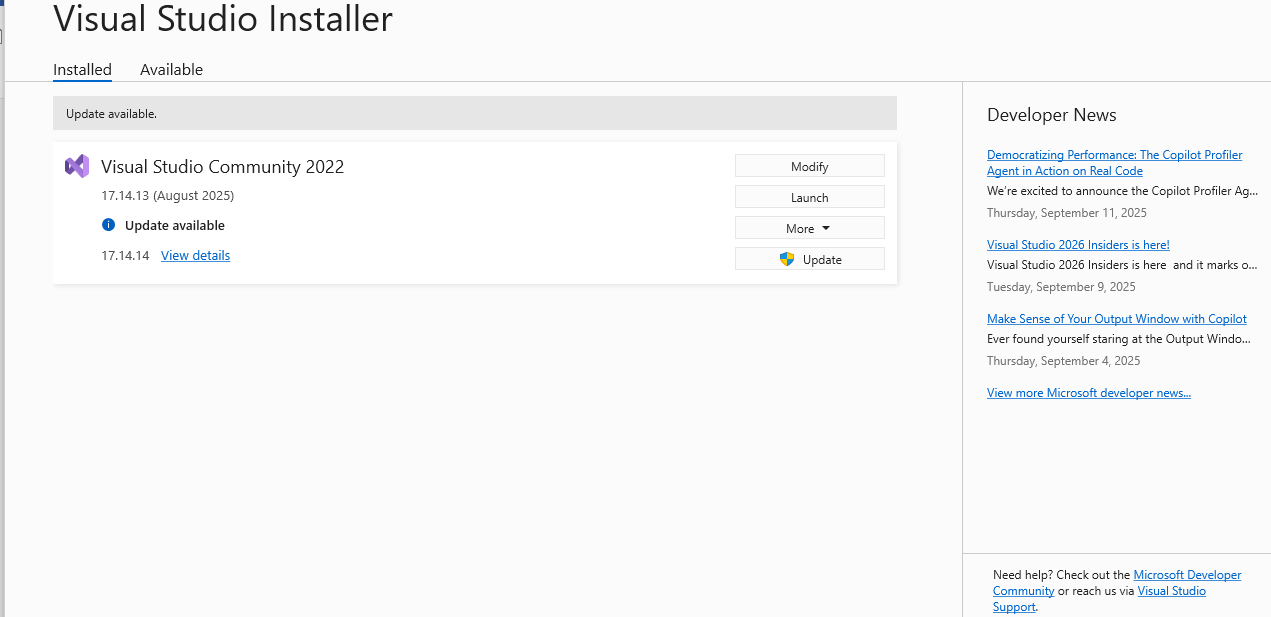
**Research & Write Test Cases**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test ID | User story | Scenario | Preconditions | steps | Expected result |
| |  | | --- | | **TC-01** |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | **US-01 – Event Submission** |  |  | | --- | |  | | | |  | | --- | | Organizer creates an event with valid info . |  |  | | --- | |  | | |  | | --- | | Organizer account verified & logged in |  |  | | --- | |  | | |  | | --- | | 1) Click **Create Event**  2) Fill all required fields  3) Click **Submit** |  |  | | --- | |  | | |  | | --- | | Event status becomes **Submitted**; appears in **Admin Review**; organizer sees success message |  |  | | --- | |  | |
| |  | | --- | | **TC-02** |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | **US-02 – Manage Events (Edit)** |  |  | | --- | |  | | | |  | | --- | | Organizer edits a **Draft/Submitted** event and saves |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Organizer has an event in **Draft** (or **Submitted**, not Approved) |  |  | | --- | |  | |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | 1) Open **My Events**  2) Open event  3) Click **Edit**  4) Change Title  5) **Save** |  |  | | --- | |  | | | |  | | --- | | Changes saved; event keeps its state (Draft/Submitted); updated title shown in list/details |  |  | | --- | |  | |
| |  | | --- | | **TC-03** |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | **US-02 – Manage Events (Delete)** |  |  | | --- | |  | |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Organizer deletes a **Draft** event with confirmation |  |  | | --- | |  | |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Organizer has a **Draft** event |  |  | | --- | |  | |  |  | | --- | |  | | |  | | --- | | 1) Open **My Events** 2) Select Draft event  3) Click **Delete**  4) Confirm |  |  | | --- | |  | | |  | | --- | | Event removed from list; confirmation message displayed; no longer searchable |  |  | | --- | |  | |
| |  | | --- | | **TC-04** |  |  | | --- | |  | | |  | | --- | | **US-03 – Organizer Profile** |  |  | | --- | |  | | |  | | --- | | Organizer updates profile and requests verification |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Organizer logged in; profile not verified |  |  | | --- | |  | |  |  | | --- | |  | | |  | | --- | | 1) Open **Profile**  2) Upload logo  3) Enter website/contact  4) Click **Request Verification** |  |  | | --- | |  | | |  | | --- | | Profile saved; status = **Pending Verification**; organizer sees “Submitted for review” message |  |  | | --- | |  | |
| |  | | --- | | **TC-05** |  |  | | --- | |  | | |  | | --- | | **US-04 – Browse Events** |  |  | | --- | |  | | |  | | --- | | Attendee filters by **Category=Family** and **Date=Weekend** |  |  | | --- | |  | | |  | | --- | | ≥3 events exist with mixed categories/dates |  |  | | --- | |  | | |  | | --- | | 1) Open **Events**  2) Set filters (Family + Weekend)  3) Click **Apply** |  |  | | --- | |  | | |  | | --- | | Only matching events listed; count updates; no unrelated events shown |  |  | | --- | |  | |
| |  | | --- | | **TC-06** |  |  | | --- | |  | | |  | | --- | | **US-05 – Event Details** |  |  | | --- | |  | | |  | | --- | | Attendee views event details page (map/time/contact visible) |  |  | | --- | |  | | |  | | --- | | Event has location, time, contact info |  |  | | --- | |  | | |  | | --- | | 1) From list, open the event card  2) View details |  |  | | --- | |  | | |  | | --- | | Page shows title, description, images, **map**, **local time**, and **contact info** correctly |  |  | | --- | |  | |
| |  | | --- | | **TC-07** |  |  | | --- | |  | | |  | | --- | | **US-06 – Save & Share** |  |  | | --- | |  | | |  | | --- | | Attendee adds event to **Google Calendar** successfully |  |  | | --- | |  | | |  | | --- | | Event is public; user in browser session |  |  | | --- | |  | | |  | | --- | | 1) Open event details  2) Click **Add to Calendar**  3) Choose **Google** 4) Confirm |  |  | | --- | |  | | |  | | --- | | Calendar entry created with correct title, start/end time, and venue |  |  | | --- | |  | |
| |  | | --- | | **TC-08** |  |  | | --- | |  | | |  | | --- | | **US-07 – Review Submissions** |  |  | | --- | |  | | |  | | --- | | Admin **approves** a submitted event with a note |  |  | | --- | |  | | |  | | --- | | At least one event in **Submitted** state |  |  | | --- | |  | | |  | | --- | | 1) Login as **Admin** 2) Open **Review Queue**  3) Open event  4) Click **Approve**  5) Add note  6) Confirm |  |  | | --- | |  | | |  | | --- | | Status = **Approved**; event becomes public; organizer notified with the note |  |  | | --- | |  | |
| |  | | --- | | **TC-09** |  |  | | --- | |  | | |  | | --- | | **US-08 – Content Moderation** |  |  | | --- | |  | | |  | | --- | | Admin removes an **inappropriate** public event |  |  | | --- | |  | | |  | | --- | | Event is public and flagged as inappropriate |  |  | | --- | |  | | |  | | --- | | 1) Admin opens event  2) Click **Remove/Unpublish** 3) Enter reason  4) Confirm |  |  | | --- | |  | | |  | | --- | | Event hidden/unpublished; reason logged; no longer visible to attendees |  |  | | --- | |  | |
| |  | | --- | | **TC-10** |  |  | | --- | |  | | |  | | --- | | **US-09 – Organizer Verification** |  |  | | --- | |  | | |  | | --- | | Admin verifies organizer after document check (ABN/community proof) |  |  | | --- | |  | | |  | | --- | | Organizer submitted verification; docs provided |  |  | | --- | |  | | |  | | --- | | 1) Admin opens **Organizer Verification** queue 2) Review docs  3) Click **Verify** |  |  | | --- | |  | | Organizer marked **Verified**; organizer notified; verified badge appears on profile/events |
|  |  |  |  |  |  |

**Installed Selenium on Visual Studio**

****

Link Video tut:

[**https://www.youtube.com/watch?v=jyxrqFg4QnI**](https://www.youtube.com/watch?v=jyxrqFg4QnI)

Sample test:

using OpenQA.Selenium;

using OpenQA.Selenium.Chrome;

namespace DotnetSelenium

{

public class Tests

{

[SetUp]

public void Setup()

{

}

[Test]

public void Test1()

{

//Sudo code for setting up selenium

//1. create a new instance of Selenium Web Driver

IWebDriver driver = new ChromeDriver();

// 2. Navigate to the URL

driver.Navigate().GoToUrl("https://www.google.com/");

// 2a. Maximize the browser window

driver.Manage().Window.Maximize();

//3. Find the element

IWebElement webElement = driver.FindElement(By.Name("q"));

// 4.Type in the element

webElement.SendKeys("Selenium");

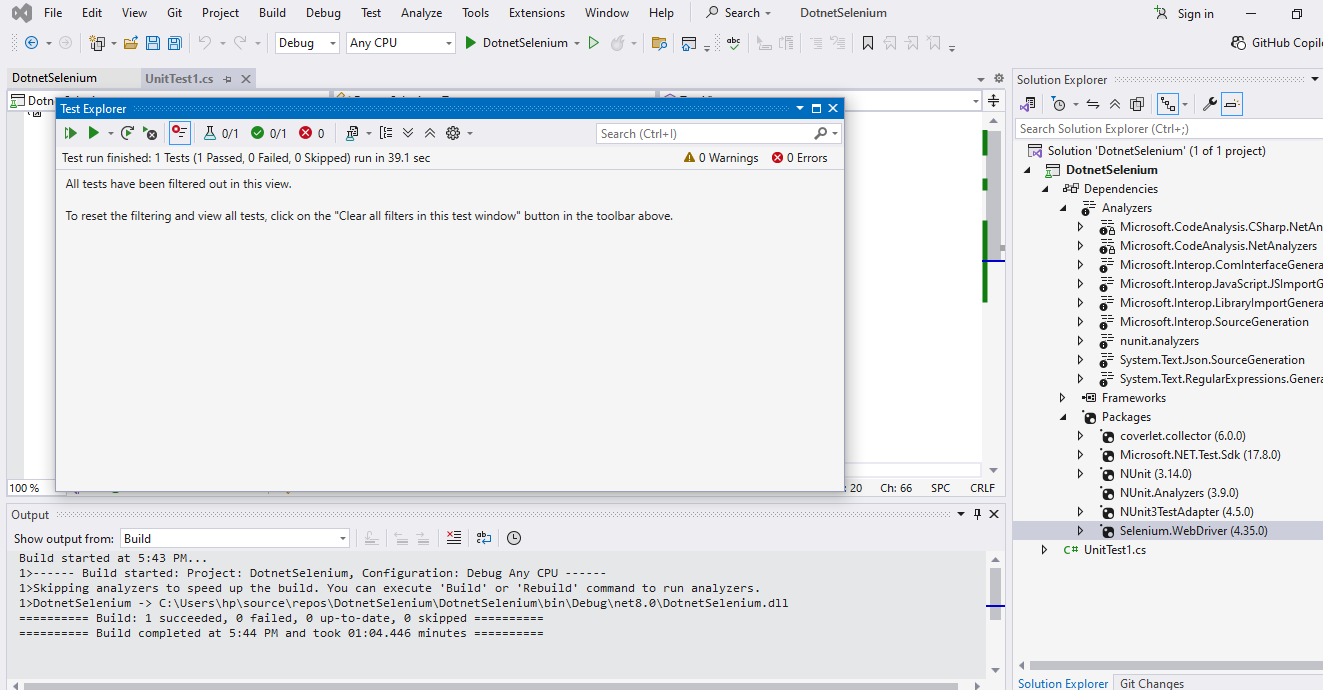
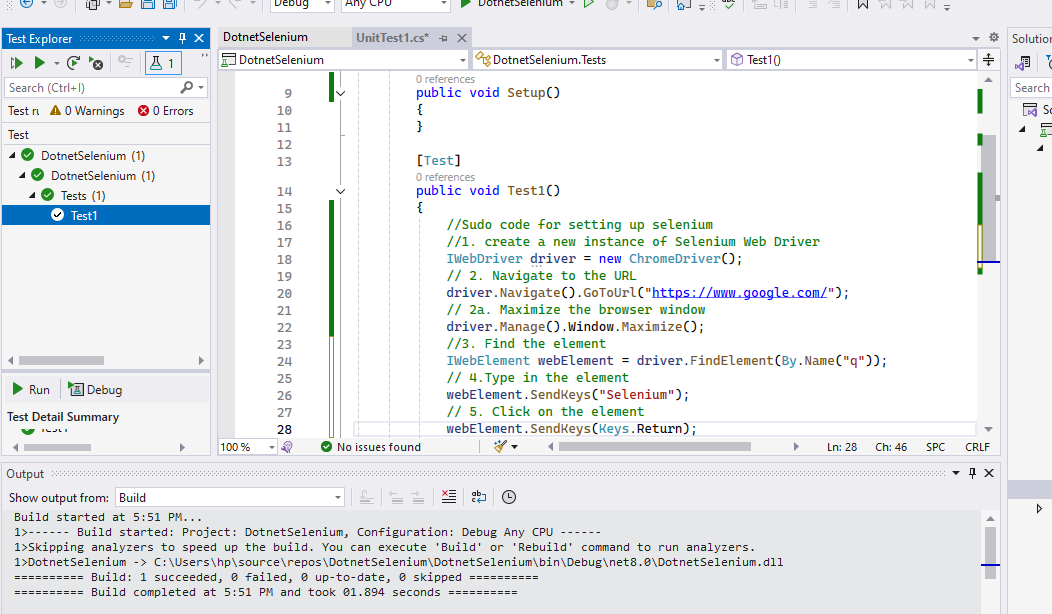
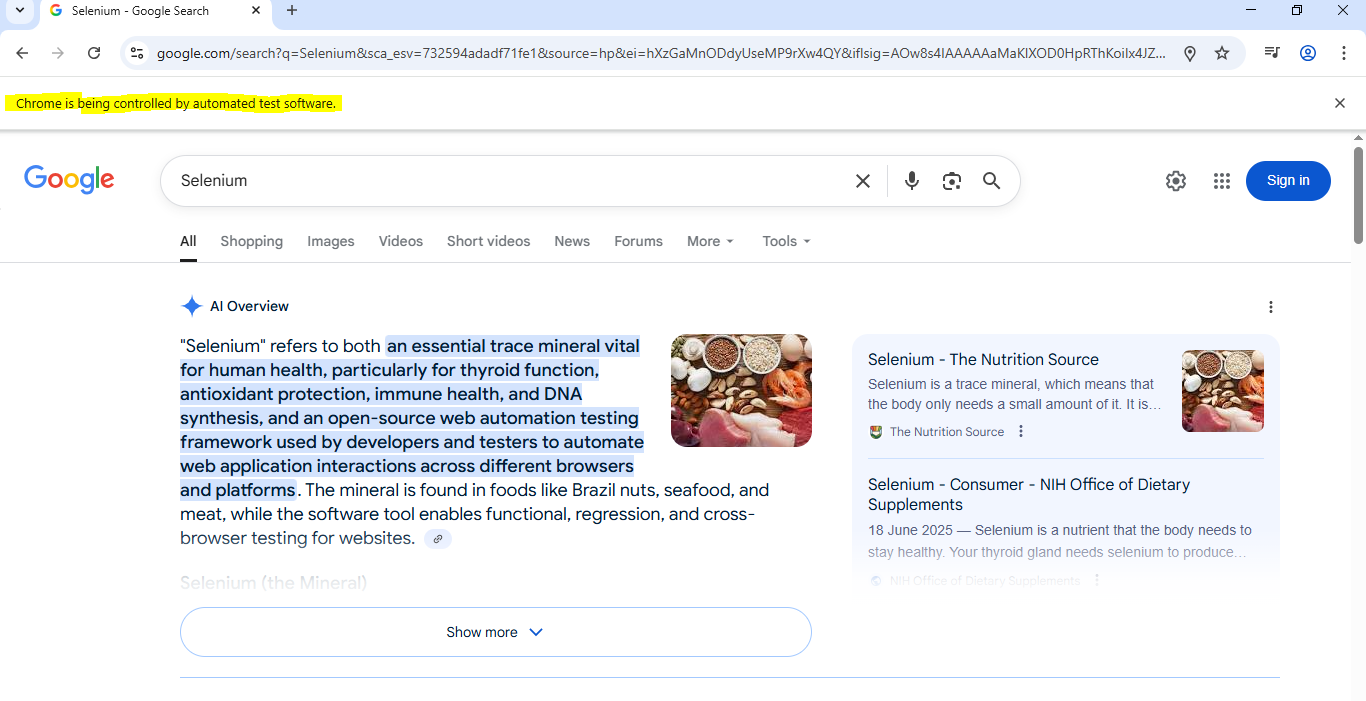
// 5. Click on the element

webElement.SendKeys(Keys.Return);

}

}

}



**Identify Testing Tools**

**Postman**

* What it is: A tool to check if APIs (the “back-end connections”) are working.
* What it does: You can send requests like “give me all events” and see the reply from the system.
* Simple example: Type the event API link in Postman → click Send → it shows event details in JSON format.

**Selenium**

* What it is: A tool to control a web browser automatically.
* What it does: You write small scripts that click buttons, type in forms, and check results.
* Simple example: Selenium opens Chrome, logs in as an organizer, fills out the event form, clicks submit, and checks that the event was created.

**JMeter**

* What it is: A tool to test performance (speed under heavy load).
* What it does: Pretends to be many users using the system at the same time.
* Simple example: You set 200 fake users in JMeter → all apply filters at once → JMeter measures how fast the system responds.

**Fiddler**

* What it is: A tool to watch and record network traffic (the messages going between your computer and the server).
* What it does: Helps you see what’s sent and what’s received.
* Simple example: You click “Register for Event” → Fiddler shows the exact request your browser sent and the server’s reply.

**Cypress**

* What it is: A modern tool for testing websites from start to finish.
* What it does: Runs in the browser and checks that pages, buttons, and flows work correctly.
* Simple example: Cypress opens your site, applies “Family Weekend” filter, and checks that only Family events show in the list.

**Zephyr**

is a **test management tool** that works inside Jira. It helps teams create, organize, and track test cases directly with user stories and bugs. In simple words, it’s used to make testing easier by connecting requirements, test cases, and results all in one place.

**Manual vs Automated Testing**

Manual Testing

* The tester checks the system by hand: clicking buttons, filling forms, and observing results.
* *Example in your app:* Open the “Create Event” page, type in event details, and check if the event is saved.
* *Good for:* new features, exploratory testing, and quick checks.
* *Limitations:* slow, can miss things if repeated many times.

Automated Testing

* Tests are written as scripts or run in tools to repeat actions automatically.
* *Example in your app:* Use Selenium to automatically log in as an organizer, fill the event form, and submit it.
* *Good for:* regression testing, repeated checks, continuous integration.
* *Limitations:* setup takes more time, needs technical skills.

**2) Functional vs Non-Functional Testing**

Functional Testing

* Checks what the system does (features).
* *Example:* Verify an attendee can filter events by “Family + Weekend” and see the right list.
* *Goal:* Make sure each user story requirement works correctly.

Non-Functional Testing

* Checks how the system works (quality, performance, security).
* *Performance Example:* Test if the system can show event results quickly when 100 users search at the same time.
* *Security Example:* Test if only admins can approve or delete events.
* *Usability Example:* Check if the interface is simple and easy to use for attendees.

**3) Core Testing Types**

Smoke Testing

* Quick test to see if the system’s “main parts” are working.
* *Example:* Can organizers create events? Can attendees browse events? If these fail, no need to continue.

Sanity Testing

* Focused check after a small change or bug fix.
* *Example:* Developer fixes an error in “Save Event to Calendar”. You test just that part to confirm it’s fixed.

Regression Testing

* After new features are added, check that old features still work.
* *Example:* After adding “Share Event” feature, test again that “Browse Events” and “Event Details” still work.

User Acceptance Testing (UAT)

* Real users test the system in real scenarios before release.
* *Example:* Ask students or community members to use the app, browse events, and give feedback if it matches their needs.

Unit Testing

* Test one small piece of code at a time.
* Example: check if the date field only accepts future dates.
* *Goal:* Make sure each small function works correctly by itself.

Integration Testing

* Test how two or more parts work together.
* Example: when an organizer submits the event form the data goes into the database and then shows in the “My Events” list.
* *Goal:* Make sure connected parts of the system talk to each other properly.