Problem 1: Logic for Longest Substring Without Repeating Characters

Python Code:

```
python CopyEdit def
longest_unique_substring(s):
    seen = set()
left     =     0
max_len = 0

    for right in range(len(s)):              while
s[right] in seen:           seen.remove(s[left])
left += 1      seen.add(s[right])      max_len
= max(max_len, right - left + 1)

    return max_len

# Example usage
print(longest_unique_substring("abcabcbb"))  # Output: 3
print(longest_unique_substring("bbbbb"))     # Output: 1
```

Pseudo Code (for understanding):

```
sql
CopyEdit                                    function
longest_unique_substring(input_string):
    initialize set for seen characters
set left and max_length to 0

    loop over characters using right pointer:       if character
is already in set:            remove characters from left until
duplicate is removed           add current character to set
update max_length with current window size

    return max_length
```

Problem 2: Testing the Website

Assume a two-page flow:

1. **Input Page** (URL: https://agrichain.com/qa/input) ○ Input field (ID: stringInput) ○ Submit button (ID: submitBtn)
2. **Result Page** (URL: https://agrichain.com/qa/result) ○ Displays result in a tag with ID: result

Recreate manual and automation test cases with 2-3 line descriptions

```
import pandas as pd
manual_test_cases = [
  {
    "TC ID": "TC01",
    "Title": "Valid input with mixed characters",
    "Input": "abcabcbb",
    "Expected Output": "3",
    "Priority": "High",
    "Severity": "Major",
    "Description": "Tests a normal case with repeated characters after a few unique ones. Expected longest unique substring is 'abc'."
  },
  {
    "TC ID": "TC02",
    "Title": "All identical characters",
    "Input": "bbbbbb",
    "Expected Output": "1",
    "Priority": "Medium",
    "Severity": "Minor",
    "Description": "Validates system behavior when all characters are the same. Only one unique character is expected."
  },
  {
    "TC ID": "TC03",
    "Title": "Empty input string",
    "Input": "",
    "Expected Output": "0 or Error Message",
    "Priority": "Medium",
    "Severity": "Major",
    "Description": "Tests how system handles empty input. It should not break and should show a proper message or return 0."
  },
  {
    "TC ID": "TC04",
    "Title": "Single character input",
    "Input": "x",
    "Expected Output": "1",

    "Priority": "Low",

    "Severity": "Minor",

    "Description": "A minimal valid case with one character. The longest substring length should be 1."

  },

  {

    "TC ID": "TC05",

    "Title": "Alphanumeric characters",

    "Input": "a1b2c3",

    "Expected Output": "6",

    "Priority": "Low",

    "Severity": "Minor",

    "Description": "Tests mix of digits and letters. All are unique, so the result should equal string length."

  },

  {

    "TC ID": "TC06",

    "Title": "Input with space characters",

    "Input": "a b c a",

    "Expected Output": "4",

    "Priority": "Medium",

    "Severity": "Major",

    "Description": "Ensures space is treated as a valid character and included in uniqueness calculation."

  },

  {

    "TC ID": "TC07",

    "Title": "Special characters input",

    "Input": "!@#!@",

    "Expected Output": "3",

    "Priority": "Medium",

    "Severity": "Major",

    "Description": "Tests special characters for uniqueness handling. Symbols are treated just like letters."
    },
    {
        "TC ID": "TC08",
        "Title": "Access result page directly",
        "Input": "N/A",
        "Expected Output": "Error or redirect",
        "Priority": "Low",
        "Severity": "Major",
        "Description": "Checks if skipping input page and going to result page directly is gracefully handled."
    },
    {
        "TC ID": "TC09",
        "Title": "Performance with long input",
        "Input": "a" * 1000,
        "Expected Output": "1",
        "Priority": "Medium",
        "Severity": "Major",
        "Description": "Tests performance with large input to ensure system doesn’t crash or freeze."
    },
    {
        "TC ID": "TC10",
        "Title": "UI layout verification",
        "Input": "N/A",
        "Expected Output": "UI elements visible",
        "Priority": "Low",
        "Severity": "Minor",
        "Description": "Confirms visibility of input fields, submit button, and labels on both pages."
    }
]

automation_test_cases = [
{
    "TC ID": "ATC01",
    "Title": "Automate valid input flow",
    "Input": "abcabcbb",
    "Expected Output": "3",
    "Priority": "High",
    "Severity": "Major",
    "Description": "Automates full flow: enter input, submit, and validate result matches logic."
},
{
    "TC ID": "ATC02",
    "Title": "Automate repeated character input",
    "Input": "aaaaa",
    "Expected Output": "1",
    "Priority": "Medium",
    "Severity": "Minor",
    "Description": "Automates entry of repeated characters to ensure expected unique length is returned."
},
{
    "TC ID": "ATC03",
    "Title": "Special characters automation",
    "Input": "!@#@!",
    "Expected Output": "3",
    "Priority": "Medium",
    "Severity": "Minor",
    "Description": "Tests result correctness using symbols and verifies correct calculation using automation."
},
{
    "TC ID": "ATC04",
    "Title": "Automate blank input error",
    "Input": "",
        "Expected Output": "0 or error",

        "Priority": "Medium",

        "Severity": "Major",

        "Description": "Ensures system handles blank inputs with valid error message via automated test."

    },

    {

        "TC ID": "ATC05",

        "Title": "Automate navigation test",

        "Input": "abc",

        "Expected Output": "Return to input page",

        "Priority": "Low",

        "Severity": "Minor",

        "Description": "Verifies user can navigate back from result page and UI elements reload properly."

    }

]
```

Web Automation Script (Selenium + Python)

Test Case: ATC01 – Validate correct output for input abcabcbb

```
from selenium import webdriver from
selenium.webdriver.common.by import By

# Setup browser driver =
webdriver.Chrome()
driver.get("https://agrichain.com/qa/input
")

# Step 1: Enter input string driver.find_element(By.ID,
"stringInput").send_keys("abcabcbb") # Step 2: Click submit
driver.find_element(By.ID, "submitBtn").click()

# Step 3: Validate output result =
driver.find_element(By.ID, "result").text assert
result == "3", f"Expected 3 but got {result}"

# Cleanup driver.quit()
```

High-Level Automation Framework Structure agrichain_automation/

```
│
├── tests/
│   └── test_longest_substring.py        ← Test scripts
│
├── pages/
│   ├── input_page.py               ← InputPage class (locators + actions)
│   └── result_page.py              ← ResultPage class (locators + validation)
│
├── utils/
│   ├── browser_setup.py            ← WebDriver setup
│   └── config.py                   ← Base URL and constants
│
├── data/
│   └── testdata.json               ← External data (optional)
│
├── reports/
│   └── test_report.html            ← Execution reports
│
└── requirements.txt                ← All dependencies (Selenium, PyTest, etc.)
```