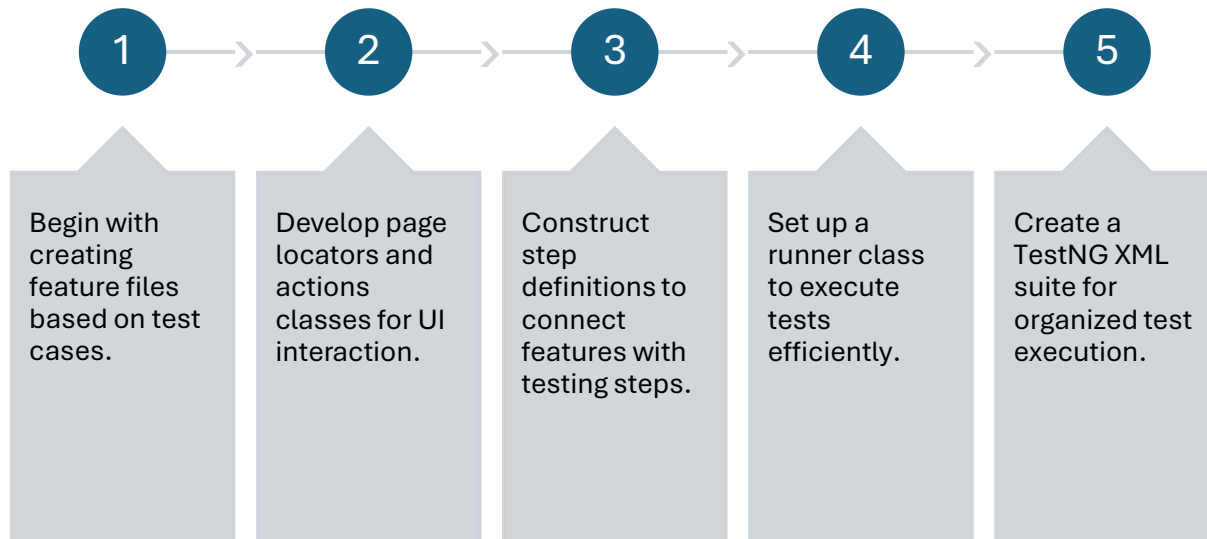
Graph shows improvement in coding efficiency over time.

Approximate Time Savings:

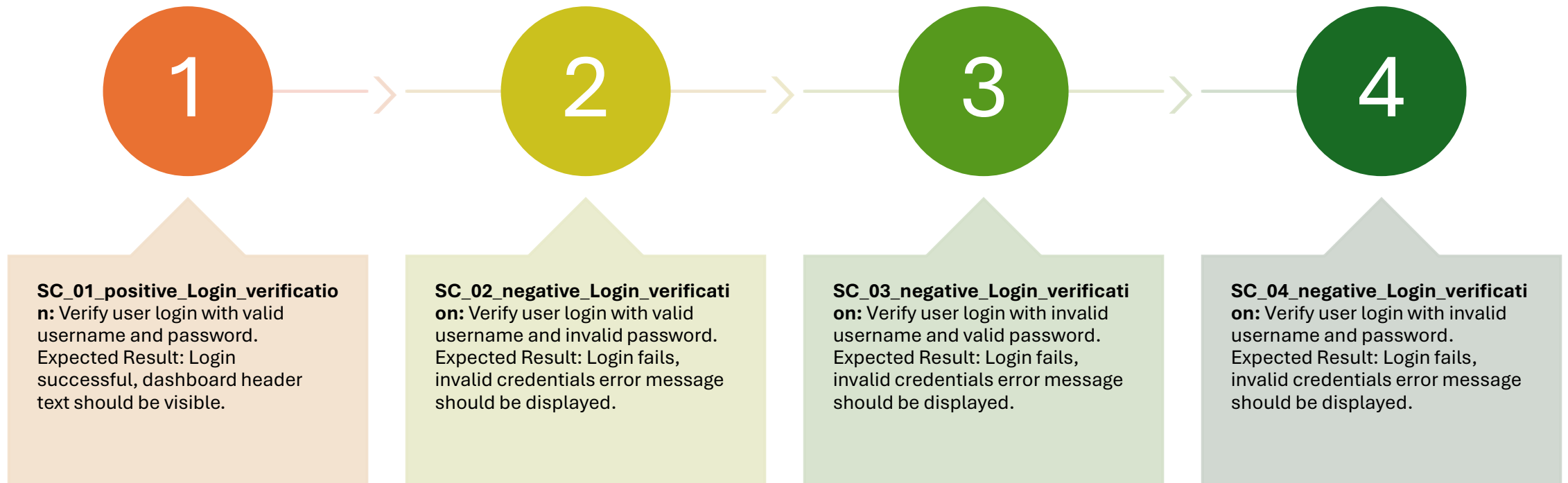
- **Writing Code:** 20% time saved
- **Debugging:** 15% time saved
- **Generating Documentation:** 25% time saved

Scripts Development Process – Practical Demonstration

Automating Test Development Process Using GitHub Copilot



Test Scenarios



Test Data

valid_username= "Admin"

valid_password="admin123"

invalid_username="MyAdmin"

invalid_password="admin1234"

Locators



username_Textbox_locator=//input[@placeholder='Username']



password_Textbox_locator=//input[@placeholder='Password']



login_button_locator=//button[@type='submit']



dashboard_headerText=//h6[@class='oxd-text oxd-text--h6 oxd-topbar-header-breadcrumb-module']



invalidCredentials_errorMessage=//p[@class='oxd-text oxd-text--p oxd-alert-content-text']

Feature File Output

Feature: Login Page Verification

Background:

Given the user navigates to the application URL "<https://opensource-demo.orangehrmlive.com/web/index.php/auth/login>"

Scenario Outline: Verify user login functionality

Given the user enters username "<username>" in the username textbox

And the user enters password "<password>" in the password textbox

When the user clicks on the login button

Then <expected_result>

Examples:

username	password	expected_result
Admin	admin123	the user should be logged in successfully
Admin	admin1234	the login should fail
MyAdmin	admin123	the login should fail
MyAdmin	admin1234	the login should fail

Scenario: SC_01_positive_Login_verification

Given the user enters valid username "Admin" in the username textbox

And the user enters valid password "admin123" in the password textbox

When the user clicks on the login button

Then the user should be logged in successfully

And the dashboard header text should be visible

Scenario: SC_02_negative_Login_verification

Given the user enters valid username "Admin" in the username textbox

And the user enters invalid password "admin1234" in the password textbox

When the user clicks on the login button

Then the login should fail

And the invalid credentials error message should be displayed

Scenario: SC_03_negative_Login_verification

Given the user enters invalid username "MyAdmin" in the username textbox

And the user enters valid password "admin123" in the password textbox

When the user clicks on the login button

Then the login should fail

And the invalid credentials error message should be displayed

Scenario: SC_04_negative_Login_verification

Given the user enters invalid username "MyAdmin" in the username textbox

And the user enters invalid password "admin1234" in the password textbox

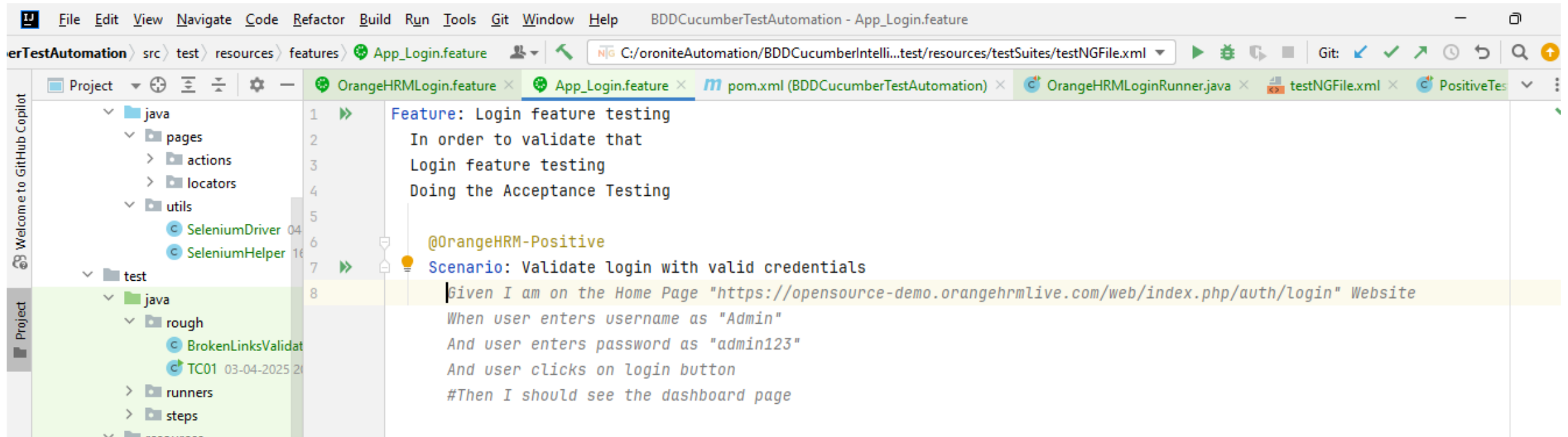
When the user clicks on the login button

Then the login should fail

And the invalid credentials error message should be displayed

Intelligent Recommendations for Feature File Development

- Provide suggested test steps for creating feature files from the given scenario, utilizing insights from the project codebase.
- Note – **Press tab to accept the suggestions, make sure to review the suggestions and customize if needed.**



The screenshot shows an IDE window titled "BDDCucumberTestAutomation - App_Login.feature". The left sidebar displays a project tree with folders like "java", "pages", "actions", "locators", "utils", and "test". The main editor area shows the content of "App_Login.feature" with line numbers 1 through 8. The feature file content is as follows:

```
1 Feature: Login feature testing
2   In order to validate that
3   Login feature testing
4   Doing the Acceptance Testing
5
6   @OrangeHRM-Positive
7   Scenario: Validate login with valid credentials
8     Given I am on the Home Page "https://opensource-demo.orangehrmlive.com/web/index.php/auth/login" Website
       When user enters username as "Admin"
       And user enters password as "admin123"
       And user clicks on login button
       #Then I should see the dashboard page
```

The IDE interface includes a menu bar (File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, Git, Window, Help) and a toolbar with various icons for navigation and execution. The status bar at the bottom indicates the current file is "App_Login.feature".

Developing Page Locators Class

Create classes for page locators and actions to interact with the user interface.



Prompt - Create LoginPageLocators Class for the above feature file using the webPage Locators

```
package pages.locators;

import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;

public class LoginPageLocators {

    @FindBy(xpath = "//input[@placeholder='Username']")
    public WebElement usernameTextbox;

    @FindBy(xpath = "//input[@placeholder='Password']")
    public WebElement passwordTextbox;

    @FindBy(xpath = "//button[@type='submit']")
    public WebElement loginButton;

    @FindBy(xpath = "//h6[@class='oxd-text oxd-text--h6 oxd-topbar-header-breadcrumb-module']")
    public WebElement dashboardHeaderText;

    @FindBy(xpath = "//p[@class='oxd-text oxd-text--p oxd-alert-content-text']")
    public WebElement invalidCredentialsErrorMessage;
}
```

Developing Page Actions Class

Create classes for page locators and actions to interact with the user interface.



Prompt - Create LoginPageActions class for the LoginPageLocators class

```
package pages.actions;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.support.PageFactory;
import pages.locators.LoginPageLocators;
public class LoginPageActions {

    WebDriver driver;
    LoginPageLocators loginPageLocators;

    public LoginPageActions(WebDriver driver) {
        this.driver = driver;
        this.loginPageLocators = new LoginPageLocators();
        PageFactory.initElements(driver, loginPageLocators);
    }

    public void enterUsername(String username) {
        loginPageLocators.usernameTextbox.sendKeys(username);
    }

    public void enterPassword(String password) {
        loginPageLocators.passwordTextbox.sendKeys(password);
    }

    public void clickLoginButton() {
        loginPageLocators.loginButton.click();
    }

    public boolean isDashboardHeaderVisible() {
        return loginPageLocators.dashboardHeaderText.isDisplayed();
    }

    public boolean isInvalidCredentialsMessageVisible() {
        return loginPageLocators.invalidCredentialsErrorMessage.isDisplayed();
    }
}
```

Creating Step Definitions

Develop step definitions to link the feature files with the test steps.



Prompt - Create LoginTestSteps for Login.feature file using LoginPageLocators and LoginPageActions class put proper validation checkpoints using TestNG Assertions

```
import utils.SeleniumDriver;

public class LoginTestSteps {

    WebDriver driver = SeleniumDriver.getDriver();
    LoginPageActions loginPageActions = new LoginPageActions(driver);

    @Given("the user navigates to the application URL [string]")
    public void navigateToApplicationURL(String url) {
        driver.get(url);
    }

    @Given("the user enters valid username [string] in the username textbox")
    @Given("the user enters invalid username [string] in the username textbox")
    public void enterUsername(String username) {
        loginPageActions.enterUsername(username);
    }

    @Given("the user enters valid password [string] in the password textbox")
    @Given("the user enters invalid password [string] in the password textbox")
    public void enterPassword(String password) {
        loginPageActions.enterPassword(password);
    }

    @When("the user clicks on the login button")
    public void clickLoginButton() {
        loginPageActions.clickLoginButton();
    }

    @Then("the user should be logged in successfully")
    public void validateSuccessfulLogin() {
        Assert.assertTrue(loginPageActions.isDashboardHeaderVisible(), "Dashboard header text is not visible");
    }

    @Then("the login should fail")
    public void validateFailedLogin() {
        Assert.assertTrue(loginPageActions.isInvalidCredentialsMessageVisible(), "Invalid credentials error message is not visible");
    }

    @Then("the dashboard header text should be visible")
    public void validateDashboardHeaderText() {
        Assert.assertTrue(loginPageActions.isDashboardHeaderVisible(), "Dashboard header text is not visible");
    }

    @Then("the invalid credentials error message should be displayed")
    public void validateInvalidCredentialsErrorMessage() {
        Assert.assertTrue(loginPageActions.isInvalidCredentialsMessageVisible(), "Invalid credentials error message is not visible");
    }
}
```

Setting Up the Runner Class

Create a runner class to execute the tests efficiently.



Prompt - Create LoginTestRunner class by referring PositiveTest Runner class

```
package runners;

import io.cucumber.testng.CucumberOptions;
import io.cucumber.testng.AbstractTestNGCucumberTests;

@CucumberOptions(
    plugin = {"json:target/login/cucumber.json",
             "pretty", "html:target/login/cucumber.html",
             "com.aventstack.extentreports.cucumber.adapter.ExtentCucumberAdapter:"},
    features = "src/test/resources/FeatureFiles/Login.feature",
    glue = "steps",
    tags = "@Login"
)

public class LoginTestRunner extends AbstractTestNGCucumberTests {

    //add the code snippet to run the tests in parallel
    @Override
    @org.testng.annotations.DataProvider(parallel = true)
    public Object[][] scenarios() {
        return super.scenarios();
    }
}
```

Creating TestNG XML Suite

Set up a TestNG XML suite for organized test execution.



Prompt - create testNGsuite xml file to run the LoginTestRunner class

Creating Login Test Automation

Below is the `testng-suite.xml` file to run the `LoginTestRunner` class:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="LoginTestSuite" parallel="classes" thread-count="4">
  <test name="LoginTest">
    <classes>
      <class name="runners.LoginTestRunner"/>
    </classes>
  </test>
</suite>
```




Overview of GitHub Copilot Chat Options

- **Simplify This:** Refactors complex code into simpler versions.
- **Generate Docs:** Creates documentation for your code automatically.
- **Generate Tests:** Suggests unit tests for your code.
- **Explain This:** Provides explanations of code snippets.
- **Fix This:** Identifies and suggests fixes for code errors.

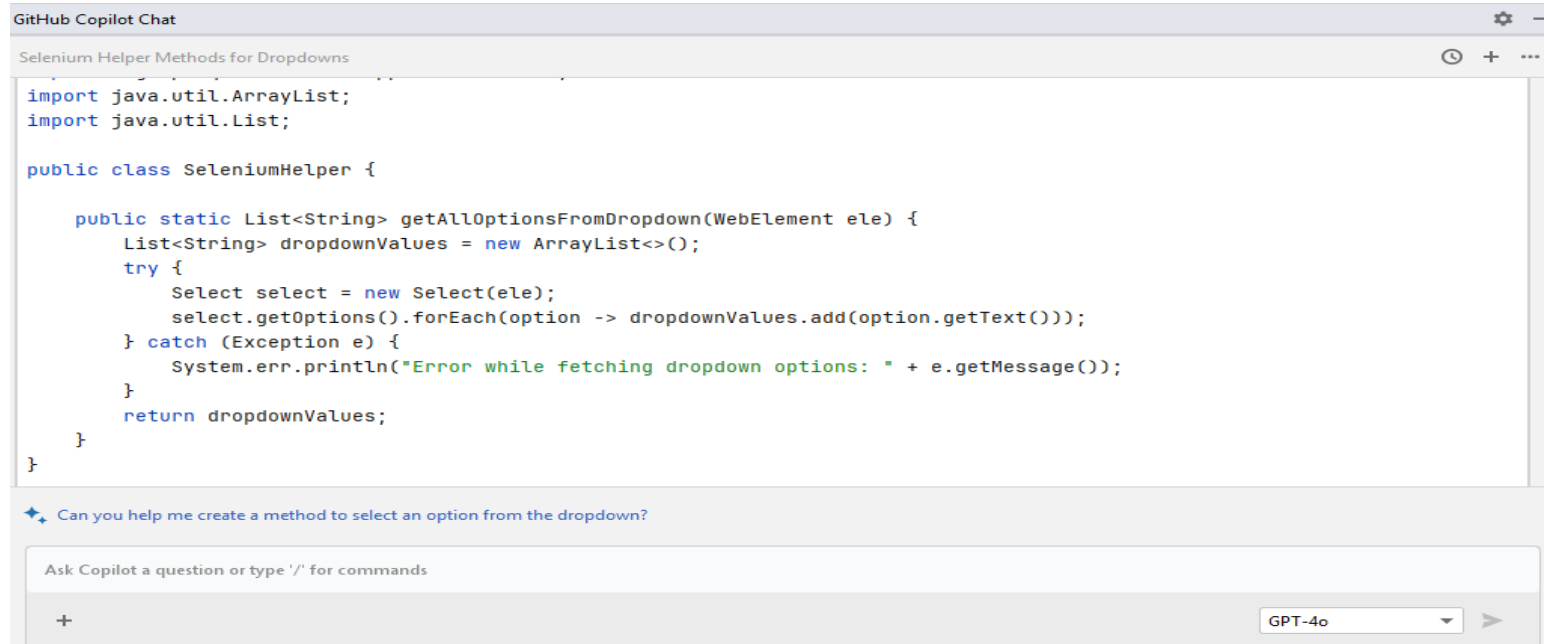
Creating code logic with appropriate comments.

- Prompt- Create one static method named pageScrollToWebElement which will scroll the web page to the given web element coming into this method as an WebElement parameter
- Note – **Press tab to accept the suggestions, make sure to review the suggestions and customize if needed.**

```
//create one static method named pageScrollToWebElement which will scroll the webpage to the given web element
// coming into this method as an WebElement parameter
public static void pageScrollToWebElement(WebElement webElement) {
    try {
        // Scroll the webpage to the given web element
        ((org.openqa.selenium.JavascriptExecutor) webElement).executeScript("arguments[0].scrollIntoView(true);", webElement);
    } catch (Exception e) {
        System.out.println("Error scrolling to the element: " + e.getMessage());
    }
}
```

Creating code using GitHub Copilot Chat

- Prompt - Create a method for SeleniumHelper class, method should be static named `getAllOptionsFromDropdown(WebElement ele)`, the logic should be able to fetch all the options from a dropdown webElement, WebElement should come as a parameter to this method. use Try catch block to handle exceptions related to WebElement. this method should return the list of all the dropdown values as String.
- Note – **code suggestions must be reviewed thoroughly before using and customize if needed. Else it can cause errors.**



The screenshot shows the GitHub Copilot Chat interface. The chat window title is "Selenium Helper Methods for Dropdowns". The code suggestion is as follows:

```
import java.util.ArrayList;
import java.util.List;

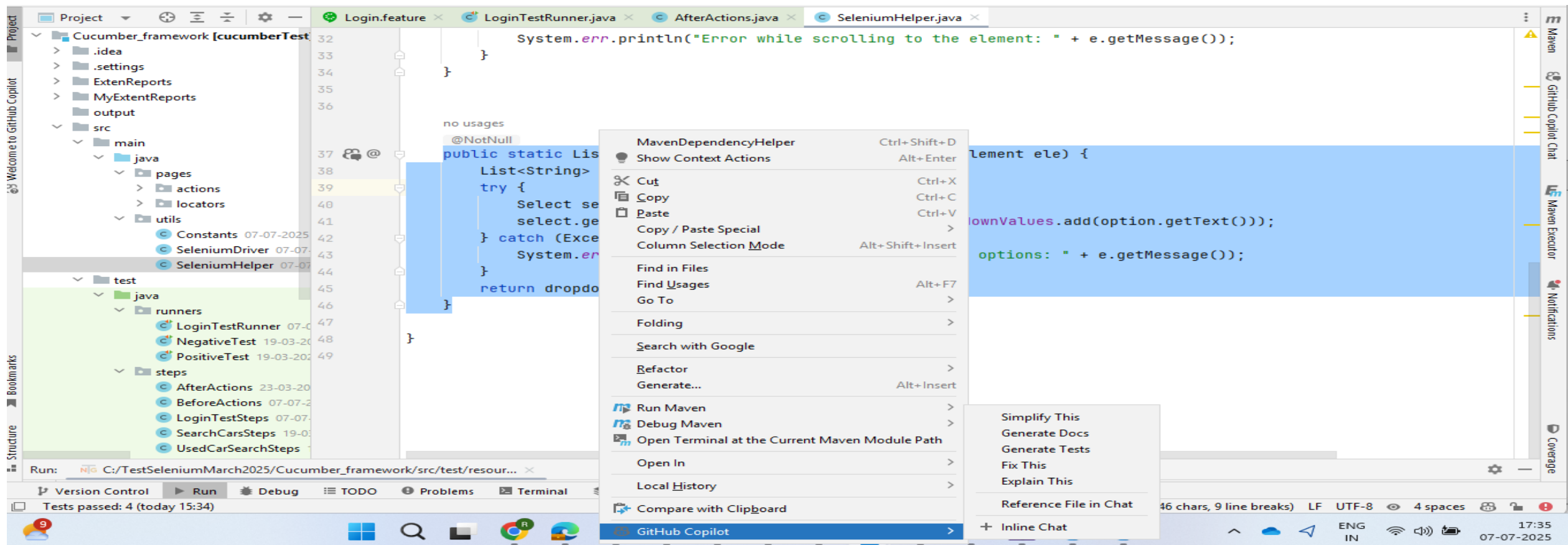
public class SeleniumHelper {

    public static List<String> getAllOptionsFromDropdown(WebElement ele) {
        List<String> dropdownValues = new ArrayList<>();
        try {
            Select select = new Select(ele);
            select.getOptions().forEach(option -> dropdownValues.add(option.getText()));
        } catch (Exception e) {
            System.err.println("Error while fetching dropdown options: " + e.getMessage());
        }
        return dropdownValues;
    }
}
```

Below the code, there is a prompt: "Can you help me create a method to select an option from the dropdown?". At the bottom, there is a text input field with the placeholder "Ask Copilot a question or type '/' for commands", a plus icon, and a dropdown menu showing "GPT-4o".

Options available with GitHub Copilot chat

- Select a particular code snippet, right click, go to GitHub Copilot. It will show the below options show in the screenshot below
- Note – **code suggestions must be reviewed thoroughly before using and customize if needed. Else it can cause errors.**



Fix errors using GitHub Copilot

- Select a particular code snippet, right click, go to GitHub Copilot, select **“Fix This”** option. It will show the suggestions to fix the code .
- Note – **code suggestions must be reviewed thoroughly before using and customize if needed. Else it can cause errors.**

Project

- Cucumber_framework [cucumberTest]
- .idea
- .settings
- ExtentReports
- MyExtentReports
- output
- src
 - main
 - java
 - pages
 - actions
 - locators
 - utils
 - Constants 07-07-2025
 - SeleniumDriver 07-07-2025
 - SeleniumHelper 07-07-2025
 - test
 - java
 - runners
 - LoginTestRunner 07-07-2025
 - NegativeTest 19-03-2025
 - PositiveTest 19-03-2025
 - steps
 - AfterActions 23-03-2025
 - BeforeActions 07-07-2025
 - LoginTestSteps 07-07-2025
 - SearchCarsSteps 19-03-2025
 - UsedCarSearchSteps 19-03-2025

GitHub Copilot Chat

Creating Test Runners and Utility Methods in Selenium

3 steps completed successfully

Problem 1: Incorrect variable usage in pageScrollToWebElement method
The variable `webElement` is used in the method, but it is not defined. The correct variable name is `webElement1`, which is passed as a parameter to the method.

Fix:
Replace `webElement` with `webElement1` in the method.

Before:

```
((org.openqa.selenium.JavascriptExecutor) webElement).executeScript("arguments[0].scrollIntoView(true);", webElement);
```

After:

```
((org.openqa.selenium.JavascriptExecutor) webElement1).executeScript("arguments[0].scrollIntoView(true);", webElement1);
```

Can you help me with writing a test case for the Login feature?

Ask Copilot a question or type '/' for commands

+ GPT-4o

Run: C:/TestSeleniumMarch2025/Cucumber_framework/src/test/resour...

Version Control Run Debug TODO Problems Terminal Package Search Services Build

Code explanation using GitHub Copilot

- Select a particular code snippet, right click, go to GitHub Copilot, select “**Explain This**” option. It will show the suggestions to fix the code .
- Note – **code suggestions must be reviewed thoroughly before using and customize if needed. Else it can cause errors.**

The screenshot displays the GitHub Copilot Chat window within an IDE. The title bar reads "GitHub Copilot Chat". Below it, the context "Selenium Helper Methods and TestNG Suite" is shown. The input field contains the command `/explain`. The response from GitHub Copilot is as follows:

GitHub Copilot

> 2 steps completed successfully

The selected code defines a static method named `getAllOptionsFromDropdown` in the `SeleniumHelper` class. This method is designed to retrieve all the options from a dropdown `WebElement` and return them as a list of strings. It uses Selenium's `Select` class to interact with the dropdown and includes exception handling to manage potential issues.

Parameter

The method accepts a single parameter:

```
WebElement ele
```

This represents the dropdown element from which the options will be extracted.

✦ Can you help me with writing a test case for the `getAllOptionsFromDropdown` method?

At the bottom, there is a text input field with the placeholder "Ask Copilot a question or type '/' for commands". To the right of the input field is a dropdown menu set to "GPT-4o" and a send button (arrow icon).

Code optimization using GitHub Copilot

- Select a particular code snippet, right click, go to GitHub Copilot, select “Simplify This” option. It will show the suggestions to fix the code .
- Note – **code suggestions must be reviewed thoroughly before using and customize if needed. Else it can cause errors.**



The screenshot displays the GitHub Copilot Chat interface within an IDE. The chat window is titled "Enhancing SeleniumHelper Class Methods". It shows a Java code snippet for a method that fetches dropdown options. Below the code, a "Simplifications:" section lists two changes made by Copilot: replacing a lambda expression with a traditional for loop and combining object creation with a method call. The sidebar on the right shows icons for Maven, GitHub Copilot Chat, and Maven Executor.

GitHub Copilot Chat

Enhancing SeleniumHelper Class Methods

```
List<String> dropdownValues = new ArrayList<>();
try {
    for (WebElement option : new Select(ele).getOptions()) {
        dropdownValues.add(option.getText());
    }
} catch (Exception e) {
    System.err.println("Error fetching dropdown options: " + e.getMessage());
}
return dropdownValues;
}
```

Simplifications:

1. **Replaced `forEach` with a traditional `for` loop:** This makes the code more straightforward for developers who may not be familiar with lambda expressions.
2. **Combined object creation and method call:** The `Select` object is created inline with the `getOptions()` call, reducing unnecessary variable declarations.

✦ Can you help me with another method for the SeleniumHelper class?

Doc generation using GitHub Copilot

- Select a particular code snippet, right click, go to GitHub Copilot, select “Generate Docs” option. It will show the suggestions to fix the code .
- Note – **code suggestions must be reviewed thoroughly before using and customize if needed. Else it can cause errors.**

GitHub Copilot Chat

Selenium Dropdown Handling and Code Simplification

```
/**
 * Retrieves all options from a dropdown WebElement and returns them as a list of strings.
 *
 * @param ele The WebElement representing the dropdown.
 * @return A list of strings containing the text values of all dropdown options.
 *         If an exception occurs, an empty list is returned.
 * @throws Exception Handles any exception that may occur while interacting with the WebElement.
 */
public static List<String> getAllOptionsFromDropdown(WebElement ele) {
    List<String> dropdownValues = new ArrayList<>();
    try {
        // Create a Select object to interact with the dropdown
        Select select = new Select(ele);

        // Iterate through all options and add their text to the list
        select.getOptions().forEach(option -> dropdownValues.add(option.getText()));
    } catch (Exception e) {
```


Limitations And Best Practices

Limitations

Limitations	Description
Limited scope of knowledge	The tool learns from code available in GitHub repositories. However, it may struggle with certain problems, complicated code structures, specific programming languages, or data formats that are not well-represented in the examples it was trained on.
Break down the bigger problem into smaller functions	Break down complex code into smaller, manageable functions for better clarity and understanding.
Inaccurate code	The tool provider informs that the generated code may not meet the user's expectations, may not solve the actual problem, and may contain errors
Inaccurate chat responses	When using the chat feature, how accurate the answers are mostly depending on how well you ask the question or give a command. The documentation states that "Copilot Chat is not meant to answer questions that aren't about coding." This means that if you ask about topics outside of coding, like design, the answers might not make sense or be relevant.

GitHub Copilot



Ask clear, specific coding questions



Stick to programming topics



Break down complex code into smaller parts

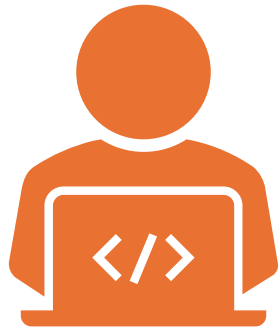


Review and test generated code



Understand that Copilot may not always

Tool limitations guidelines



Since GitHub Copilot is based on generative AI, you must always remember that it may generate incorrect code or responses. Therefore, when using the tool, you must be aware of potential limitations.



Note: To mitigate potential limitations, always conduct code reviews, verify alternative solutions, and provide clear and precise prompts.

Prompts generation guidelines

- [Prompt engineering for Copilot Chat - GitHub Docs](#)
- Here are the **GitHub Copilot prompt generation standards and guidelines**, based on official documentation and best practices

1. Start General, Then Get Specific

- Begin with a broad description of what you want to achieve, then narrow it down with specific requirements.

Example:

- "Write a function that checks if a number is prime.
It should return true for prime numbers and false otherwise."

2. Give Examples

- Providing input/output examples helps Copilot understand your intent more clearly.

Example:

- "Write a function that extracts all dates from a string.
Input: 'Meeting on 12/05/2025 and 13-06-2025'
Output: ['12/05/2025', '13-06-2025']"

3. Break Complex Tasks into Smaller Steps

- Instead of asking Copilot to do everything at once, break the task into manageable parts.

Example:

- Step 1: Generate a login page class with locators.
- Step 2: Write methods for entering credentials.
- Step 3: Create a test case using those methods.

4. Avoid Ambiguity

- Be specific in your language. Avoid vague terms like “this” or “that.”

Example:

Instead of: "What does this function do?"

Say: "What does the validateUserInput() method do?"

5. Indicate Relevant Code or Context

If you're working in an IDE, make sure the relevant files are open. Copilot uses the surrounding code to generate better suggestions.

6. Experiment and Iterate

Try rephrasing your prompt if the output isn't what you expected. Small changes can lead to better results.

7. Follow Good Coding Practices

Use clear naming conventions, consistent formatting, and comments. This helps Copilot understand your code style and generate cleaner suggestions.

Ref- <https://docs.github.com/en/copilot/concepts/prompt-engineering-for-copilot-chat>

Summary of GitHub Copilot in Automation

- Introduction to GitHub Copilot and its AI capabilities.
- Significant benefits include time savings in coding tasks.
- Integration with various IDEs enhances developer efficiency.
- Automating test development streamlines the process significantly.
- Awareness of limitations is crucial for effective usage.





"Thank you! I'm
happy to answer
any questions you
may have."