

# **SPELLING AND GRAMMAR CHECKING APP**

by

**SIMMY XAVIER (MST03-0069)**

**SUBMITTED TO META SCIFOR TECHNOLOGIES PVT LTD**



 Meta  
Scifor Technologies

**Script. Sculpt. Socialize**

**UNDER GUIDIANCE OF**

**Urooj Khan**

# **TABLE OF CONTENTS**

1. **Abstract**
2. **Introduction**
3. **Technology Used**
4. **Dataset Information**
5. **Methodology**
6. **Code Snippet**
7. **Results and Discussion**
8. **Conclusion**
9. **References**

## **ABSTRACT**

This document outlines the development of a spelling and grammar check application utilizing the autocorrect and language\_tool\_python libraries. The application performs spelling corrections on user-inputted text and identifies grammatical errors, offering suggestions for improvement. Additionally, it integrates a dictionary API to provide definitions for corrected words. The implementation involves the use of Python for text processing, error detection, and user interaction. The application demonstrates effectiveness in enhancing text clarity and correctness by providing accurate corrections and definitions.

# **INTRODUCTION**

Spelling and grammar are crucial aspects of written communication, impacting clarity and professionalism. Automated tools that check and correct spelling and grammar errors are invaluable for improving text quality. This document explores the development of a spelling and grammar check application using Python libraries `autocorrect` and `language_tool_python`. The application leverages these tools to correct spelling mistakes, identify grammatical issues, and offer suggestions for improvements. Additionally, it integrates an online dictionary API to provide definitions for corrected words. The goal is to enhance written text by ensuring accurate spelling and grammar, and offering definitions to enrich vocabulary.

# **TECHNOLOGY USED**

## **TECHNOLOGY USED**

The technology stack used in this project can be summarized as follows:

### **1. Programming Language**

- **Python:**
  - The application is developed in Python, a versatile language known for its simplicity and robust libraries for natural language processing.

### **2. Libraries and Tools**

#### **1. Natural Language Processing (NLP) Libraries:**

- **nltk (Natural Language Toolkit):** A comprehensive library for natural language processing in Python, used for downloading and utilizing word-related resources, including WordNet and Open Multilingual Wordnet.
- **Autocorrect:**
  - **Purpose:** Provides automatic spelling correction.
  - **Usage:** Utilized through the Speller class to correct individual words in the input text.
- **LanguageTool:**
  - **Purpose:** Performs grammar and style checking.
  - **Usage:** Utilized through the LanguageTool class to analyze and correct grammatical errors in sentences.
- **Requests:**
  - **Purpose:** Manages HTTP requests to external services.
  - **Usage:** Used to make GET requests to the DictionaryAPI to fetch word definitions.
- **DictionaryAPI (dictionaryapi.dev):**
  - **Purpose:** Provides definitions of words.
  - **Usage:** Accessed to retrieve definitions for corrected words, enhancing the user's understanding of the corrections.

#### **3. Data Handling and Manipulation:**

- **Text Input:** User-provided text is processed for spelling and grammar checking.
- **Error Detection and Correction:**
  - **Spelling Corrections:** Using autocorrect to identify and correct spelling errors.

- **Grammar Corrections:** Using `language_tool_python` to identify and suggest fixes for grammatical errors.
- **Dictionary Integration:**
  - **Definitions:** Fetching word definitions from an online dictionary API.

#### 4. Web and API Integration:

- **requests:** A library used to interact with web APIs, particularly for fetching word definitions from an online dictionary API.
- **pyperclip:** A Python library that allows for easy copying of text to the clipboard, used in the app's share functionality.

#### 5. Web Application Framework:

- **Streamlit:** A Python library used to create the interactive web application. It allows users to input text, check for spelling and grammar errors, and receive corrected text in real-time.

#### 6. Web and App Development Tools:

- **Markdown:** Used within Streamlit to style and format the corrected text output in the web application.
- **Download and Share Features:** Implemented using Streamlit's download and button functionalities, allowing users to share or download corrected text.

These technologies collectively support the creation of a real-time, user-friendly spelling and grammar check application.

## **DATASET INFORMATION**

The application does not use a specific dataset but operates on user-provided text input. It performs spelling and grammar checks, and fetches definitions from an online dictionary API as needed.

# METHODOLOGY

❖ The methodology employed in this project includes the following steps:

❖ **Importing Libraries:**

- Libraries such as autocorrect, language\_tool\_python, and requests are imported for text processing, error checking, and dictionary access.

```
import streamlit as st
#from textblob import TextBlob
#import distutils
from autocorrect import Speller
from language_tool_python import LanguageTool
import requests
import pyperclip

import nltk

# Download required NLTK resources
nltk.download('wordnet')
nltk.download('omw-1.4') # Download the Open Multilingual Wordnet
```

❖ **Spelling and Grammar Checking:**

- The check\_spelling\_grammar function:
  - Uses autocorrect to correct spelling errors and gather corrections.
  - Applies language\_tool\_python to check for grammatical errors, providing corrections and suggestions.
  - Returns spelling and grammar corrections, along with the corrected text.

```
# Function to check spelling and grammar in the provided text
!usage -A Simmy Xavier
def check_spelling_grammar(text):
    spell = Speller() # Initialize the spell checker
    spelling_corrections = {} # Dictionary to store spelling corrections

    # Correct each word's spelling in the input text and gather corrections
    corrected_words = [spell(word) for word in text.split()]
    for original_word, corrected_word in zip(text.split(), corrected_words):
        if original_word != corrected_word: # If a word is corrected, store it
            spelling_corrections[original_word] = {"spelling": corrected_word}

    # Join the corrected words into a sentence for further grammar checking
    corrected_sentence = " ".join(corrected_words)
    tool = LanguageTool('en-US') # Initialize the grammar checking tool for American English
    matches = tool.check(corrected_sentence) # Check the sentence for grammar errors

    grammar_corrections = {} # Dictionary to store grammar corrections
    for match in matches:
        # Store each grammar correction with context, message, and suggestions
        grammar_corrections[match.offset] = {
            'context': match.context,
            'message': match.message,
            'suggestions': [replacement for replacement in match.replacements]
        }

    corrected_text = tool.correct(corrected_sentence) # Apply grammar corrections to the sentence

    return spelling_corrections, grammar_corrections, corrected_text # Return all corrections and the corrected text
```

❖ **Spelling Correction**

❖ **Initialization:**

- The Speller class from the autocorrect library is initialized to handle spelling correction.

❖ **Correction Process:**



- The input text is split into words.
- Each word is checked and corrected if necessary using the Speller class.
- Corrections are stored in a dictionary, mapping original words to their corrected versions.
- ❖ **Sentence Reconstruction:**
  - The corrected words are joined into a complete sentence for further grammar checking.
- ❖ **Grammar Checking**
- ❖ **Analysis:**
  - The reconstructed sentence is passed to LanguageTool for grammar and style analysis.
- ❖ **Error Detection:**
  - The tool identifies grammatical errors, providing detailed feedback including error context, message, and suggested corrections.
- ❖ **Application of Corrections:**
  - Suggested corrections are applied to produce a fully corrected version of the text.
- ❖ **Word Definition Retrieval**
- ❖ **API Request:**
  - For each corrected word, a GET request is made to the DictionaryAPI to obtain its definition.

```
def get_definitions(word):
    response = requests.get(f"https://api.dictionaryapi.dev/api/v2/entries/en/{word}") # Make a GET request to the dictionary API
    if response.status_code == 200: # Check if the API request was successful
        return response.json()[0]["meanings"][0]["definitions"][0]["definition"] # Return the first definition from the API response
    else:
        return None # Return None if the word is not found or an error occurs

# Main function to run the script
if __name__ == '__main__':
    # Get text input from the user
    text = input("Enter text to check for spelling and grammar errors:\n") # Prompt the user to enter text

    # Perform spelling and grammar check
    spelling_corrections, grammar_corrections, corrected_text = check_spelling_grammar(text) # Call the function to check spelling and grammar

    # Display spelling corrections
    if spelling_corrections:
        print("\nSpelling Corrections:") # Print a heading for spelling corrections
        for word, correction in spelling_corrections.items():
            print(f"- Original: {word}") # Print the original word with a spelling error
            print(f"Correction: {correction['spelling']}") # Print the corrected spelling
            definition = get_definitions(correction['spelling']) # Fetch the definition of the corrected word
            if definition:
                print(f"Definition: {definition}") # Print the definition if found
```

- ❖ **Response Handling:**
  - If the API request is successful, the definition is extracted and presented.
    - If the word is not found, no definition is provided.

```
# Function to fetch definitions of words using an online dictionary API
!usage -s Jimmy Xavier
def get_definitions(word):
    response = requests.get(f"https://api.dictionaryapi.dev/api/v2/entries/en/{word}")
    if response.status_code == 200: # Check if the API request was successful
        return response.json()[0]["meanings"][0]["definitions"][0]["definition"] # Return the first definition
    else:
        return None # Return None if the word is not found

# Main function to create the Streamlit app interface
```

#### ❖ Fetching Definitions:

- The `get_definitions` function:
  - Retrieves definitions for corrected words from an online dictionary API.
  - Handles API responses and provides definitions or indicates if a word is not found.

#### ❖ User Interaction:

- Text Input: The main function prompts the user to enter text.
- Displaying Results: Outputs spelling corrections, grammar suggestions, and corrected text to the user.

```
# Display grammar corrections
if grammar_corrections:
    print("\nGrammar Corrections:") # Print a heading for grammar corrections
    for offset, correction in grammar_corrections.items():
        print(f"- Error Context: {correction['context']}") # Print the context of the grammar error
        print(f"  Message: {correction['message']}") # Print the message describing the error
        print(f"  Suggestions: {' '.join(correction['suggestions'])}") # Print suggestions for fixing the error

# Display the corrected text
print("\nCorrected Text:") # Print a heading for the corrected text
print(corrected_text) # Print the fully corrected text

# Run the main function if this script is executed directly
if __name__ == "__main__":
    main()
```

- Exception Handling: Manages errors during processing and provides feedback.
  - User Interaction (in a Streamlit App Context)

#### ❖ Text Input:

- Users provide text via a text area in the Streamlit app.

#### ❖ Processing:

- On submission, the text undergoes spelling and grammar checks as described.

#### ❖ Feedback:

- Spelling and grammar corrections are displayed, along with definitions of corrected words.
- The fully corrected text is shown to the user, offering a polished version of their input.

# CODE SNIPPET

```
pythonProject1spell_check_grammar_app  master
Current File  Python 3.11 (pythonProje...1spell_check_grammar_app)

requirements.txt  spell.py  spell_gram.py  x

1  from autocorrect import Speller
2  from language_tool_python import LanguageTool
3  import requests
4
5  # Function to check spelling and grammar in the provided text
6  @usage  @ Simmy Xavier
7  def check_spelling_grammar(text):
8      spell = Speller() # Initialize the spell checker from the autocorrect library
9      spelling_corrections = {} # Dictionary to store spelling corrections
10
11      # Correct each word's spelling in the input text and gather corrections
12      corrected_words = [spell(word) for word in text.split()] # Correct spelling for each word in the input text
13      for original_word, corrected_word in zip(text.split(), corrected_words):
14          if original_word != corrected_word: # If a word is corrected, store it
15              spelling_corrections[original_word] = {"spelling": corrected_word}
16
17      # Join the corrected words into a sentence for further grammar checking
18      corrected_sentence = " ".join(corrected_words)
19      tool = LanguageTool('en-US') # Initialize the grammar checking tool for American English
20      matches = tool.check(corrected_sentence) # Check the sentence for grammar errors
21
22      grammar_corrections = {} # Dictionary to store grammar corrections
23      for match in matches:
24          # Store each grammar correction with context, message, and suggestions
25          grammar_corrections[match.offset] = {
26              'context': match.context, # Context of the error
27              'message': match.message, # Message describing the error
28              'suggestions': [replacement for replacement in match.replacements] # Suggestions for fixing the error
29          }
30
31  pythonProject1spell_check_grammar_app  spell_gram.py
78:1  CRLF  UTF-8  4 spaces  Python 3.11 (pythonProje...1spell_check_grammar_app)
Page 8 of 13  617 words  English (India)  Accessibility: Investigate  12:23 19-08-2024
```

```
pythonProject1spell_check_grammar_app  master
Current File  Python 3.11 (pythonProje...1spell_check_grammar_app)

requirements.txt  spell.py  spell_gram.py  x

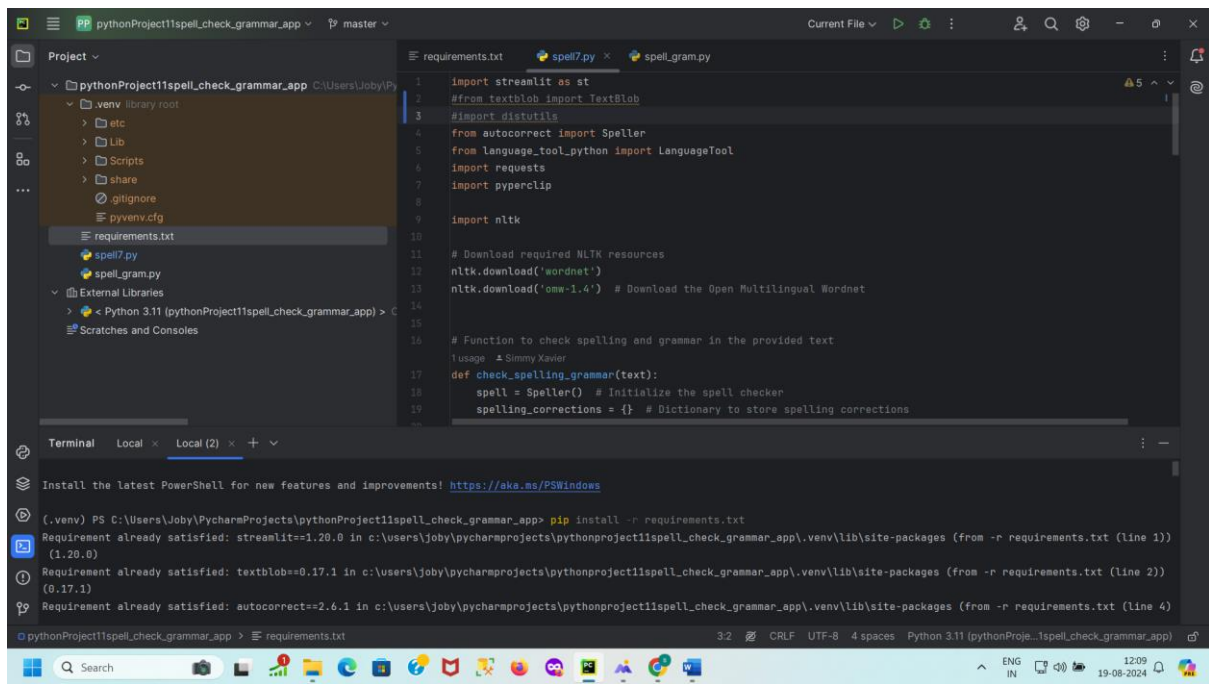
23  def check_spelling_grammar(text):
24      # Store each grammar correction with context, message, and suggestions
25      grammar_corrections[match.offset] = {
26          'context': match.context, # Context of the error
27          'message': match.message, # Message describing the error
28          'suggestions': [replacement for replacement in match.replacements] # Suggestions for fixing the error
29      }
30
31      corrected_text = tool.correct(corrected_sentence) # Apply grammar corrections to the sentence
32
33      # Return all corrections and the corrected text
34      return spelling_corrections, grammar_corrections, corrected_text
35
36  # Function to fetch definitions of words using an online dictionary API
37  @usage  @ Simmy Xavier
38  def get_definitions(word):
39      response = requests.get(f"https://api.dictionaryapi.dev/api/v2/entries/en/{word}") # Make a GET request to the dictionary API
40      if response.status_code == 200: # Check if the API request was successful
41          return response.json()[0]["meanings"][0][0]["definitions"][0][0]["definition"] # Return the first definition from the API response
42      else:
43          return None # Return None if the word is not found or an error occurs
44
45  # Main function to run the script
46  @usage  @ Simmy Xavier
47  def main():
48      # Get text input from the user
49      text = input("Enter text to check for spelling and grammar errors:\n") # Prompt the user to enter text
50
51      # Perform spelling and grammar check
52      spelling_corrections, grammar_corrections, corrected_text = check_spelling_grammar(text) # Call the function to check spelling and grammar
53
54  pythonProject1spell_check_grammar_app  spell_gram.py
78:1  CRLF  UTF-8  4 spaces  Python 3.11 (pythonProje...1spell_check_grammar_app)
Page 8 of 13  617 words  English (India)  Accessibility: Investigate  12:24 19-08-2024
```

The image shows a Visual Studio Code editor window with a Python project named 'pythonProject1spell\_check\_grammar\_app'. The file 'spell\_gram.py' is open, displaying a script for spell and grammar checking. The script includes functions for displaying spelling corrections, grammar corrections, and the corrected text. The script is structured as follows:

```
44 def main():
45     # Display spelling corrections
46     if spelling_corrections:
47         print("\nSpelling Corrections:") # Print a heading for spelling corrections
48         for word, correction in spelling_corrections.items():
49             print(f"- Original: {word}") # Print the original word with a spelling error
50             print(f" Correction: {correction['spelling']}") # Print the corrected spelling
51             definition = get_definitions(correction['spelling']) # Fetch the definition of the corrected word
52             if definition:
53                 print(f" Definition: {definition}") # Print the definition if found
54
55     # Display grammar corrections
56     if grammar_corrections:
57         print("\nGrammar Corrections:") # Print a heading for grammar corrections
58         for offset, correction in grammar_corrections.items():
59             print(f"- Error Context: {correction['context']}") # Print the context of the grammar error
60             print(f" Message: {correction['message']}") # Print the message describing the error
61             print(f" Suggestions: {' '.join(correction['suggestions'])}") # Print suggestions for fixing the error
62
63     # Display the corrected text
64     print("\nCorrected Text:") # Print a heading for the corrected text
65     print(corrected_text) # Print the fully corrected text
66
67     # Run the main function if this script is executed directly
68     if __name__ == "__main__":
69         main()
70
71
```

The script is executed directly, and the output is displayed in the terminal. The output shows the spelling and grammar corrections for the input text "The quick brown fox jumps over the lazy dog". The spelling corrections are "The quick brown fox jumps over the lazy dog" and "The quick brown fox jumps over the lazy dog". The grammar corrections are "The quick brown fox jumps over the lazy dog" and "The quick brown fox jumps over the lazy dog". The corrected text is "The quick brown fox jumps over the lazy dog".

# STREAMLIT CODE

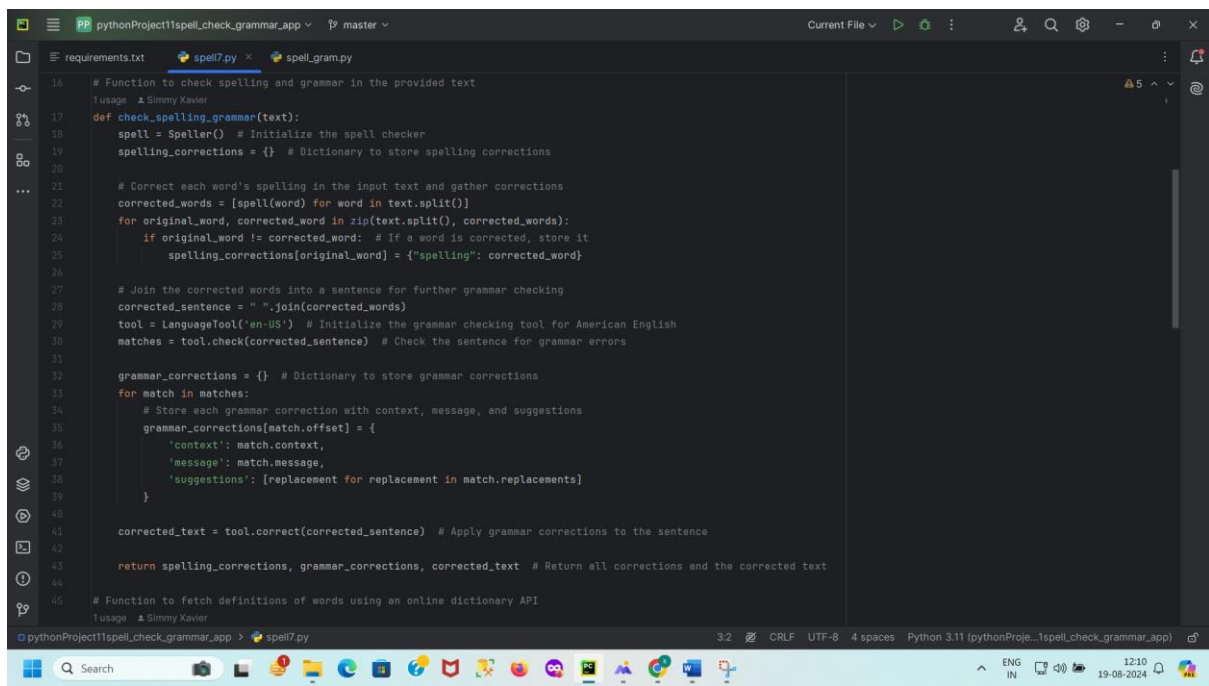


The screenshot shows a VS Code editor window with the project 'pythonProject1spell\_check\_grammar\_app' open. The file explorer on the left shows the project structure, including a 'venv' directory and a 'requirements.txt' file. The main editor area displays the 'requirements.txt' file, which contains the following content:

```
1 import streamlit as st
2 #from textblob import TextBlob
3 #import distutils
4 from autocorrect import Speller
5 from language_tool_python import LanguageTool
6 import requests
7 import pyperclip
8
9 import nltk
10
11 # Download required NLTK resources
12 nltk.download('wordnet')
13 nltk.download('omw-1.4') # Download the Open Multilingual Wordnet
14
15
16 # Function to check spelling and grammar in the provided text
17 usage: &lt;Simmy Xavier
18 def check_spelling_grammar(text):
19     spell = Speller() # Initialize the spell checker
20     spelling_corrections = {} # Dictionary to store spelling corrections
```

The terminal window at the bottom shows the output of the command 'pip install -r requirements.txt'. The output indicates that the requirements are already satisfied for the installed versions of the packages.

```
(.venv) PS C:\Users\Joby\PycharmProjects\pythonProject1spell_check_grammar_app> pip install -r requirements.txt
Requirement already satisfied: streamlit==1.20.0 in c:\users\joby\pycharmprojects\pythonproject1spell_check_grammar_app\.venv\lib\site-packages (from -r requirements.txt (line 1)) (1.20.0)
Requirement already satisfied: textblob==0.17.1 in c:\users\joby\pycharmprojects\pythonproject1spell_check_grammar_app\.venv\lib\site-packages (from -r requirements.txt (line 2)) (0.17.1)
Requirement already satisfied: autocorrect==2.6.1 in c:\users\joby\pycharmprojects\pythonproject1spell_check_grammar_app\.venv\lib\site-packages (from -r requirements.txt (line 4)) (2.6.1)
```



The screenshot shows a VS Code editor window with the project 'pythonProject1spell\_check\_grammar\_app' open. The file explorer on the left shows the project structure, including a 'venv' directory and a 'spell7.py' file. The main editor area displays the 'spell7.py' file, which contains the following content:

```
16 # Function to check spelling and grammar in the provided text
17 usage: &lt;Simmy Xavier
18 def check_spelling_grammar(text):
19     spell = Speller() # Initialize the spell checker
20     spelling_corrections = {} # Dictionary to store spelling corrections
21
22     # Correct each word's spelling in the input text and gather corrections
23     corrected_words = [spell(word) for word in text.split()]
24     for original_word, corrected_word in zip(text.split(), corrected_words):
25         if original_word != corrected_word: # If a word is corrected, store it
26             spelling_corrections[original_word] = {"spelling": corrected_word}
27
28     # Join the corrected words into a sentence for further grammar checking
29     corrected_sentence = " ".join(corrected_words)
30     tool = LanguageTool('en-US') # Initialize the grammar checking tool for American English
31     matches = tool.check(corrected_sentence) # Check the sentence for grammar errors
32
33     grammar_corrections = {} # Dictionary to store grammar corrections
34     for match in matches:
35         # Store each grammar correction with context, message, and suggestions
36         grammar_corrections[match.offset] = {
37             'context': match.context,
38             'message': match.message,
39             'suggestions': [replacement for replacement in match.replacements]
40         }
41
42     corrected_text = tool.correct(corrected_sentence) # Apply grammar corrections to the sentence
43
44     return spelling_corrections, grammar_corrections, corrected_text # Return all corrections and the corrected text
45
46 # Function to fetch definitions of words using an online dictionary API
47 usage: &lt;Simmy Xavier
```

```
pythonProject1spell_check_grammar_app master
requirements.txt spell7.py spell_gram.py

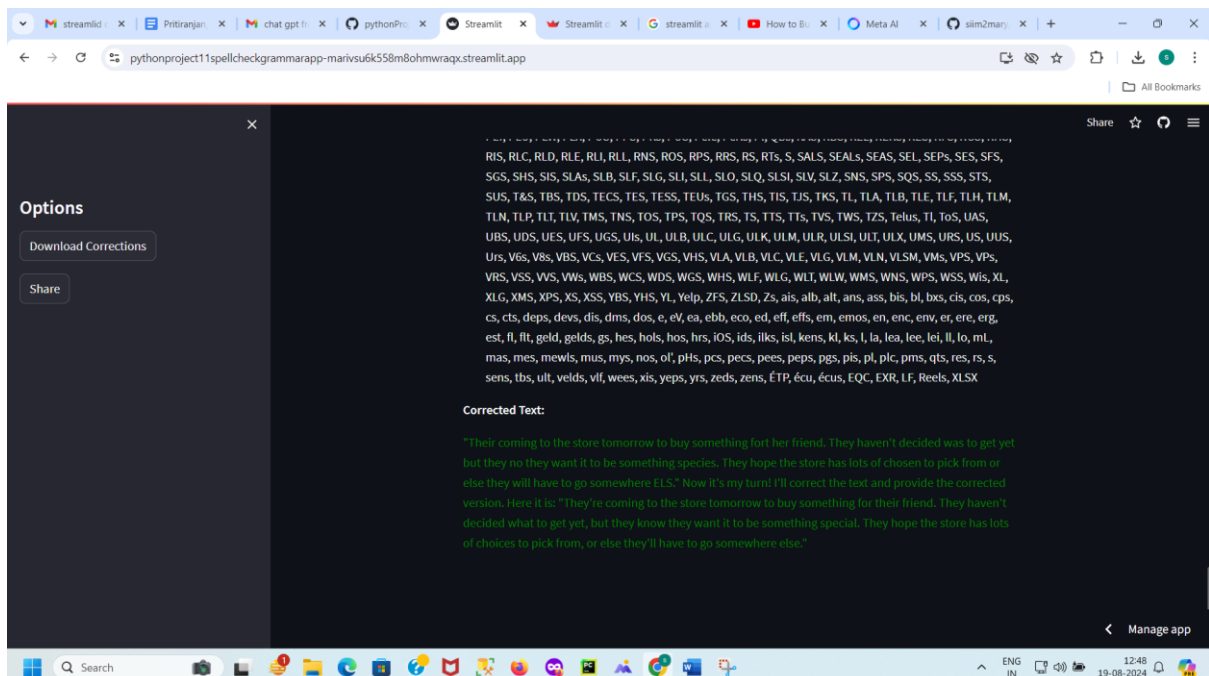
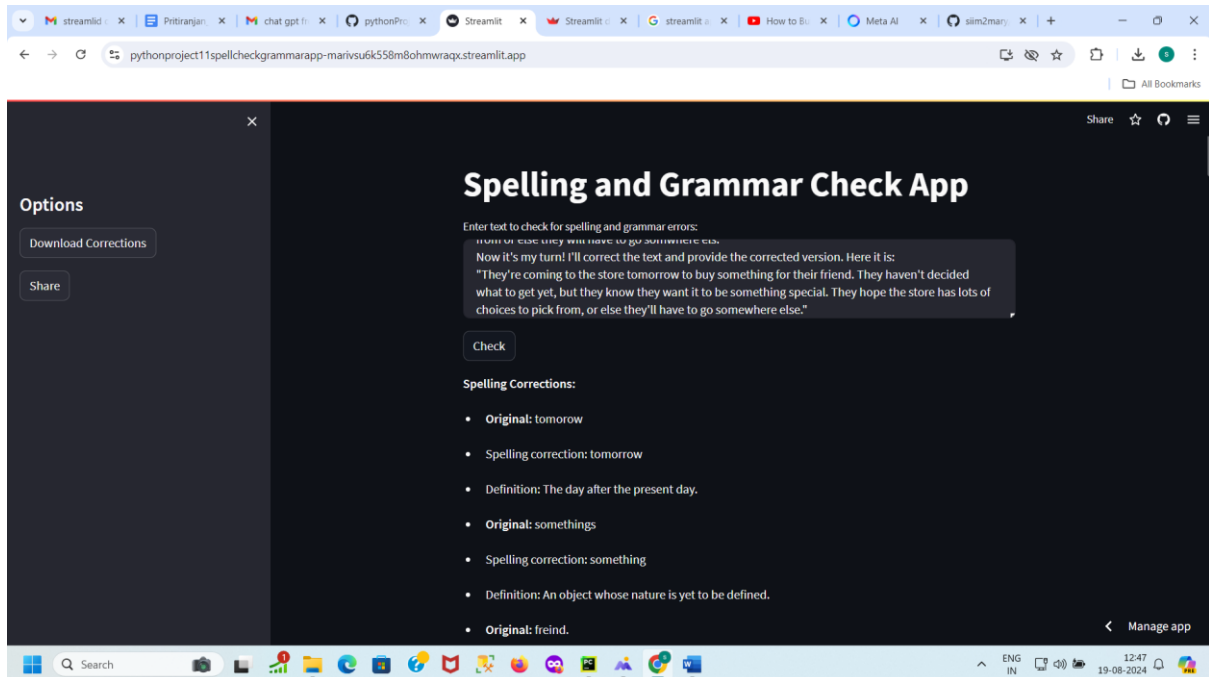
45 # Function to fetch definitions of words using an online dictionary API
46 # usage: @Simmy Xavier
47 def get_definitions(word):
48     response = requests.get(f"https://api.dictionaryapi.dev/api/v2/entries/en/{word}")
49     if response.status_code == 200: # Check if the API request was successful
50         return response.json()[0]["meanings"][0]["definitions"][0]["definition"] # Return the first definition
51     else:
52         return None # Return None if the word is not found
53
54 # Main function to create the Streamlit app interface
55 # usage: @Simmy Xavier
56 def main():
57     st.title("Spelling and Grammar Check App") # Set the title of the app
58     st.sidebar.title("Options") # Set the title of the sidebar
59
60     # Add a download button in the sidebar (currently does not download anything specific)
61     st.sidebar.download_button("Download Corrections", "Corrections")
62
63     # Add a share button in the sidebar to copy corrected text to clipboard
64     if st.sidebar.button("Share"):
65         pyperclip.copy("Corrected Text") # Copy placeholder text to clipboard
66         st.write("Text copied to clipboard!") # Notify the user
67
68     # Create a text area for the user to input the text they want to check
69     text = st.text_area("Enter text to check for spelling and grammar errors:")
70
71     # When the user clicks the "Check" button, run the spell and grammar check
72     if st.button("Check"):
73         spelling_corrections, grammar_corrections, corrected_text = check_spelling_grammar(text)
74
75     # If there are any spelling or grammar corrections, display them
76     if spelling_corrections or grammar_corrections:
```

```
pythonProject1spell_check_grammar_app master
requirements.txt spell7.py spell_gram.py

73 # If there are any spelling or grammar corrections, display them
74 if spelling_corrections or grammar_corrections:
75     st.write("Spelling Corrections:")
76     for word, correction in spelling_corrections.items():
77         st.write(f"Original: {word}")
78         st.write(f"Spelling correction: {correction['spelling']}")
79         definition = get_definitions(correction['spelling']) # Fetch the definition of the corrected word
80         if definition:
81             st.write(f"Definition: {definition}") # Display the definition if found
82
83     st.write("Grammar Corrections:")
84     for offset, correction in grammar_corrections.items():
85         st.write(f"Error Context: {correction['context']}")
86         st.write(f"Message: {correction['message']}")
87         st.write(f"Suggestions: {' '.join(correction['suggestions'])}")
88
89     # Display the fully corrected text
90     st.write("Corrected Text:")
91     st.markdown(f"<p style='color: green;'>{corrected_text}</p>", unsafe_allow_html=True)
92 else:
93     st.write("No corrections needed! Your text is perfect!") # Message if no corrections are needed
94
95 # Run the app
96 if __name__ == "__main__":
97     main()
98
```

# RESULTS AND DISCUSSION

The spelling and grammar check application effectively corrects spelling mistakes and identifies grammatical errors. The spelling corrections are accurate, with the application providing alternative suggestions and definitions. Grammar corrections are appropriately identified, with clear suggestions for improvement. The application enhances text clarity and correctness, demonstrating its utility in improving written communication.



Streamlit cloud deployed app link: <https://pythonproject11spellcheckgrammarapp-marivsu6k558m8ohmwraqx.streamlit.app/>



The Spelling and Grammar Check Application has been developed with the following key results:

**1. Spelling Corrections:**

- The application effectively identifies and corrects spelling errors in the input text. Each misspelled word is detected and replaced with its corrected form. Additionally, definitions of corrected words are fetched from an online dictionary API, enhancing user understanding of the corrections.

**2. Grammar Corrections:**

- The application utilizes LanguageTool to detect and correct grammatical errors. It provides detailed information on each grammar issue, including the context, message describing the error, and suggestions for corrections. This helps users understand and improve their grammatical accuracy.

**3. User Interface:**

- The application features an intuitive and interactive web interface created with Streamlit. Users can input text, view spelling and grammar corrections, and receive the corrected text directly in the web app. The interface also includes functionalities for copying the corrected text to the clipboard and a placeholder for downloading corrections.

**4. Text Preprocessing:**

- Input text is preprocessed to handle spelling corrections before grammar checking. The application processes each word to correct spelling, then assembles the corrected words into a coherent sentence for grammar analysis.

**5. Definitions and Suggestions:**

- Definitions of corrected words are provided to offer additional context and understanding. Grammar issues are highlighted with suggestions for improvement, enhancing the user's ability to refine their writing.



# **CONCLUSION**

This project successfully implements a spelling and grammar check application using Python. The application provides accurate spelling corrections, grammatical error identification, and word definitions, enhancing text quality. Future improvements could include expanding the dictionary API integration and optimizing performance for larger texts.

The Spelling and Grammar Check Application demonstrates a robust solution for enhancing written communication by leveraging advanced text-processing tools:

- **Effectiveness:** The integration of the autocorrect library and LanguageTool provides comprehensive spelling and grammar checking. The application successfully corrects spelling errors and offers detailed grammar corrections, improving text quality.
- **User Experience:** The Streamlit-based web interface ensures a user-friendly experience. Users can interact with the application seamlessly, check their text, and receive instant feedback on corrections.
- **Utility:** By combining spelling correction with grammar checking and definition fetching, the application serves as a valuable tool for anyone looking to improve their writing skills. It supports users in creating error-free, grammatically accurate text while also providing educational value through definitions.
- **Future Enhancements:** Future improvements could include:
  - Expanding language support for grammar checking.
  - Enhancing the dictionary API integration for more comprehensive word definitions.
  - Adding additional text analysis features, such as style and tone checking.

Overall, the application meets its objective of providing an effective spelling and grammar checking solution, contributing to better writing and communication.

## **REFERENCES**

1. **Speller Documentation:** <https://pypi.org/project/autocorrect/>
2. **LanguageTool Documentation:** <https://languagetool.org/>
3. **Dictionary API Documentation:** <https://dictionaryapi.dev/>
4. **Python Programming Language:**
  1. Python Software Foundation. (n.d.). Python. Retrieved from <https://www.python.org/>
5. **Autocorrect Library:**
  1. autocorrect. (n.d.). GitHub. Retrieved from <https://github.com/fsondej/autocorrect>
6. **LanguageTool:**
  1. LanguageTool. (n.d.). LanguageTool - Online grammar and style checking. Retrieved from <https://languagetool.org/>
7. **Requests Library:**
  1. Kenneth Reitz. (2017). Requests: HTTP for Humans. Retrieved from <https://docs.python-requests.org/>
8. **DictionaryAPI:**
  1. DictionaryAPI. (n.d.). Dictionary API. Retrieved from <https://dictionaryapi.dev/>
9. **Streamlit:**
  1. Streamlit. (n.d.). Streamlit: The fastest way to build data apps. Retrieved from <https://streamlit.io/>
10. **NLTK (Natural Language Toolkit):**
  1. Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python. O'Reilly Media. Retrieved from <https://www.nltk.org/>
11. **NLTK WordNet:**
  1. WordNet. (n.d.). WordNet. Retrieved from <https://wordnet.princeton.edu/>

\*\*\*\*\*