# SUMMER TRAINING COURSE-MAJOR PROJECT REPORT

*LOVELY PROFESSIONAL UNIVERSITY*

COURSE BY-BOARD INFINITY

*NAME – SHASHANK SHEKHAR*

*COURSE- BTECH CSE*

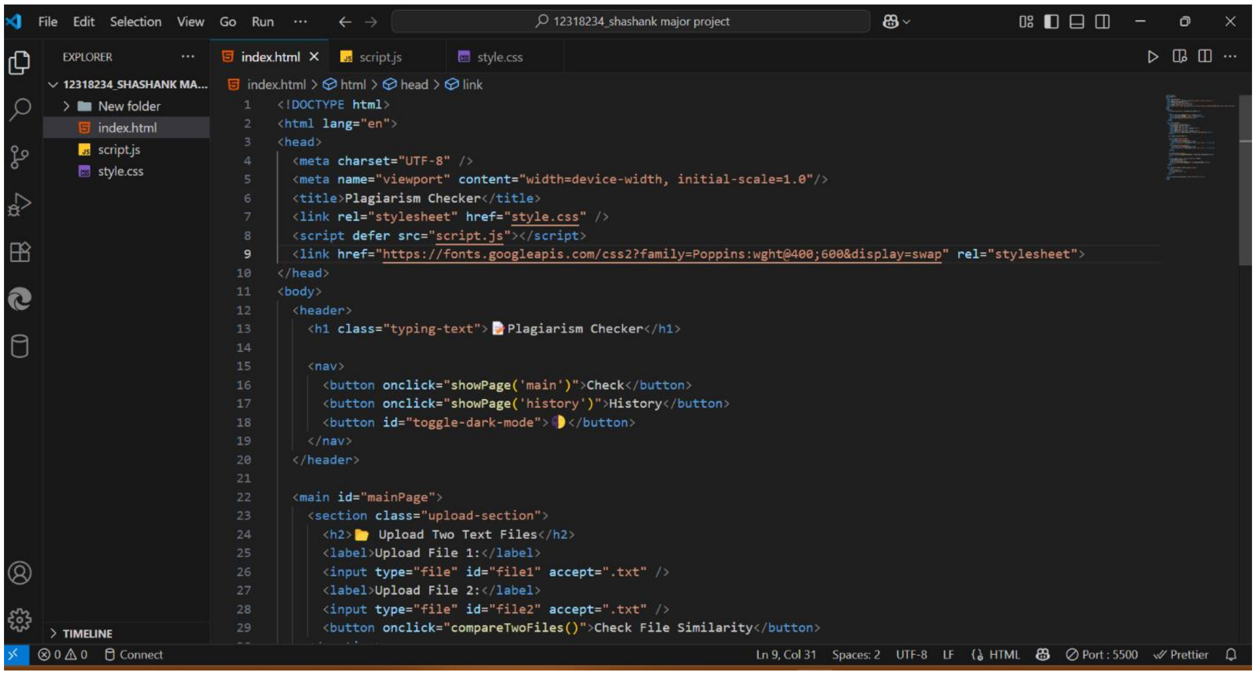
*REGISTRATION NUMBER-12318234*

*PROJECT TYPE-MAJOR PROJECT*

*TOPIC-PLAGIARISM CHECKER*

*SUBJECT-DSA FOR INTERVIEW*

HTML CODE AND EXPLANATION :



```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Plagiarism Checker</title>
  <link rel="stylesheet" href="style.css" />
  <script defer src="script.js"></script>
  <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap" rel="stylesheet">
</head>
<body>
  <header>
    <h1 class="typing-text">📝Plagiarism Checker</h1>

    <nav>
      <button onclick="showPage('main')">Check</button>
      <button onclick="showPage('history')">History</button>
      <button id="toggle-dark-mode">🌓</button>
    </nav>
  </header>

  <main id="mainPage">
    <section class="upload-section">
      <h2>📁 Upload Two Text Files</h2>
      <label>Upload File 1:</label>
      <input type="file" id="file1" accept=".txt" />
      <label>Upload File 2:</label>
      <input type="file" id="file2" accept=".txt" />
      <button onclick="compareTwoFiles()">Check File Similarity</button>
```

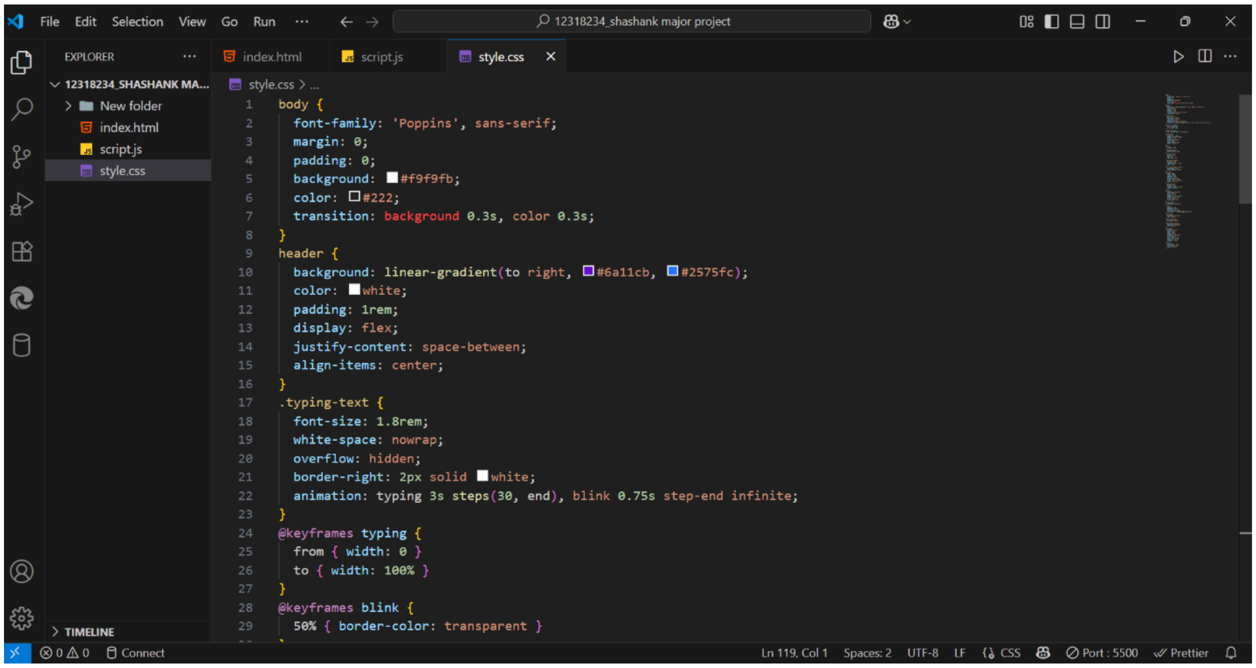
EXPLANATION:

This is the **main HTML file** that structures the content we can see on the webpage.

**Key Sections:**

- <header>
  - Title: "📝 Plagiarism Checker"
  - Navigation buttons:
    - **Check** (shows main page)
    - **History** (shows comparison history)
    - **Dark mode toggle** (🌓 icon)
- <main id="mainPage">
  - Section for uploading two .txt files and comparing them
  - Section to **paste text directly** and check similarity
  - A **button to compare**
  - A result section that shows similarity percentage and a download report button
- <main id="historyPage">
  - Displays history of past comparisons saved in the current session
- **Scroll to Top** button (⬆)

CSS CODE AND EXPLANATION:


File Edit Selection View Go Run ···
12318234_shashank major project
EXPLORER
12318234_SHASHANK MA...
New folder
index.html
script.js
style.css
style.css > ...
TIMELINE
Ln 119, Col 1 Spaces: 2 UTF-8 LF CSS Port : 5500 Prettier
Connect

```css
body {
  font-family: 'Poppins', sans-serif;
  margin: 0;
  padding: 0;
  background: #f9f9fb;
  color: #222;
  transition: background 0.3s, color 0.3s;
}
header {
  background: linear-gradient(to right, #6a11cb, #2575fc);
  color: white;
  padding: 1rem;
  display: flex;
  justify-content: space-between;
  align-items: center;
}
.typing-text {
  font-size: 1.8rem;
  white-space: nowrap;
  overflow: hidden;
  border-right: 2px solid white;
  animation: typing 3s steps(30, end), blink 0.75s step-end infinite;
}
@keyframes typing {
  from { width: 0 }
  to { width: 100% }
}
@keyframes blink {
  50% { border-color: transparent }
```

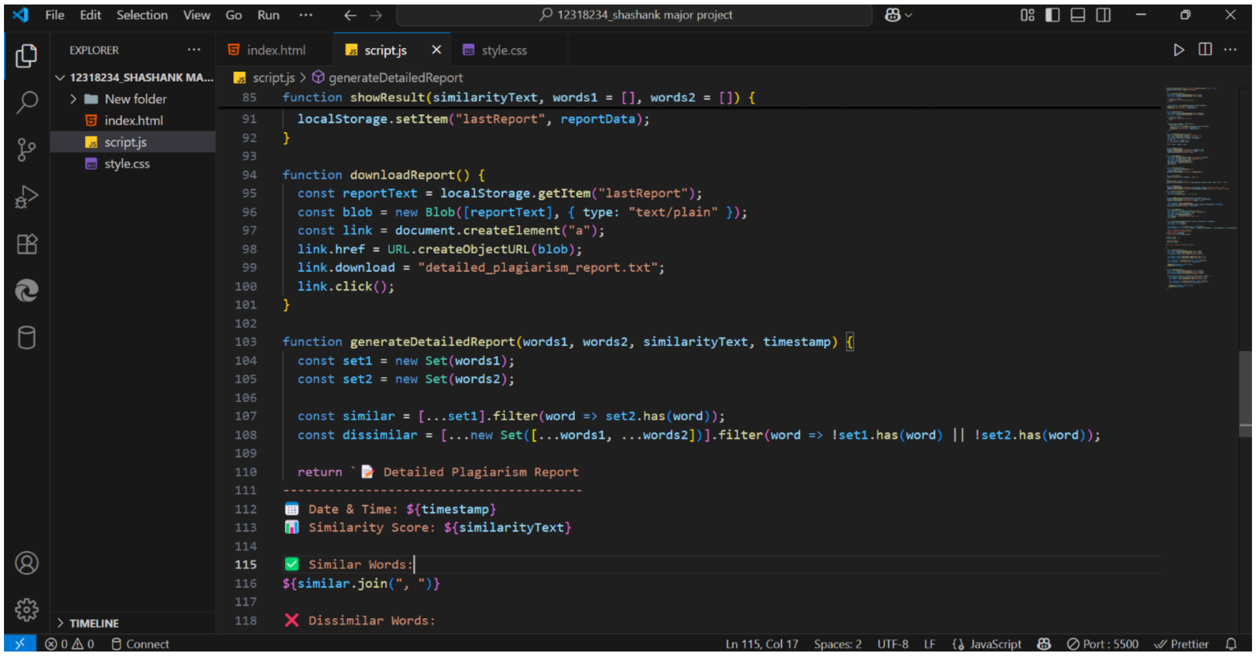
EXPLANATION:

This CSS file defines **how everything looks**.

**Highlights:**

- body: Uses a smooth font (Poppins), light background
- .typing-text: Animated typing effect for the title
- nav button: Stylish navigation buttons
- .text-area-wrapper: Holds two text boxes side-by-side for pasting text
- .dark-mode: A separate theme with dark background and light text
- #scrollBtn: A floating "back to top" button

JAVASCRIPT CODE AND EXPLANATION:


File Edit Selection View Go Run ···
12318234_shashank major project
EXPLORER
12318234_SHASHANK MA...
New folder
index.html
script.js
style.css
TIMELINE
script.js > generateDetailedReport
Ln 115, Col 17 Spaces: 2 UTF-8 LF JavaScript Port : 5500 Prettier
Connect

```javascript
function showResult(similarityText, words1 = [], words2 = []) {
  localStorage.setItem("lastReport", reportData);
}

function downloadReport() {
  const reportText = localStorage.getItem("lastReport");
  const blob = new Blob([reportText], { type: "text/plain" });
  const link = document.createElement("a");
  link.href = URL.createObjectURL(blob);
  link.download = "detailed_plagiarism_report.txt";
  link.click();
}

function generateDetailedReport(words1, words2, similarityText, timestamp) {
  const set1 = new Set(words1);
  const set2 = new Set(words2);

  const similar = [...set1].filter(word => set2.has(word));
  const dissimilar = [...new Set([...words1, ...words2])].filter(word => !set1.has(word) || !set2.has(word));

  return `📝 Detailed Plagiarism Report
----------------------------------------
📅 Date & Time: ${timestamp}
📊 Similarity Score: ${similarityText}

✅ Similar Words:
${similar.join(", ")}

❌ Dissimilar Words:
```

EXPLANATION:
This JavaScript file controls **how the app behaves** when users interact.
**Main Functions:**

**1. compareTextInputs()**
Compares similarity between two **pasted texts**:
js
const text1 = document.getElementById("text1").value;
const text2 = document.getElementById("text2").value;

- Converts texts to lowercase
- Splits into word arrays
- Calculates character-by-character similarity using stringSimilarity()
- Calls showResult() and saveHistory()

**2. compareTwoFiles()**
Reads **two uploaded .txt files**, gets their contents via:
js

Promise.all([file1.text(), file2.text()])
Then:
- Splits text into word arrays
- Uses stringSimilarity() to compute match %
- Shows result and saves in history

**3. stringSimilarity(str1, str2)**
A simple matching algorithm:
- Removes non-alphanumeric characters

- Compares characters at each position
- Calculates percentage of matching characters

js

```
return (same / length) * 100;
```

**4. showResult(similarityText, words1, words2)**
Displays result and also:

- Adds timestamp via getCurrentDateTime()
- Generates detailed report via generateDetailedReport()
- Saves the report to localStorage

**5. downloadReport()**
Downloads the **detailed similarity report** as a .txt file .

**6. saveHistory(entry)**
Appends a new entry to the history list (<ul id="historyList">) every time you run a comparison.

**7. showPage(page)**
Toggles between:

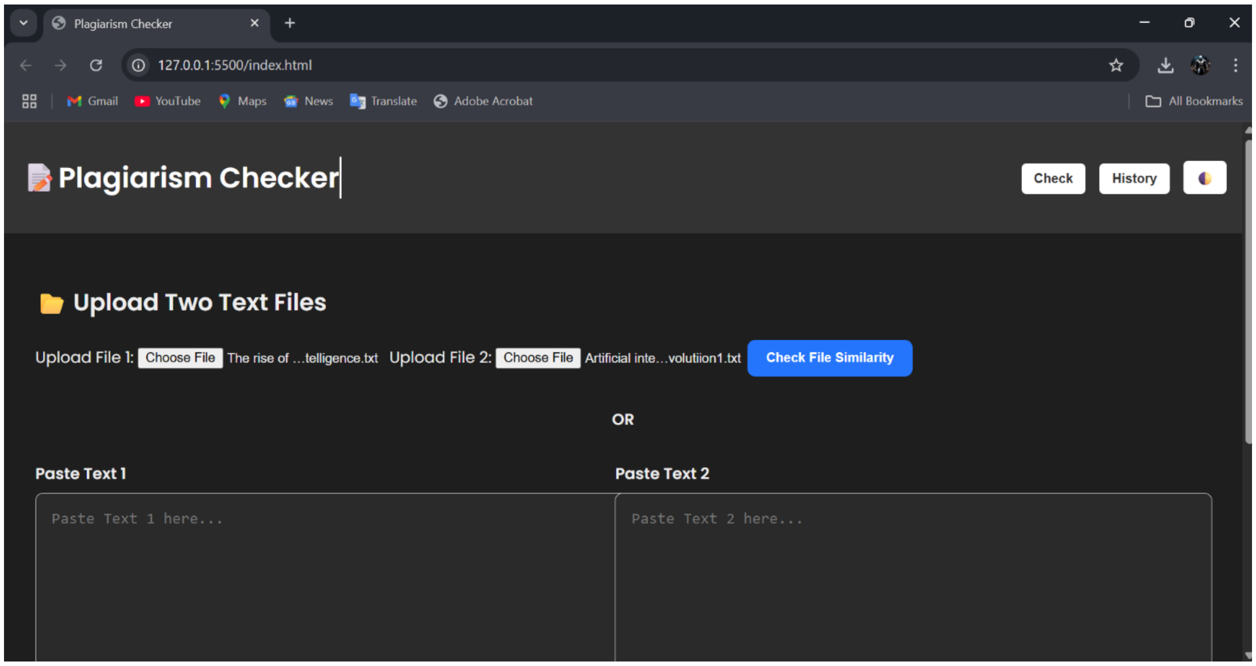
- Main comparison page
- History page

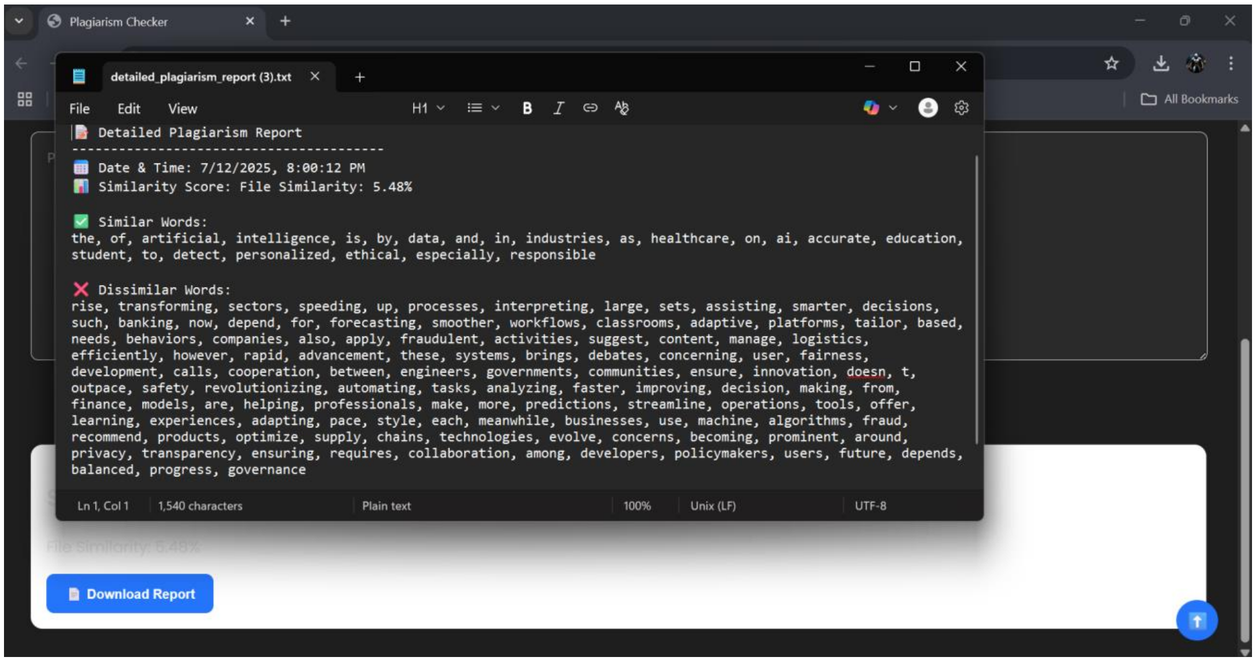
**8. Dark Mode Toggle**
Adds or removes the dark-mode class from body:
js

```
document.body.classList.toggle('dark-mode');
```

**9. Scroll to Top**
Shows a floating button when scrolled down. Clicking it scrolls smoothly to the top.

WEBSITE SCREENSHOT:

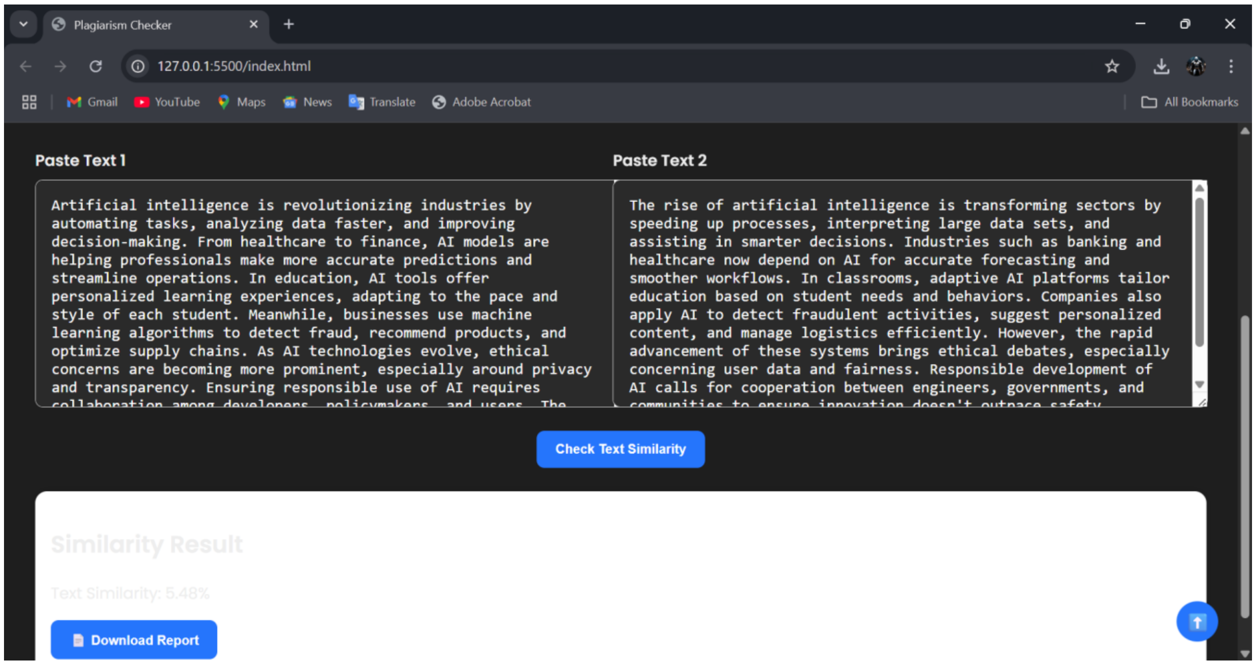




```
📄 Detailed Plagiarism Report
----------------------------------------
📅 Date & Time: 7/12/2025, 8:00:12 PM
📊 Similarity Score: File Similarity: 5.48%

✅ Similar Words:
the, of, artificial, intelligence, is, by, data, and, in, industries, as, healthcare, on, ai, accurate, education,
student, to, detect, personalized, ethical, especially, responsible

❌ Dissimilar Words:
rise, transforming, sectors, speeding, up, processes, interpreting, large, sets, assisting, smarter, decisions,
such, banking, now, depend, for, forecasting, smoother, workflows, classrooms, adaptive, platforms, tailor, based,
needs, behaviors, companies, also, apply, fraudulent, activities, suggest, content, manage, logistics,
efficiently, however, rapid, advancement, these, systems, brings, debates, concerning, user, fairness,
development, calls, cooperation, between, engineers, governments, communities, ensure, innovation, doesn, t,
outpace, safety, revolutionizing, automating, tasks, analyzing, faster, improving, decision, making, from,
finance, models, are, helping, professionals, make, more, predictions, streamline, operations, tools, offer,
learning, experiences, adapting, pace, style, each, meanwhile, businesses, use, machine, algorithms, fraud,
recommend, products, optimize, supply, chains, technologies, evolve, concerns, becoming, prominent, around,
privacy, transparency, ensuring, requires, collaboration, among, developers, policymakers, users, future, depends,
balanced, progress, governance
```



**Paste Text 1**

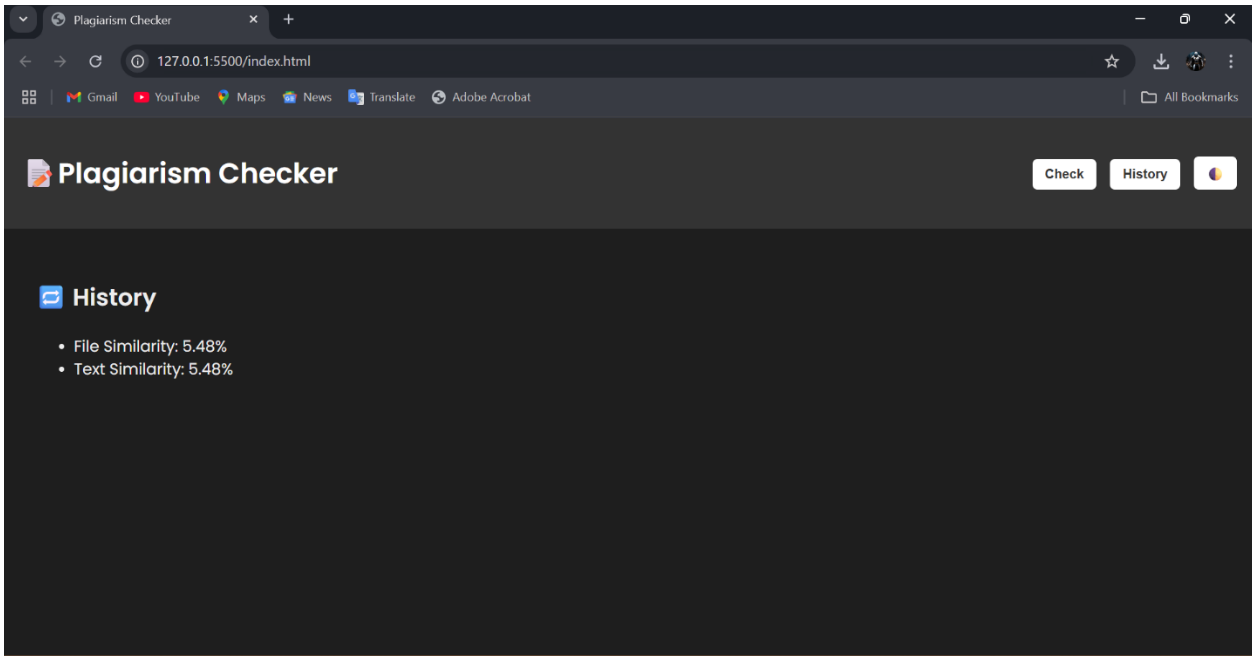
Artificial intelligence is revolutionizing industries by automating tasks, analyzing data faster, and improving decision-making. From healthcare to finance, AI models are helping professionals make more accurate predictions and streamline operations. In education, AI tools offer personalized learning experiences, adapting to the pace and style of each student. Meanwhile, businesses use machine learning algorithms to detect fraud, recommend products, and optimize supply chains. As AI technologies evolve, ethical concerns are becoming more prominent, especially around privacy and transparency. Ensuring responsible use of AI requires collaboration among developers, policymakers, and users. The

**Paste Text 2**

The rise of artificial intelligence is transforming sectors by speeding up processes, interpreting large data sets, and assisting in smarter decisions. Industries such as banking and healthcare now depend on AI for accurate forecasting and smoother workflows. In classrooms, adaptive AI platforms tailor education based on student needs and behaviors. Companies also apply AI to detect fraudulent activities, suggest personalized content, and manage logistics efficiently. However, the rapid advancement of these systems brings ethical debates, especially concerning user data and fairness. Responsible development of AI calls for cooperation between engineers, governments, and communities to ensure innovation doesn't outpace safety.

Similarity Result

Text Similarity: 5.48%



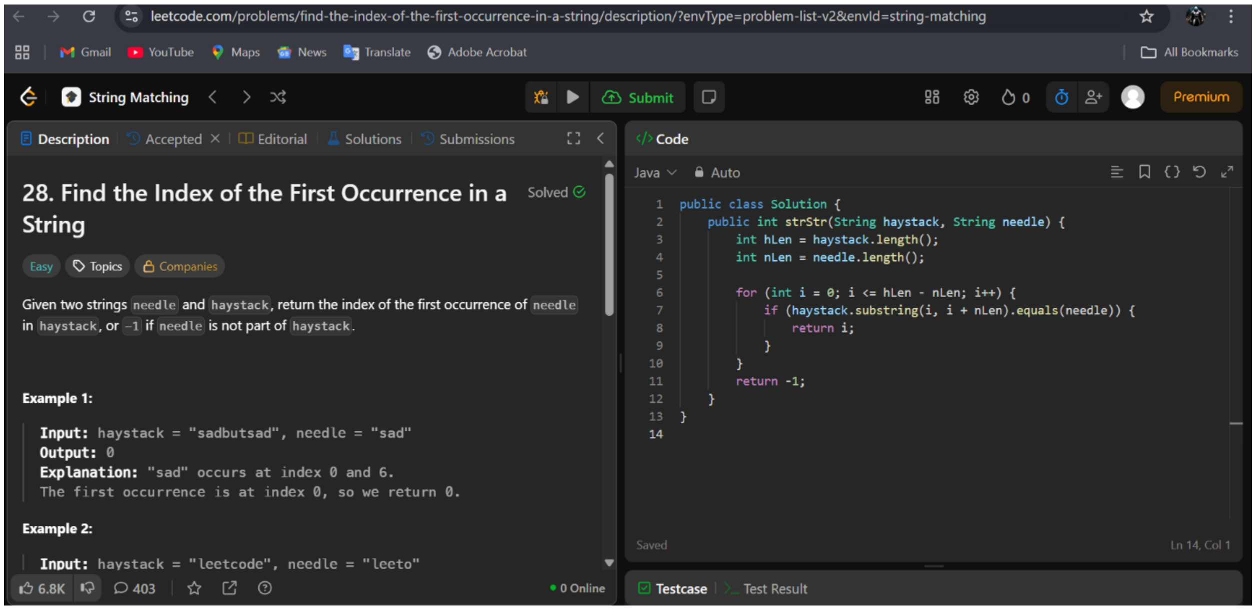
Plagiarism Checker

History

- File Similarity: 5.48%
- Text Similarity: 5.48%

GITHUB REPOSITORY LINK: https://github.com/sshashank13/Plagiarism-Checker

LEETCODE QUESTION SCREENSHOT RELATED WITH STRING MATCHING FOR BETTER UNDERSTANDING OF CONCEPT:


leetcode.com/problems/find-the-index-of-the-first-occurrence-in-a-string/description/?envType=problem-list-v2&envId=string-matching
Gmail YouTube Maps News Translate Adobe Acrobat All Bookmarks
String Matching
Submit
Premium
Description Accepted Editorial Solutions Submissions
Code
Java Auto
Easy Topics Companies
Solved
6.8K 403
0 Online
Saved
Ln 14, Col 1
Testcase Test Result

28. Find the Index of the First Occurrence in a String

Given two strings `needle` and `haystack`, return the index of the first occurrence of `needle` in `haystack`, or `-1` if `needle` is not part of `haystack`.

**Example 1:**

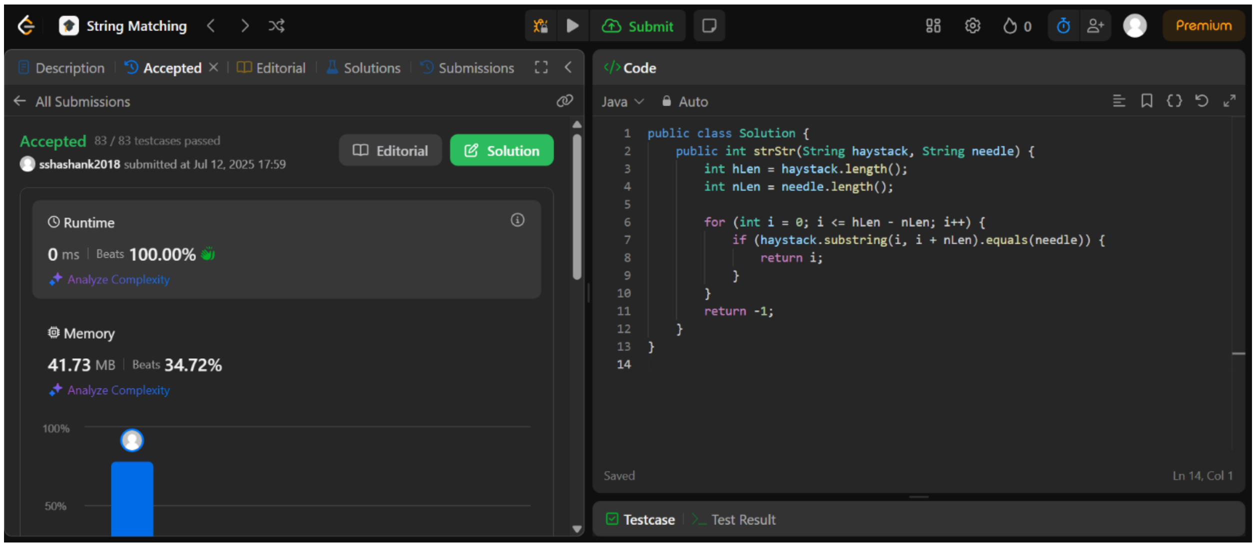
```
Input: haystack = "sadbutsad", needle = "sad"
Output: 0
Explanation: "sad" occurs at index 0 and 6.
The first occurrence is at index 0, so we return 0.
```

**Example 2:**

```
Input: haystack = "leetcode", needle = "leeto"
```

```java
public class Solution {
    public int strStr(String haystack, String needle) {
        int hLen = haystack.length();
        int nLen = needle.length();

        for (int i = 0; i <= hLen - nLen; i++) {
            if (haystack.substring(i, i + nLen).equals(needle)) {
                return i;
            }
        }
        return -1;
    }
}
```



Accepted 83 / 83 testcases passed

sshashank2018 submitted at Jul 12, 2025 17:59

Runtime

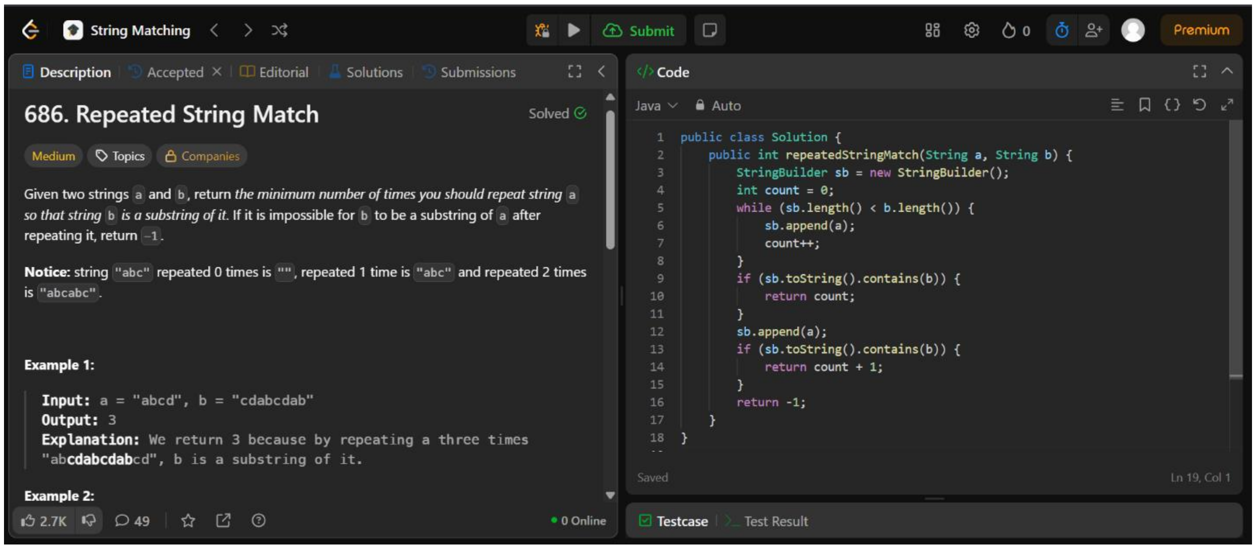
0 ms | Beats 100.00%

Memory

41.73 MB | Beats 34.72%

```java
public class Solution {
    public int strStr(String haystack, String needle) {
        int hLen = haystack.length();
        int nLen = needle.length();

        for (int i = 0; i <= hLen - nLen; i++) {
            if (haystack.substring(i, i + nLen).equals(needle)) {
                return i;
            }
        }
        return -1;
    }
}
```



String Matching

Description | Accepted | Editorial | Solutions | Submissions

686. Repeated String Match

Solved

Medium | Topics | Companies

Given two strings `a` and `b`, return *the minimum number of times you should repeat string* `a` *so that string* `b` *is a substring of it*. If it is impossible for `b` to be a substring of `a` after repeating it, return `-1`.

**Notice:** string `"abc"` repeated 0 times is `""`, repeated 1 time is `"abc"` and repeated 2 times is `"abcabc"`.

**Example 1:**

```
Input: a = "abcd", b = "cdabcdab"
Output: 3
Explanation: We return 3 because by repeating a three times "abcdabcdabcd", b is a substring of it.
```

**Example 2:**

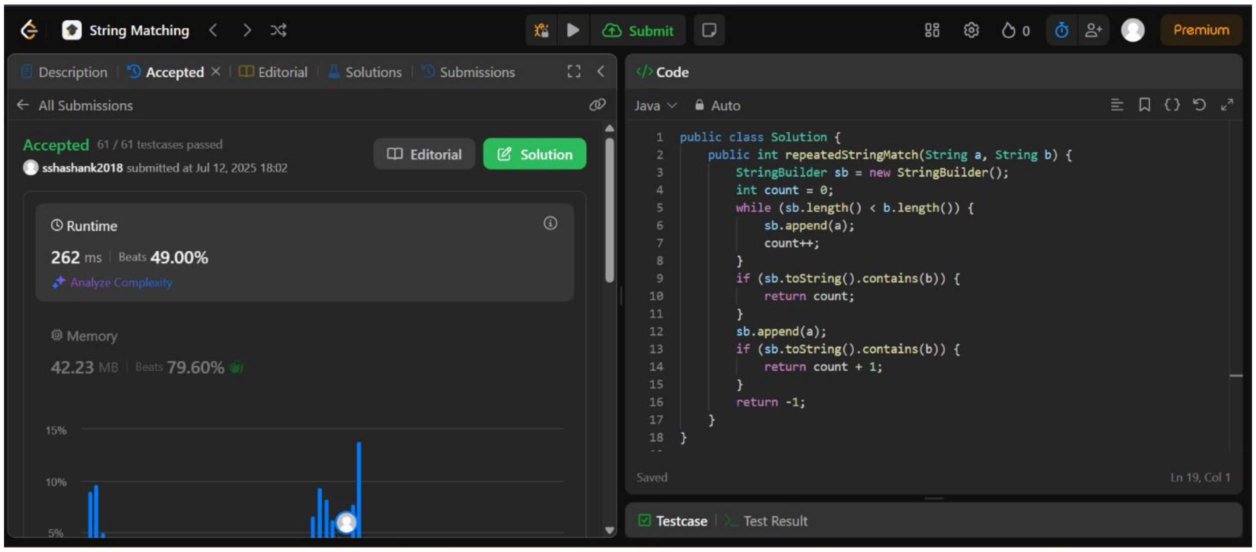
2.7K | 49 | 0 Online

Code — Java — Auto

```java
public class Solution {
    public int repeatedStringMatch(String a, String b) {
        StringBuilder sb = new StringBuilder();
        int count = 0;
        while (sb.length() < b.length()) {
            sb.append(a);
            count++;
        }
        if (sb.toString().contains(b)) {
            return count;
        }
        sb.append(a);
        if (sb.toString().contains(b)) {
            return count + 1;
        }
        return -1;
    }
}
```

Saved — Ln 19, Col 1

Testcase | Test Result



String Matching

Description | Accepted | Editorial | Solutions | Submissions

All Submissions

Accepted 61 / 61 testcases passed

sshashank2018 submitted at Jul 12, 2025 18:02

Editorial | Solution

Runtime

262 ms | Beats 49.00%

Analyze Complexity

Memory

42.23 MB | Beats 79.60%

Code — Java — Auto

```java
public class Solution {
    public int repeatedStringMatch(String a, String b) {
        StringBuilder sb = new StringBuilder();
        int count = 0;
        while (sb.length() < b.length()) {
            sb.append(a);
            count++;
        }
        if (sb.toString().contains(b)) {
            return count;
        }
        sb.append(a);
        if (sb.toString().contains(b)) {
            return count + 1;
        }
        return -1;
    }
}
```

Saved — Ln 19, Col 1

Testcase | Test Result