**PRO WRITING COMPANION**

**USING FLASK**

**A PROJECT REPORT**

***Submitted by***

|  |  |  |
| --- | --- | --- |
| **SARAVANA PON BALAN F** | **-** | **953120104044** |
| **LAKSHMANAN S** | **-** | **953120104019** |
| **MAHESH RAJA V** | **-** | **953120104022** |

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

***in***

**COMPUTER SCIENCE AND ENGINEERING**

**THAMIRABHARANI ENGINEERING COLLEGE, TIRUNELVELI**

**ANNA UNIVERSITY::CHENNAI 600 025**

**MAY 2022**

### ANNA UNIVERSITY::CHENNAI 600 025

### BONAFIDE CERTIFICATE

Certified that this project report **"PRO WRITING COMPANION"** is the bonafide work of **"SARAVANA PON BALAN F(953120104044)**, **LAKSHMANAN S(953120104019) AND MAHESH RAJA V(953120104022)"** who carried out the project work under my supervision.

###### SIGNATURE

Dr. A. ANNA LAKSHMI M. E., Ph. D.,

###### HEAD OF THE DEPARTMENT

Department of Computer Science and Engineering

Thamirabharani Engineering College Tirunelveli - 627 358

###### SIGNATURE

Mr. D. G. DAVID M. E.,

###### SUPERVISOR

Assistant Professor

Department of Computer Science and Engineering

Thamirabharani Engineering College Tirunelveli - 627 358

**Submitted to Anna University Viva -Voce held on \_\_\_\_\_\_\_\_\_\_\_**

**INTERNAL EXAMINER EXTERNAL EXAMINER**

### ACKNOWLEDGEMENT

We wish our heartful thanks to Mr. M. R. Paulraj, Chairman, Mr. S. Perumal, Vice Chairman, Mr. S. Murugesapandian, Treasurer and Mr. Senthil Kumar Palraj, Secretary for the blessing and constant support over our project period.

We wish to express our thanks to Dr. D. Ravindran, M. E., Ph. D., Principal, Thamirabharani Engineering College.

We wish to express our indebtedness to our Head of the Department Dr. A. Anna Lakshmi M. E., Ph. D., for her valuable guidance, advice and encouragement in successfully completing our work.

Our special thanks to our Project Coordinator and our Project guide Mr. R. Jeeva M. E., Assistant Professor, Department of Computer Science and Engineering for his valuable guidance and support to complete our project successfully.

We also express our sincere thanks to entire Faculty and Lab Assistants in Computer Science and Engineering Department for their co-operation and support.

We are very grateful to our well-wishers as without their encouragement we would not able to make this project work as success.

#### ABSTRACT

Pro Writing Companion is a powerful writing tool that enhances your skills with its user-friendly Flask-based application. This comprehensive software incorporates essential features such as grammar and spelling checkers, along with a text summarization function. Whether you're a student, professional, or simply someone who wants to improve their writing, this application is designed to elevate your writing abilities. The grammar checker helps identify and correct grammatical errors, while the spelling checker ensures error-free text. Additionally, the summarization feature condenses lengthy texts into concise summaries, saving you time and effort. With Pro Writing Companion, take your writing to new heights effortlessly and efficiently.

#### TABLE OF CONTENT

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **TITLE**  **ABSTRACT** | **PAGE NO**  **IV** |
|  | **LIST OF FIGURES** | **VII** |
|  | **LIST OF ABBREVIATIONS** | **VIII** |
| **1** | **INTRODUCTION** | **1** |
|  | 1.1 PROJECT OVERVIEW | 1 |
|  | 1.2 OBJECTIVES | 1 |
| **2** | **BACKGROUND AND RELATED WORK** | **3** |
|  | 2.1 WRITING SKILLS AND CHALLENGES | 3 |
|  | 2.2 EXISTING WRITING TOOLS | 3 |
|  | 2.3 FLASK FRAMEWORK | 5 |
| **3** | **SYSTEM DESIGN AND ARCHITECTURE** | **6** |
|  | 3.1 APPLICATION WORKFLOW | 7 |
|  | 3.2 DESIGNING THE HOME PAGE | 8 |
|  | 3.3 IMPLEMENTING THE GRAMMAR CHECKER | 9 |
|  | 3.4 INTEGRATING THE SPELLING CHECKER | 10 |
|  | 3.5 CREATING THE SUMMARIZATION PAGE | 11 |
| **4** | **IMPLEMENTATION AND DEVELOPMENT** | **12** |
|  | 4.1 SETTING UP THE FLASK ENVIRONMENT | 12 |
|  | 4.2 IMPLEMENTING THE HOME PAGE | 14 |
|  | 4.3 IMPLEMENTING THE GRAMMAR CHECKER | 16 |
|  | 4.4 IMPLEMENTING THE SPELLING CHECKER | 17 |
|  | 4.5 IMPLEMENTING THE SUMMARIZATION PAGE | 18 |
| **5** | **TESTING AND EVALUATION** | 19 |
|  | 5.1 TEST CASES AND SCENARIOS | 19 |
| **6** | **SYSTEM REQUIREMENTS** | 22 |
|  | 6.1 HARDWARE REQUIREMENTS | 22 |
|  | 6.2 SOFTWARE REQUIREMENTS | 22 |
| **7** | **SYSTEM DESIGN** | 23 |
|  | 7.1 UML DIAGRAMS | 23 |
|  | 7.1.1 USE CASE DIAGRAM | 23 |
|  | 7.1.2 CLASS DIAGRAM | 24 |
| **8** | **CONCLUSION AND FUTURE WORK** | 25 |
|  | 8.1 SUMMARY OF ACHIEVEMENTS | 25 |
|  | 8.2 LIMITATIONS AND FUTURE IMPROVEMENTS | 26 |
| **9** | **REFERENCES** | 28 |
|  | APPENDIX A - SAMPLE CODE | 28 |
|  | APPENDIX B – OUTPUT SCREENSHOTS | 33 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **TITLE** | **PAGE NO.** |
| 1.1 | Application Workflow | 2 |
| 2.1 | Home Page Design | 19 |
| 2.2 | Grammar Checkers Integration | 25 |
| 2.3 | Spelling Checker Integration | 28 |
| 2.4 | Summarization Page Design | 28 |
| 3.1 | Example of Grammar Check Result | 29 |
| 3.2 | Example of Spelling Check Result | 39 |
| 3.3. | Example of Summarization Result | 43 |
| 4.1 | Performance Evaluation Results | 46 |

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| CPU | Central Processing Unit |
| OS | Operating System |
| UML | Unified Modeling Language |
| NLTK | Natural Language Toolkit |
| OOP | Object Oriented Programming |
| PIP | pip Install Packages |
| HTML | Hypertext Markup Language |
| CSS | Cascading Style Sheets |
| JS | JavaScript |
| UI | User Interface |
| NLP | Natural Language Processing |
| DBMS | Database Management System |

**1.INTRODUCTION**

Pro Writing Companion is an innovative application designed to enhance your writing skills. With our user-friendly interface and advanced features, including a grammar checker, spelling checker, and summarization page, we aim to elevate your writing to new heights. Say goodbye to errors and hello to polished, professional writing.

* 1. **Project Overview:**

The Pro Writing Companion is an application developed to empower users in improving their writing skills. With a user-friendly interface and a range of powerful features, including grammar checking, spelling correction, and summarization, our application aims to assist users in producing high-quality written content. By eliminating errors and providing valuable insights, the Pro Writing Companion enhances the clarity, coherence, and overall effectiveness of users' writing. Whether for professional, academic, or personal purposes, this application serves as a valuable tool for individuals looking to elevate their writing skills and achieve excellence in their written communication.

**1.2 Project Objectives:**

The objective of your project, the Pro Writing Companion, can be to provide users with a comprehensive tool to enhance their writing skills and improve the quality of their written content. The application aims to achieve the following objectives:

1. Grammar Checking: Offer a reliable grammar checking feature that identifies grammatical errors in the user's text and provides suggestions for corrections. This helps users enhance the clarity, coherence, and correctness of their writing.
2. Spelling Checking: Provide a robust spelling checking functionality that detects and suggests corrections for misspelled words. This helps users eliminate spelling mistakes and improve the overall accuracy of their writing.
3. Summarization: Implement a summarization page that generates concise summaries of longer texts or articles. This feature enables users to extract key information and main points quickly, saving them time and effort.
4. User-friendly Interface: Create an intuitive and user-friendly interface that allows users to easily navigate the application, access different features, and interact with their written content. The UI should be visually appealing and responsive, enhancing the overall user experience.
5. Backend Integration: Utilize Flask and IBM DB2 to handle the backend functionalities, such as storing user data, managing requests, and providing a seamless experience for users.
6. Accuracy and Performance: Strive for high accuracy in grammar and spelling checks, ensuring that the application provides reliable suggestions and corrections. Additionally, optimize the performance of the application to ensure quick response times and efficient usage.
7. Accessibility: Aim to make the Pro Writing Companion accessible to a wide range of users by ensuring compatibility with different devices, browsers, and screen sizes. Consider implementing accessibility features such as keyboard navigation and support for assistive technologies.

**2. BACKGROUND AND RELATED WORK**

**2.1 Writing Skills And Challenges**

Writing skills are essential in various aspects of life, including academics, professional settings, and personal communication. Effective writing involves conveying ideas, thoughts, and information in a clear, coherent, and engaging manner. However, many individuals face challenges when it comes to writing, which can hinder their ability to communicate effectively through written text. Some common challenges include:

1. Grammar and Syntax: Incorrect grammar and syntax can impact the clarity and understanding of written content. Challenges in this area include subject-verb agreement, sentence structure, punctuation, and proper word usage.
2. Spelling and Vocabulary: Misspelled words and limited vocabulary can undermine the credibility and professionalism of written work. Difficulties may arise from homophones, unfamiliar words, and inconsistent spelling.
3. Organization and Coherence: Arranging ideas in a logical and coherent manner is crucial for effective communication. Challenges may include developing a clear thesis statement, organizing paragraphs, and maintaining a coherent flow of information.
4. Conciseness and Clarity: Writing concisely while maintaining clarity can be challenging. Struggling to express ideas succinctly or using convoluted sentence structures can make writing difficult to understand.
5. Summarization: Extracting key information from longer texts can be time-consuming and challenging. Summarization skills are valuable for condensing information into shorter, cohesive summaries.

**2.2 Existing Writing Tools**

There are several existing writing tools and applications available that aim to assist individuals in improving their writing skills and overcoming common challenges. These tools offer various features and functionalities to enhance grammar, spelling, organization, and overall writing quality. Here are some notable examples:

1. Grammarly: Grammarly is a popular online writing assistant that provides real-time grammar checking, spell checking, and style suggestions. It offers browser extensions and integrates with various writing platforms to provide comprehensive writing support.
2. Hemingway Editor: The Hemingway Editor helps writers improve the readability and clarity of their writing. It highlights complex sentences, suggests simpler alternatives, and provides readability scores to make text more accessible and engaging.
3. ProWritingAid: ProWritingAid is an all-in-one writing tool that offers grammar checking, style editing, and writing analysis. It provides detailed reports on issues like grammar, spelling, style, and overused words, helping writers refine their work.
4. Microsoft Word: Microsoft Word, a widely used word processing software, includes built-in spelling and grammar checking features. It highlights potential errors and offers suggestions for corrections, helping users identify and fix mistakes in their writing.
5. Google Docs: Google Docs is a collaborative online document editor that includes basic spelling and grammar checking features. It allows multiple users to work on a document simultaneously and provides suggestions for corrections as you type.
6. Thesaurus.com: Thesaurus.com is an online resource that offers synonyms, antonyms, and related words to help writers enhance their vocabulary and avoid repetition. It assists in finding the most appropriate words for conveying ideas effectively.
7. Scrivener: Scrivener is a powerful writing tool designed specifically for long-form writing projects. It provides features for organizing research, outlining, and structuring content, making it a popular choice among authors and academics.
8. Evernote: Evernote is a note-taking application that enables users to capture ideas, make notes, and organize information. It allows writers to gather and structure their thoughts, making it easier to develop and refine their writing.

These existing writing tools offer valuable support in different areas of writing, from grammar and spelling to organization and overall writing quality. The Pro Writing Companion project can build upon these existing tools while providing a comprehensive solution that combines multiple features, such as grammar checking, spelling checking, and summarization, in a user-friendly interface.

**2.3 Flask Framework**

The Flask framework is a popular web framework for building web applications using Python. It is known for its simplicity, flexibility, and ease of use. Flask follows the model-view-controller (MVC) architectural pattern and provides the necessary tools and libraries to develop web applications efficiently. Here are some key aspects and features of Flask:

1. Lightweight and Minimalistic: Flask is designed to be lightweight and minimalistic, providing only the essential components for web development. It does not impose any specific project structure, allowing developers to have more freedom and flexibility in organizing their code.
2. Routing: Flask uses a routing system to map URLs to specific functions or views. Developers can define routes and associate them with corresponding functions, which are executed when a particular URL is accessed. This allows for handling different requests and building the application's endpoints.
3. Templating Engine: Flask includes a templating engine, commonly Jinja2, that enables the separation of HTML and dynamic content. Templates allow developers to create reusable HTML layouts and inject dynamic data into them, improving code readability and maintainability.
4. Request and Response Handling: Flask provides request and response handling mechanisms, allowing developers to access incoming request data and construct appropriate responses. It supports handling different HTTP methods like GET, POST, PUT, and DELETE, and facilitates working with form data, query parameters, and JSON payloads.
5. Flask Extensions: Flask has a rich ecosystem of extensions that provide additional functionalities and integrations. These extensions cover a wide range of areas, including database integration, authentication, form handling, and more. Flask extensions simplify the development process by providing ready-made solutions for common tasks.
6. Flask-WTF: Flask-WTF is an extension that integrates Flask with WTForms, a popular form handling library in Python. It simplifies form creation, validation, and handling, allowing developers to build robust and secure forms within their Flask applications.
7. Flask-SQLAlchemy: Flask-SQLAlchemy is an extension that integrates Flask with SQLAlchemy, a powerful Object-Relational Mapping (ORM) library. It provides a convenient interface for interacting with databases, abstracting away the complexities of raw SQL queries.
8. Testing Support: Flask provides support for testing web applications through its testing framework. It allows developers to write unit tests and integration tests to ensure the correctness and reliability of their applications.

Flask's simplicity, coupled with its extensive documentation and active community, makes it a popular choice for web development using Python. It offers a flexible and scalable foundation for building web applications, including the backend of your Pro Writing Companion project.

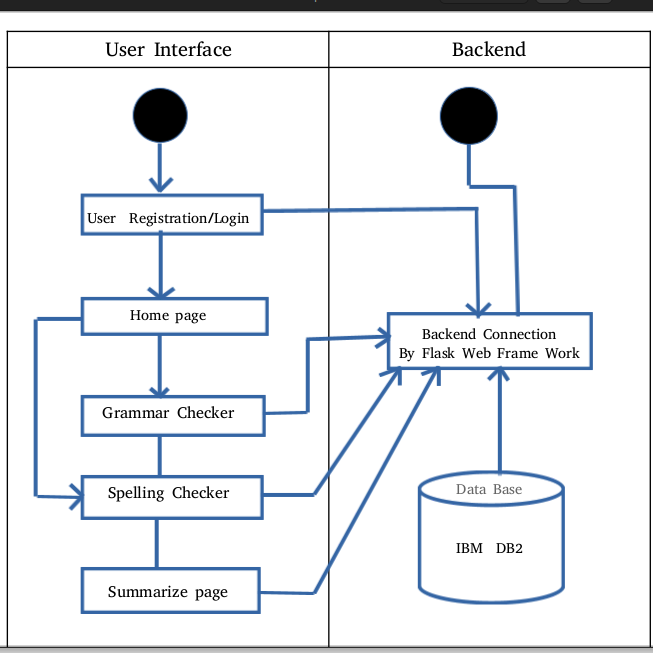
**3. SYSTEM DESIGN AND ARCHITECTURE**

The system design and architecture of your Pro Writing Companion application play a crucial role in ensuring its functionality, scalability, and maintainability. Here's an outline of the system design and architecture considerations for your project:

1 Overall Architecture

2 Client-Side (UI)

3 Server-Side (Backend)



**3.1 Application Workflow**

Homepage: The application workflow begins with the homepage, which serves as the landing page for users. It provides an overview of the features available in the Pro Writing Companion application, along with options to log in or register.

1. User Registration: If users are new to the application, they can choose to register by providing their necessary details, such as username, email address, and password. You can implement a registration form with validation to ensure data integrity. Once registered, users' credentials can be securely stored in the IBM DB2 database.
2. User Login: For returning users, the login functionality allows them to access their personalized accounts. Users can enter their registered username/email and password to authenticate their identity. Implement appropriate authentication mechanisms, such as hashing and salting passwords, to ensure secure login processes.
3. Authenticated User: Once users successfully log in, they gain access to additional functionalities and personalized features. This can include the ability to save their writing history, preferences, and settings.
4. Grammar Checker, Spelling Checker, and Summarization: Users can utilize the grammar checker, spelling checker, and summarization features by inputting their text into the respective sections. The entered text is sent to the server-side backend, where the implemented libraries (Gingerit, TextBlob, Sumy) analyze the text and provide appropriate suggestions, corrections, or summaries.
5. Saving User Data: To enhance the user experience, you can provide an option for users to save their writing history or favorite summaries. This allows users to review and access their previous work, making it easier for them to track progress or refer back to important content. The saved data can be associated with the user's account and stored in the database.
6. Logout: Users should have the ability to log out from their accounts, terminating their session and ensuring the security of their data. Implement a logout mechanism that clears user session data and redirects them to the homepage or a logged-out state.

**3.2 Designing the Home Page**

The home page of your Pro Writing Companion application serves as the initial point of interaction for users. It should provide an informative and visually appealing interface that showcases the features and benefits of your application. Here are some design considerations for creating an effective home page:

1. Clear and Concise Messaging: Ensure that the purpose and value proposition of your application are communicated clearly on the home page. Use concise and compelling messaging to explain how the Pro Writing Companion can elevate users' writing skills and improve their overall writing experience.
2. Engaging Visuals: Incorporate visually appealing elements, such as high-quality images or graphics, to make the home page visually engaging. Visuals can help capture users' attention and create a positive first impression of your application. Use images that are relevant to writing, editing, or learning to enhance the overall aesthetic appeal.
3. Feature Highlights: Highlight the key features and functionalities of your application on the home page. This can be done through concise descriptions or bullet points that outline the benefits users can expect from using the Pro Writing Companion. Use visually distinct sections or cards to showcase each feature, making it easy for users to understand and navigate.
4. Intuitive Navigation: Design a user-friendly navigation system that allows users to access different sections of your application easily. Include a clear and prominent menu or navigation bar that directs users to other pages, such as the grammar checker, spelling checker, summarization page, and user account settings. Ensure that the navigation is intuitive and accessible on different devices.

**3.3 Designing the Grammar Checker**

The grammar checker feature of your Pro Writing Companion application aims to assist users in improving the grammar and syntax of their writing. Designing an effective and user-friendly grammar checker involves considering the following aspects:

1. Input Interface: Create an input interface where users can enter or paste their text for grammar checking. Provide a text editor or textarea where users can type or paste their content. Consider including additional formatting options, such as font size, font style, and text alignment, to enhance the editing experience.
2. Grammar Checking Algorithm: Integrate the Gingerit and TextBlob libraries into your backend to perform the grammar checking. When users submit their text for checking, the application should send the text to the server-side backend, which processes the text using these libraries to identify grammar errors and provide suggestions for corrections.
3. Error Highlighting: Implement a mechanism to highlight the grammar errors in the user's text. This can be done by underlining or highlighting the erroneous words or phrases. Use distinct colors or styling to make the errors stand out, making it easier for users to identify and correct them.
4. Suggested Corrections: Display the suggested corrections for grammar errors alongside the highlighted errors. Provide users with a list or dropdown menu of alternative suggestions or corrections for each identified error. Users can then select the appropriate correction and apply it to their text.
5. User-friendly Interface: Design a clean and intuitive interface for the grammar checker. Make sure the interface is easy to understand and navigate, with clear instructions on how to use the feature. Use visually distinct sections or panels to separate the original text, highlighted errors, and suggested corrections.
6. Real-time Checking: Consider implementing real-time grammar checking as users type their content. Provide instant feedback by highlighting errors and offering suggestions in real-time, giving users the opportunity to correct mistakes as they write.

**3.4 Designing the Spelling Checker**

The spelling checker feature of your Pro Writing Companion application helps users identify and correct spelling errors in their written content. Designing an effective and user-friendly spelling checker involves considering the following aspects:

1. Input Interface: Create an input interface where users can enter or paste their text for spelling checking. Provide a text editor or textarea where users can type or paste their content. Consider including additional formatting options, such as font size, font style, and text alignment, to enhance the editing experience.
2. Spelling Checking Algorithm: Integrate the TextBlob library into your backend to perform the spelling checking. When users submit their text for checking, the application should send the text to the server-side backend, which processes the text using the TextBlob library to identify spelling errors and suggest corrections.
3. Error Highlighting: Implement a mechanism to highlight the spelling errors in the user's text. This can be done by underlining or highlighting the misspelled words. Use distinct colors or styling to make the errors stand out, making it easier for users to identify and correct them.
4. Suggested Corrections: Display the suggested corrections for spelling errors alongside the highlighted errors. Provide users with a list or dropdown menu of alternative suggestions or corrections for each identified misspelling. Users can then select the appropriate correction and apply it to their text.
5. User-friendly Interface: Design a clean and intuitive interface for the spelling checker. Make sure the interface is easy to understand and navigate, with clear instructions on how to use the feature. Use visually distinct sections or panels to separate the original text, highlighted errors, and suggested corrections.
6. Real-time Checking: Consider implementing real-time spelling checking as users type their content. Provide instant feedback by highlighting errors and offering suggestions in real-time, giving users the opportunity to correct mistakes as they write.

**3.5 Designing the Summarization Page**

The summarization page of your Pro Writing Companion application allows users to generate concise summaries of their written content. Designing an effective and user-friendly summarization page involves considering the following aspects:

1. Input Interface: Create an input interface where users can enter or paste the text they want to summarize. Provide a text editor or textarea where users can type or paste their content. Consider including additional formatting options, such as font size, font style, and text alignment, to enhance the editing experience.
2. Summarization Algorithm: Integrate the Sumy library into your backend to perform the text summarization. When users submit their text for summarization, the application should send the text to the server-side backend, which processes the text using the Sumy library to generate concise summaries.
3. Summary Display: Display the generated summary to the users in a visually distinct and easily readable format. Consider using a separate section or panel to present the summary text. Ensure that the font size and spacing are appropriate to make the summary easily scannable and comprehensible.
4. Adjustable Summary Length: Provide users with the option to adjust the length of the generated summary. Allow them to select the desired length, such as the number of sentences or percentage of the original text, to customize the summary according to their needs.
5. Additional Summary Details: Include additional information about the summary, such as the word count, readability score, or key concepts extracted from the original text. This can provide users with more insights into the summarized content and help them evaluate its relevance and quality.
6. User-friendly Interface: Design a clean and intuitive interface for the summarization page. Ensure that the interface is easy to understand and navigate, with clear instructions on how to use the feature. Use visually distinct sections or panels to separate the original text, summary display, and any additional summary details.

**4. IMPLEMENTATION AND DEVELOPMENT**

**4.1 Setting Up the Flask Environment**

Install Python: Make sure Python is installed on your system. Flask requires Python 3.7 or higher. You can download Python from the official Python website ([https://www.python.org](https://www.python.org/)) and follow the installation instructions for your operating system.

* Create a Virtual Environment (Optional): It is recommended to create a virtual environment to isolate your project's dependencies. Open a terminal or command prompt and navigate to your project directory. Run the following command to create a virtual environment:

***bash: python3 -m venv env***

Activate the virtual environment:

On Windows:

***bash: env\Scripts\activate***

On macOS and Linux:

***bash :source env/bin/activate***

* Install Flask: With the virtual environment activated, you can install Flask using the following command:

***bash : pip install flask***

This will install Flask and its dependencies.

* Create Flask App: Create a new Python file, such as app.py, in your project directory. This file will serve as the entry point for your Flask application. Open app.py and import the necessary Flask modules:

***python***

***from flask import Flask, render\_template, request***

* Initialize Flask App: Create an instance of the Flask app in app.py:

***python***

***app = Flask(\_\_name\_\_)***

* Define Routes: Define the routes for your application by adding route decorators to functions in app.py. For example, you can define a route for the home page:

***python***

***@app.route('/')***

***def home():***

***return render\_template('index.html')***

This example assumes that you have an index.html file in a templates folder.

* Run the Application: In your terminal or command prompt, navigate to your project directory and run the following command to start the Flask development server:

***bash:export FLASK\_APP=app.py***

***bash:flask run***

Flask will start the development server, and you can access your application by visiting the provided URL, usually ***http://127.0.0.1:5000/***.

**4.2 Implementing The Home Page**

To implement the home page of your Pro Writing Companion application, you need to create the necessary HTML, CSS, and Flask code. Here's a step-by-step guide to help you get started:

1. HTML Structure: Create an HTML file, such as index.html, in a templates folder. This file will contain the structure and content of your home page. Begin by adding the HTML boilerplate code and create a <body> tag.
2. Design and Layout: Design the layout and structure of your home page using HTML and CSS. Consider the following elements:
   * Header: Create a header section that includes the application logo or title and any additional navigation elements.
   * Hero Section: Design a visually appealing hero section that showcases the key features and benefits of your Pro Writing Companion application. Use headings, subheadings, and descriptive text to communicate the value proposition.
   * Call-to-Action Buttons: Add prominent call-to-action buttons that allow users to register or log in to the application. These buttons should stand out and be strategically placed for maximum visibility.
   * Feature Sections: Create sections or cards that highlight the main features of your application, such as the grammar checker, spelling checker, and summarization page. Include brief descriptions and visually appealing graphics or icons to represent each feature.
   * Footer: Include a footer section that provides additional information, such as contact details, links to privacy policy/terms of service, and social media links.
3. Styling with CSS: Create a CSS file, such as styles.css, to style your home page. Link the CSS file to your HTML file by adding a <link> tag in the <head> section of index.html. Use CSS to apply styles, such as colors, fonts, spacing, and positioning, to various elements on the home page. Ensure consistency and visual coherence throughout the page.
4. Flask Integration: In your Flask app.py file, define a route for the home page ('/'). Inside the route function, use Flask's render\_template() function to render the index.html file:

***python***

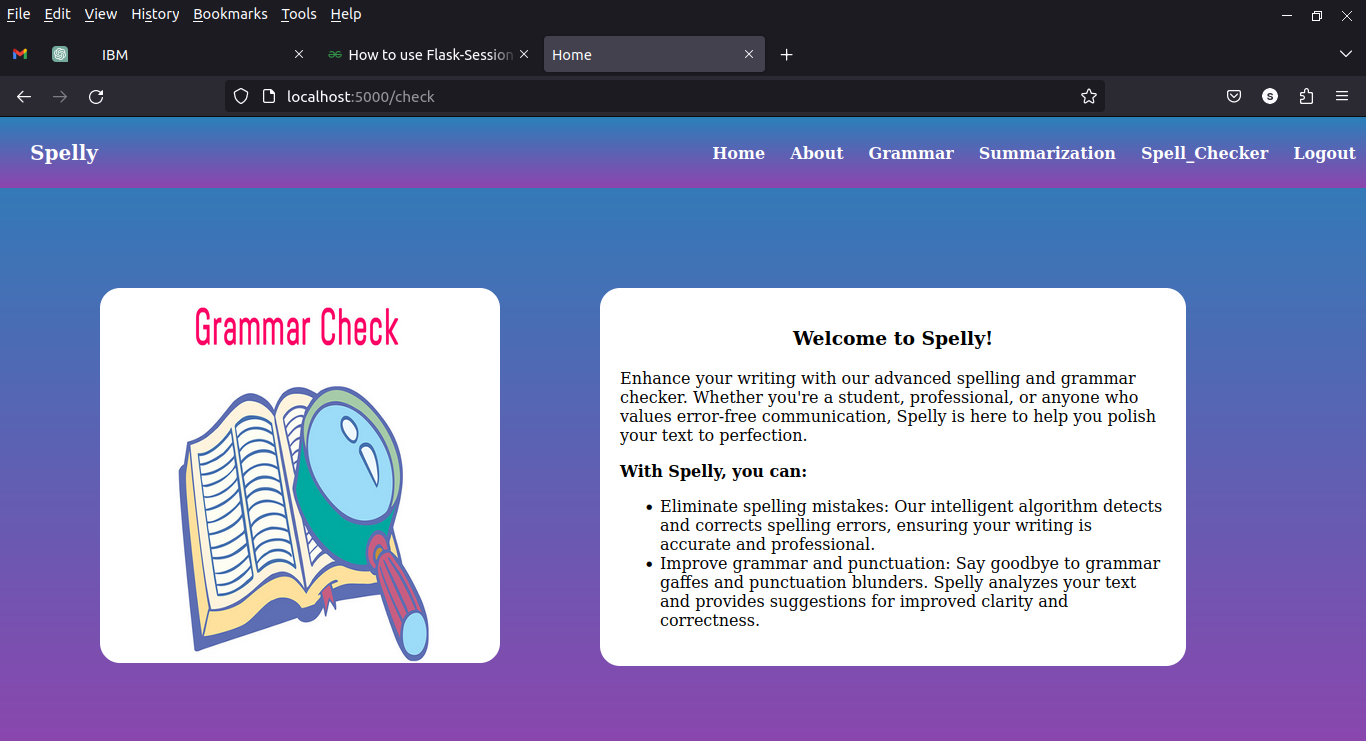
***@app.route('/')***

***def home():***

***return render\_template('index.html')***

* Test the Home Page: Start the Flask development server by running flask run in your terminal or command prompt. Visit the provided URL (usually http://127.0.0.1:5000/) in your web browser to test and view the home page.
* Iterate and Refine: Review the appearance and functionality of your home page, and make any necessary adjustments to the HTML and CSS code. Continue refining the design until you are satisfied with the visual and interactive aspects of the home page.

**Below image is model for home page:**



**4.3 Implementing the Grammar Checker**

Create a new Python file: Create a new Python file, such as grammar\_checker.py, in your project directory. This file will contain the code for the grammar checking functionality.

* Import necessary libraries: In the grammar\_checker.py file, import the required libraries for grammar checking. You mentioned earlier that you are using the Gingerit and TextBlob libraries for grammar checking. Include the following import statements at the top of your file:

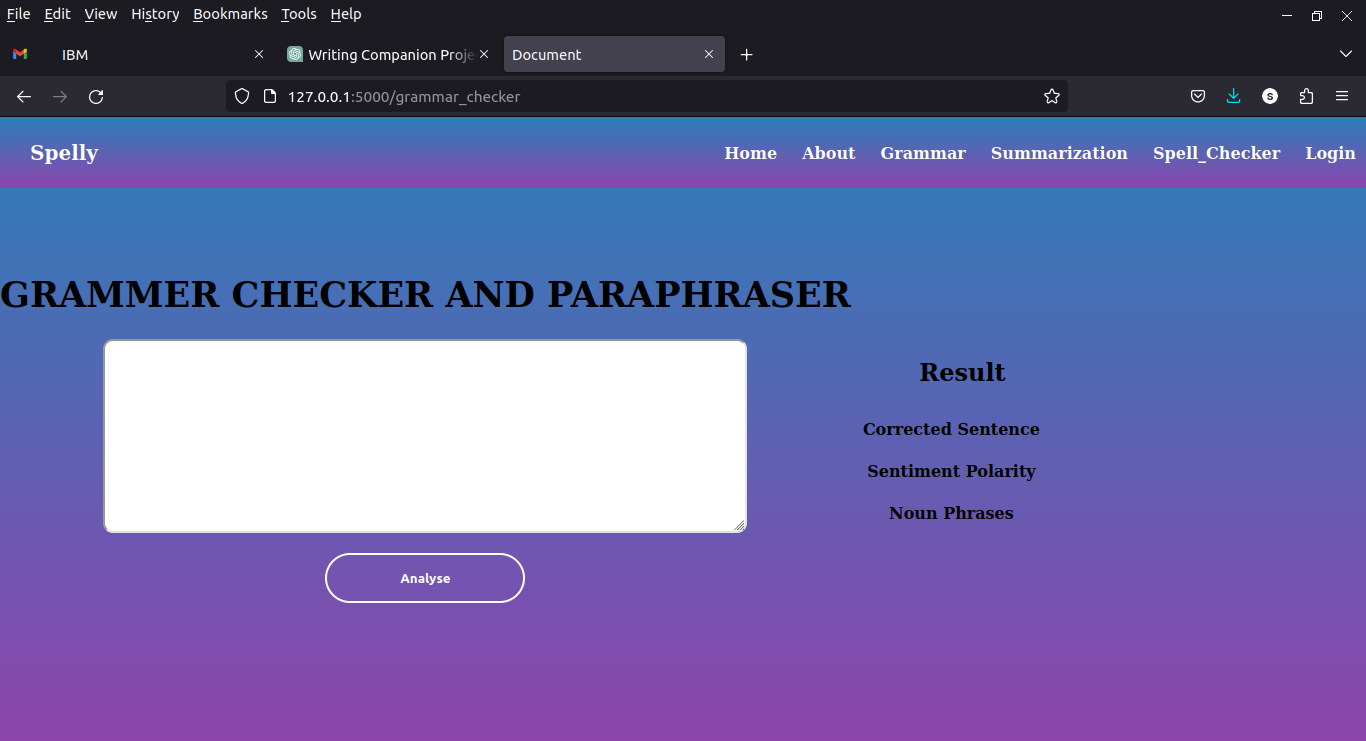
***python***

***from gingerit.gingerit import GingerIt***

***from textblob import TextBlob***

* Define the grammar checking function: Create a function, such as check\_grammar, that takes the input text as a parameter and performs the grammar checking. Within this function, you can use the Gingerit and TextBlob libraries to check and correct grammar errors
* Finally return the responses(corrected Sentence,Sentiment Polarity,Noun Phrases).

**Below image is model for grammar checker page:**



**4.4 Integrating the Spelling Checker**

To integrate the spelling checker functionality into your Pro Writing Companion application, you can create a separate Python file that handles the spelling checking process. Here's a step-by-step guide:

1. Create a new Python file: Create a new Python file, such as spelling\_checker.py, in your project directory. This file will contain the code for the spelling checking functionality.
2. Import necessary libraries: In the spelling\_checker.py file, import the required libraries for spelling checking. You mentioned earlier that you are using the TextBlob library for spelling checking. Include the following import statement at the top of your file:

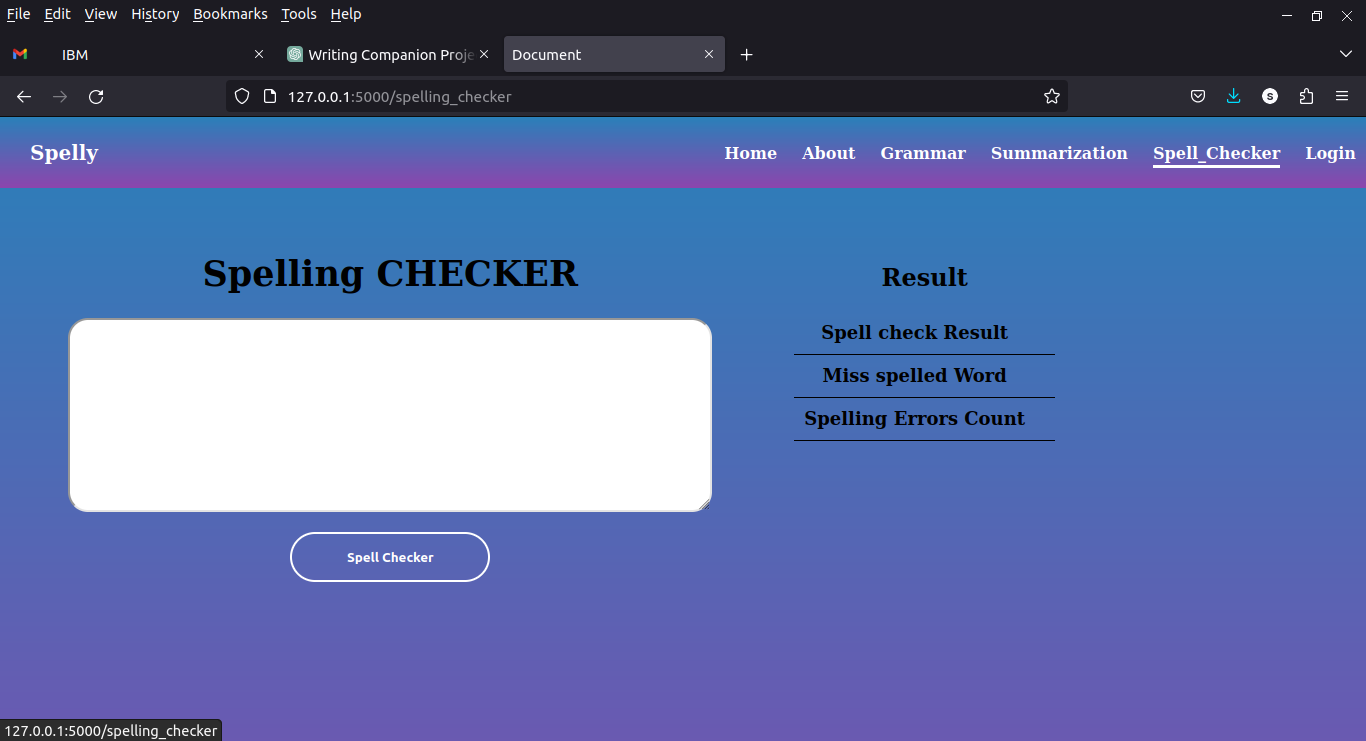
***python***

***from textblob import TextBlob***

3. Define the spelling checking function: Create a function, such as check\_spelling, that takes the input text as a parameter and performs the spelling checking. Within this function, you can use the TextBlob library to check and correct spelling errors.

4. Finally return the response to UI

**Below image is model for Spelling checker page:**

****

**4.5 Creating the Summarization Page**

Create a new Python file: Create a new Python file, such as summarization.py, in your project directory. This file will contain the code for the text summarization functionality.

* Import necessary libraries: In the summarization.py file, import the required libraries for text summarization. You mentioned earlier that you are using the Sumy library for text summarization. Include the following import statements at the top of your file:

***python***

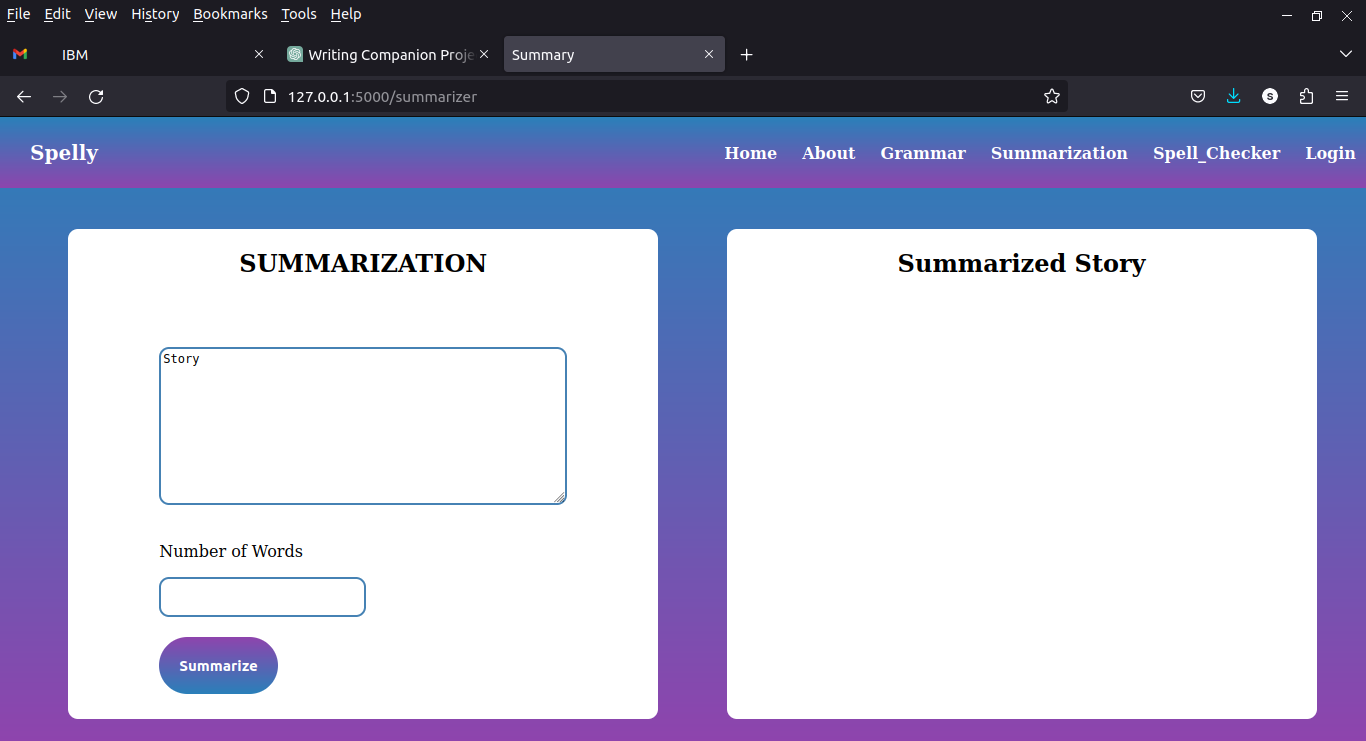
***from sumy.parsers.plaintext import PlaintextParser***

***from sumy.nlp.tokenizers import Tokenizer***

***from sumy.summarizers.lex\_rank import LexRankSummarizer***

* Define the summarization function: Create a function, such as summarize\_text, that takes the input text and the number of sentences as parameters and performs text summarization. Within this function, you can use the Sumy library to summarize the text
* Finally return the response to UI

**Below image is model for summarizer page:**

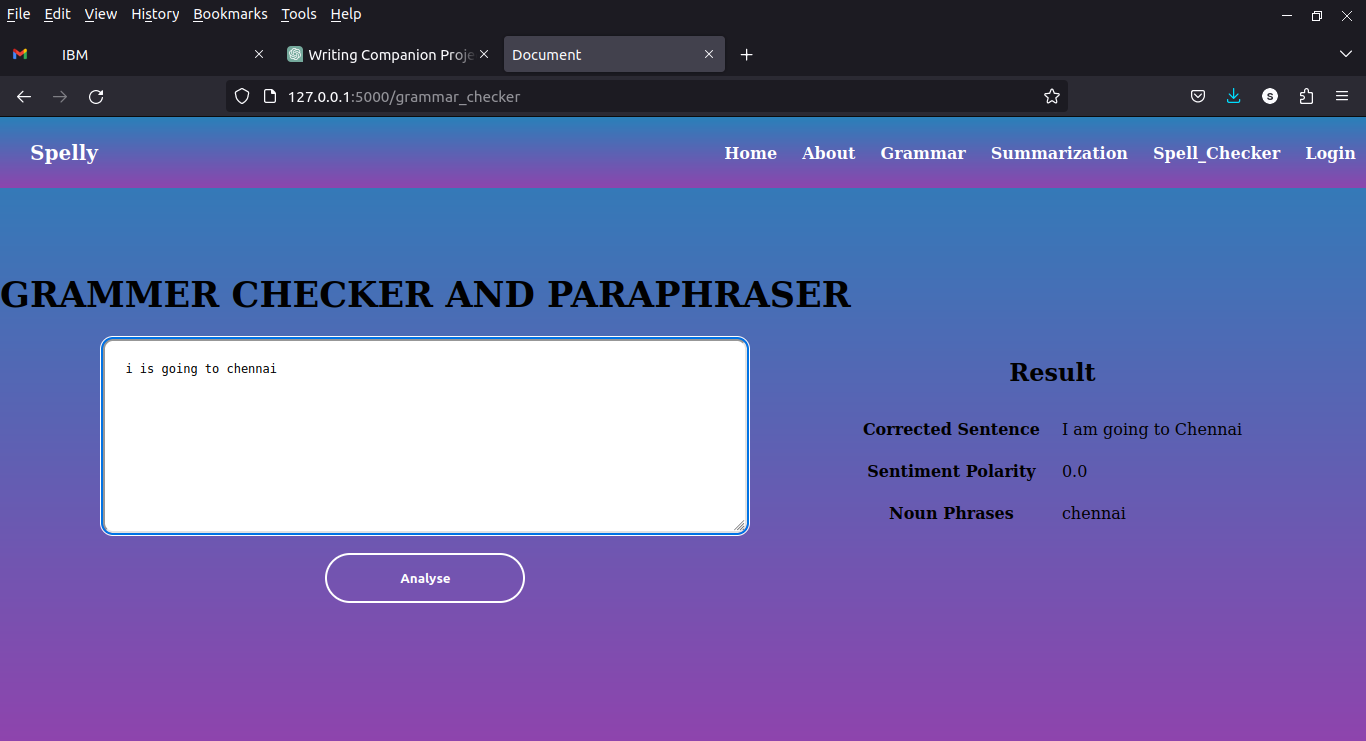
****

**5. TESTING AND EVALUATION**

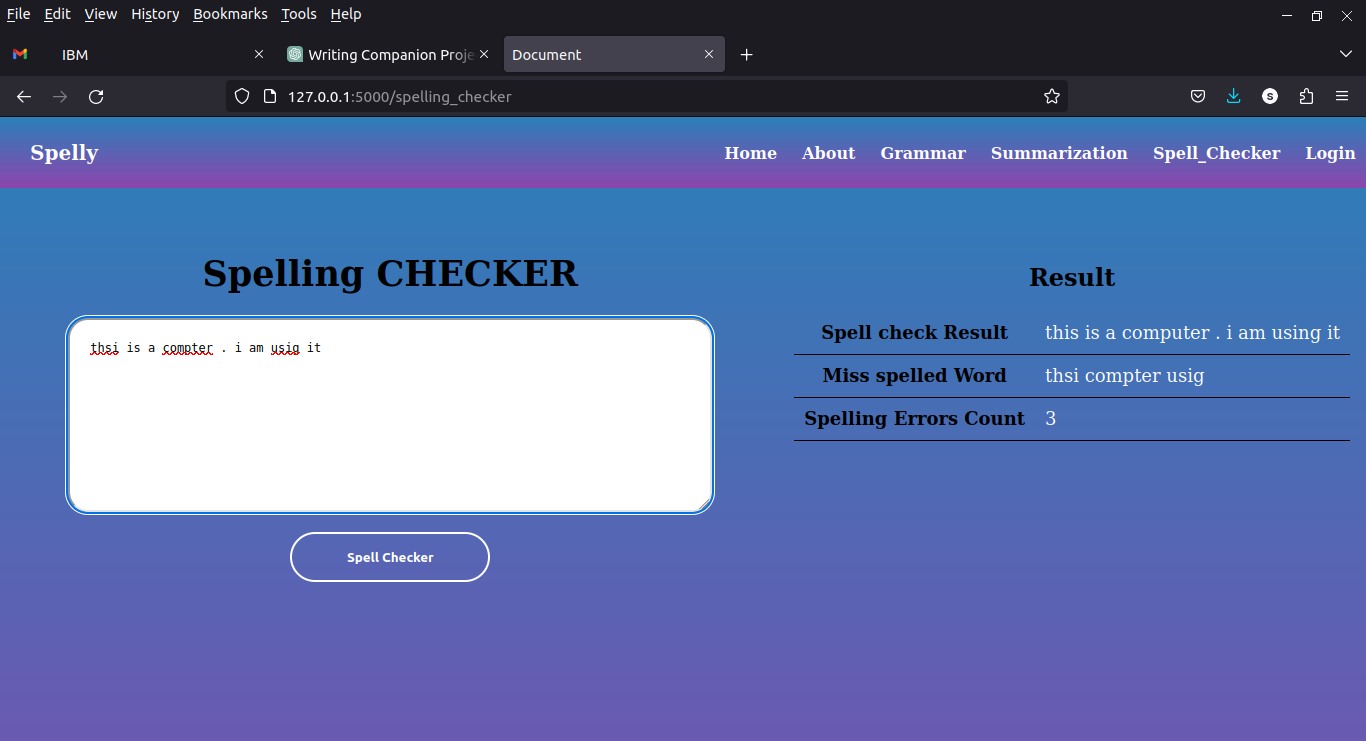
**5.1 TEST CASES AND SCENARIOS**

Here are some test cases and scenarios you can consider for testing our application:

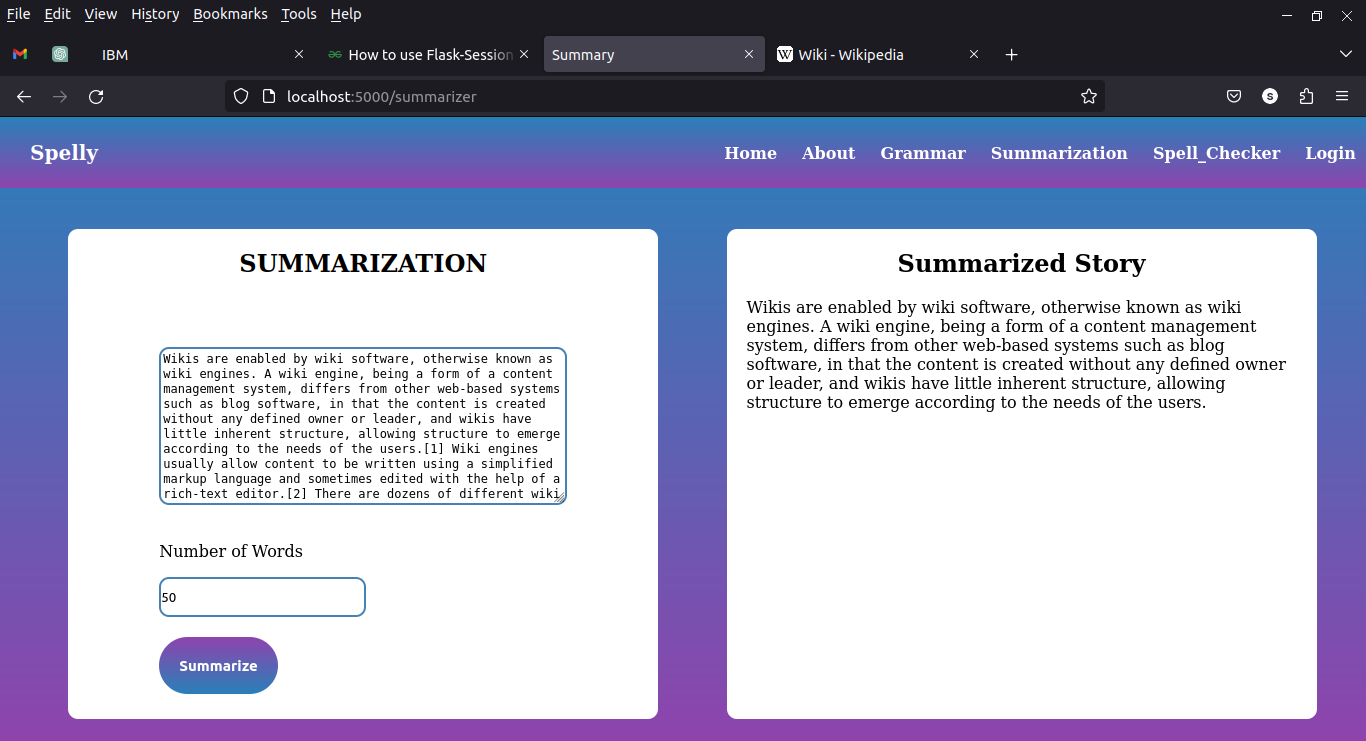
1. **Grammar Checker:**
   * Test Case 1: Input a paragraph with multiple grammar errors. Verify that the grammar checker correctly identifies and suggests corrections for each error.
   * Test Case 2: Input a grammatically correct paragraph. Verify that the grammar checker does not suggest any corrections.
   * Test Case 3: Input an empty text or invalid input. Verify that the grammar checker provides an appropriate error message or handles the input gracefully.
   * Sample output of grammar checker:



1. **Spelling Checker:**
   * Test Case 1: Input a paragraph with multiple spelling errors. Verify that the spelling checker correctly identifies and suggests corrections for each error.
   * Test Case 2: Input a paragraph with no spelling errors. Verify that the spelling checker does not suggest any corrections.
   * Test Case 3: Input an empty text or invalid input. Verify that the spelling checker provides an appropriate error message or handles the input gracefully.
   * Sample output of spelling checker:



1. **Summarization:**
   * Test Case 1: Input a long text or article. Verify that the summarization algorithm generates a concise and accurate summary based on the specified number of sentences.
   * Test Case 2: Input a short text or sentence. Verify that the summarization algorithm handles shorter inputs appropriately, either by returning the input as the summary or by providing a meaningful summary within the given context.
   * Test Case 3: Input an empty text or invalid input. Verify that the summarization algorithm provides an appropriate error message or handles the input gracefully.
   * Sample output of Summarization of text:



1. **User Interface:**
   * Test Case 1: Verify that the home page is visually appealing and displays all the relevant information and features of the application.
   * Test Case 2: Test the responsiveness of the application by accessing it on different devices and screen sizes. Verify that the UI elements are displayed properly and the application functions well across different platforms.
   * Test Case 3: Perform user interaction tests, such as clicking buttons, entering text, and submitting forms, to ensure smooth and intuitive user experience.
2. **Performance:**
   * Test Case 1: Test the application's response time by inputting a large text and measuring the time taken for the grammar checking, spelling checking, and summarization processes.
   * Test Case 2: Test the application's stability and resource usage by simulating multiple concurrent users accessing different features simultaneously.
   * Test Case 3: Monitor the application for memory leaks, excessive CPU usage, or other performance issues during extended usage periods.
3. **Error Handling:**
   * Test Case 1: Input text that exceeds the maximum character limit. Verify that the application handles and displays an appropriate error message.
   * Test Case 2: Test the application's error handling capabilities by deliberately providing invalid input or triggering unexpected scenarios. Verify that the application gracefully handles such situations without crashing or compromising data integrity.

**6 . SYSTEM REQUIREMENTS**

**6.1 Hardware Requirements**

**Processor :** Intel Core i5 or AMD Ryzen 5

**Memory(RAM) :** 4GB RAM

**Storage :** 0.2GB of disk space

**Network Connectivity :** yes

**6.2 Software Requirements**

**Operating System**  **:** Linux ,Windows 10,11

**Python :** version 3.x

**Integrated Development Environment (IDE):**Visual Studio Code or Sublime Text.

**Flask :** version 2.3.x

**IBM DB2 :** IBM Cloud Account

**Libraries and Dependencies :** install via requirements.txt

**Web Browser :** Google Chrome, Mozilla Firefox

**Web Server :** WSGI,Apache

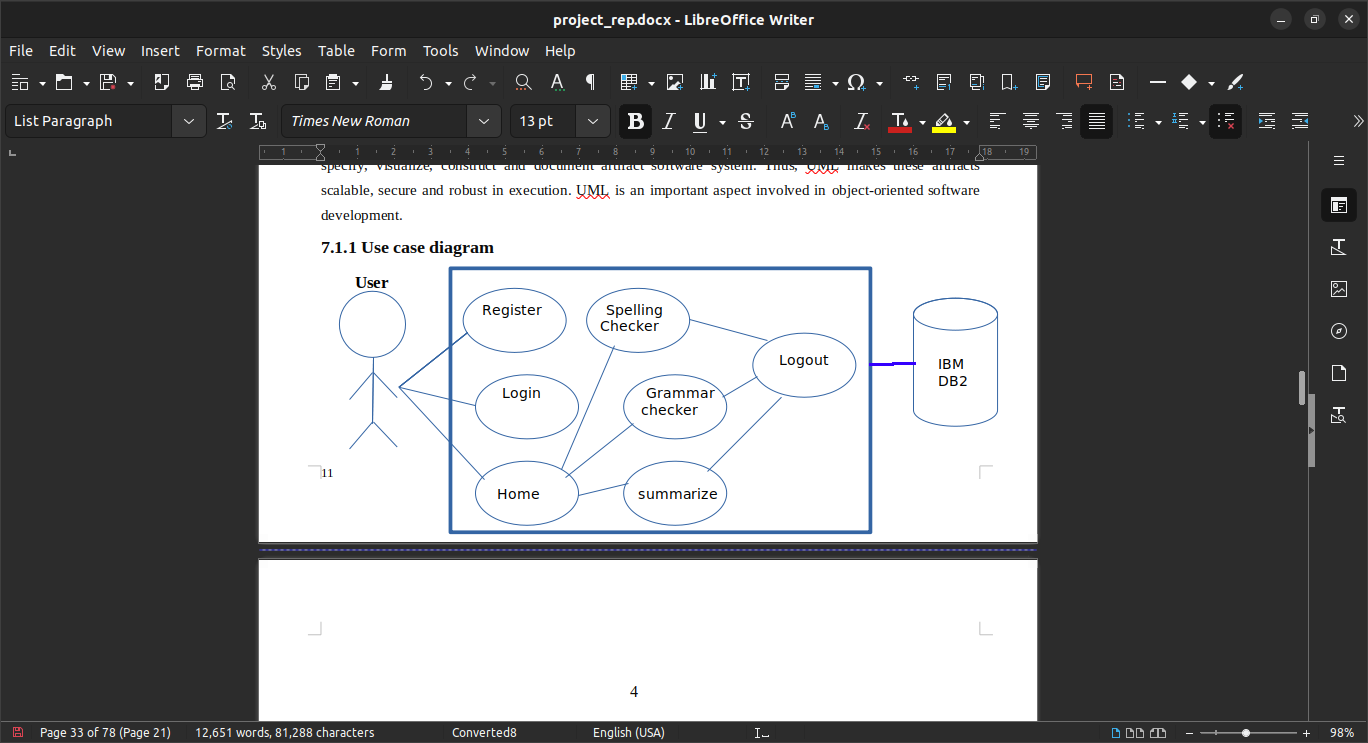
**7. SYSTEM DESIGN**

**7.1 Uml Diagrams**

Unified Modeling language (UML) is a standardized modeling language enabling developers to specify, visualize, construct and document artifact software system. Thus, UML makes these artifacts scalable, secure and robust in execution. UML is an important aspect involved inobject-oriented software development.

**7.1.1 Use Case Diagram**

Use case diagrams give a graphic overview of the actors involved in a system, different functions needed by those actors and how these different functions interact. UML use case diagrams are ideal for representing the goals of system-user interactions, defining and organizing functional requirements in a system, Specifying the context and requirements of a system, modeling the basic flow of events in a use case



**7.1.2 Class Diagram**

Class diagrams are the main building block of any object-oriented solution. It shows the classes in a system, attributes, and operations of each class and the relationship between each class. In most modeling tools, a class has three parts. Name at the top, attributes in the middle and operations or methods at the bottom. In a large system with many related classes, classes are grouped together to create class diagrams. Different relationships between classes are shown by different types of arrows.

**DataBase**

**User ID**

**Password**

**Auth()**

**User**

**User ID**

**Password**

**Register**

**Email**

**Password**

**createAccount()**

**Application**

**Grammar checker**

**Spelling Checker**

**Summarization**

**Grammarcheck()**

**SpellCheck()**

**Summarize()**

1 \*

\*

1 \* 1

1

1

1 1

**8.CONCLUSION AND FUTURE WORK**

**8.1 Summary Of Achievements**

In this project, we have developed the Pro Writing Companion application, aimed at enhancing users' writing skills. The application comprises a home page, grammar checker, spelling checker, and summarization page. It leverages various technologies, including HTML, CSS, JavaScript, Flask, IBM DB2, gingerit, TextBlob, and Sumy, to provide a comprehensive writing assistance tool.

Throughout the project, we have achieved the following milestones:

1. Designed and implemented the application's user interface, including the home page, where users can access different features.
2. Integrated the grammar checker, powered by the gingerit and TextBlob libraries. Users can input text and receive suggestions to improve grammar and sentence structure.
3. Incorporated the spelling checker, which utilizes the TextBlob library to identify and suggest corrections for spelling errors in the user's text.
4. Developed the summarization page using the Sumy library, allowing users to generate concise summaries of their input text based on a specified number of sentences.
5. Established the Flask framework as the backbone of the application, enabling seamless communication between the user interface and the backend functionalities.
6. Utilized IBM DB2 as the backend database, providing storage and retrieval capabilities for user accounts and related data.

These achievements mark significant progress in creating a powerful writing companion tool. However, there are opportunities for future work to further enhance the application:

1. Enhancing Accuracy: Continuously improve the grammar and spelling checking algorithms by incorporating more advanced natural language processing techniques and leveraging larger language models.
2. Expanding Language Support: Extend the grammar and spelling checking capabilities to support multiple languages, catering to a broader user base.
3. User Feedback and Collaboration: Incorporate feedback mechanisms to gather user suggestions and feedback, enabling iterative improvements to the application.
4. Advanced Summarization Techniques: Explore and integrate other text summarization algorithms and techniques to enhance the quality and coherence of generated summaries.
5. Performance Optimization: Optimize the application's performance by fine-tuning the backend processes and utilizing caching mechanisms for faster response times.
6. Mobile Application Development: Consider developing a mobile version of the Pro Writing Companion application to provide users with writing assistance on the go.

By addressing these areas of improvement and considering user feedback, the Pro Writing Companion application can evolve into a powerful tool that helps users improve their writing skills, providing a seamless and intuitive writing experience.

Overall, the project has demonstrated the successful implementation of key features and laid the foundation for further improvements and expansion. With continued development and enhancement, the Pro Writing Companion has the potential to become a valuable resource for writers, students, and professionals seeking to improve their written communication.

**8.2 Limitations And Future Improvements**

While the Pro Writing Companion application has achieved significant milestones, it is important to acknowledge its limitations and outline potential areas for future improvements. Here are some limitations and suggestions for future enhancements:

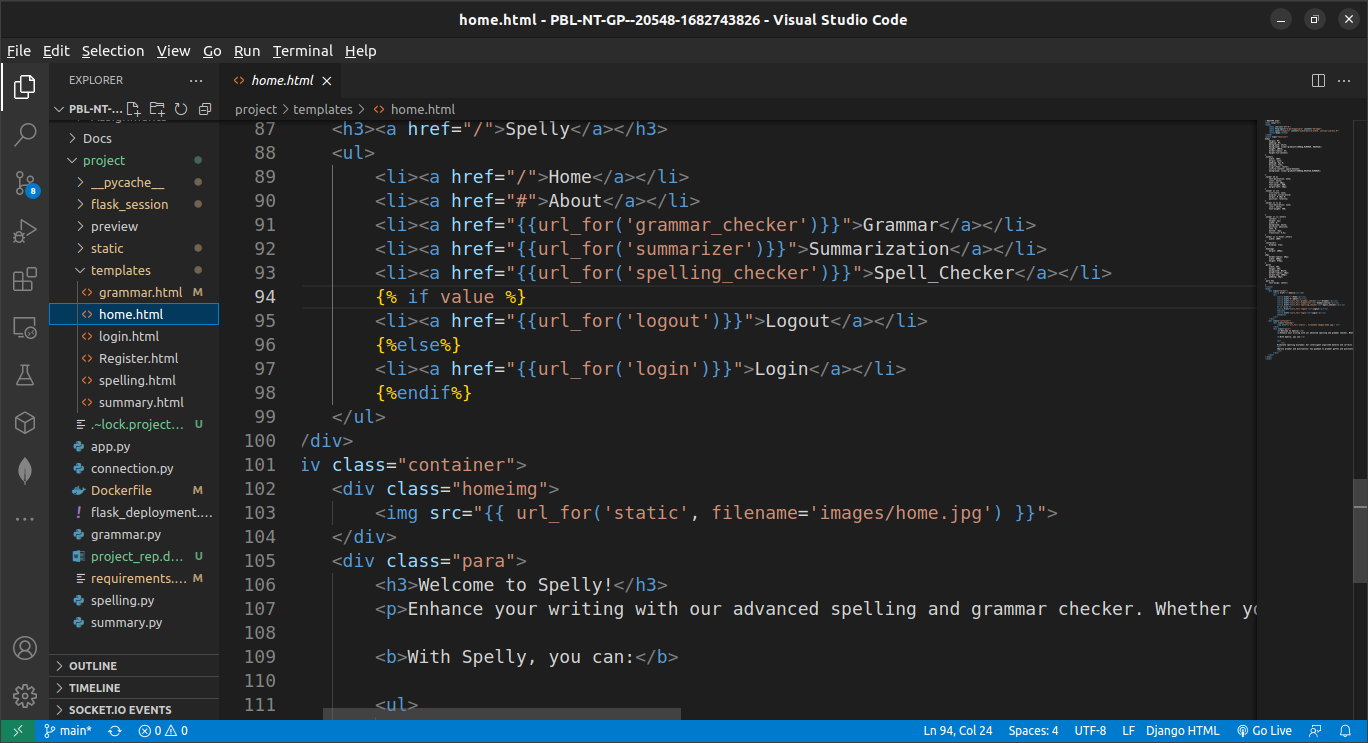
1. Accuracy of Grammar and Spelling Checking: The current implementation relies on the gingerit and TextBlob libraries for grammar and spelling checking. While they provide useful suggestions, they may not capture all grammatical or spelling errors accurately. Future work can focus on integrating more advanced and comprehensive language processing tools or models to enhance the accuracy of these checks.
2. Language Support: The application currently supports a specific language or a limited set of languages. To cater to a wider user base, future improvements could include adding support for additional languages, enabling users to check grammar and spelling in their preferred language.
3. Summarization Quality: The current implementation uses the Sumy library for text summarization. While it provides basic summarization functionality, the quality and coherence of the generated summaries can be further improved. Future work can explore advanced summarization techniques, such as neural network-based models or transformer architectures, to produce more accurate and contextually relevant summaries.
4. User Interface Enhancements: While the current user interface provides the necessary functionality, future improvements could focus on enhancing the user experience and visual appeal. Consider incorporating modern design principles, responsive layouts for different screen sizes, and intuitive navigation to create a more engaging and user-friendly interface.
5. Performance Optimization: As the application grows and handles larger volumes of user input, performance optimization becomes crucial. Future work can focus on optimizing the backend processes, database queries, and caching mechanisms to ensure faster response times and a smoother user experience.
6. Error Handling and User Feedback: Enhancements can be made to the error handling mechanism by providing more informative error messages and suggestions to users when errors occur. Additionally, incorporating a user feedback mechanism can gather valuable insights for improving the application and addressing user needs.
7. Integration with Third-Party APIs: Consider integrating with external APIs, such as language translation services or plagiarism detection tools, to enhance the functionality and usefulness of the application.
8. Mobile Application Development: In the era of mobile devices, developing a mobile version of the Pro Writing Companion application can extend its reach and convenience, allowing users to access writing assistance anytime, anywhere.
9. Continuous Updates and Maintenance: Keep the application and its underlying libraries up to date with the latest versions to benefit from bug fixes, security patches, and new features. Regular maintenance and updates ensure a stable and secure application.

By addressing these limitations and incorporating future improvements, the Pro Writing Companion application can evolve into a more robust and comprehensive tool, providing users with enhanced writing assistance and a better overall user experience.

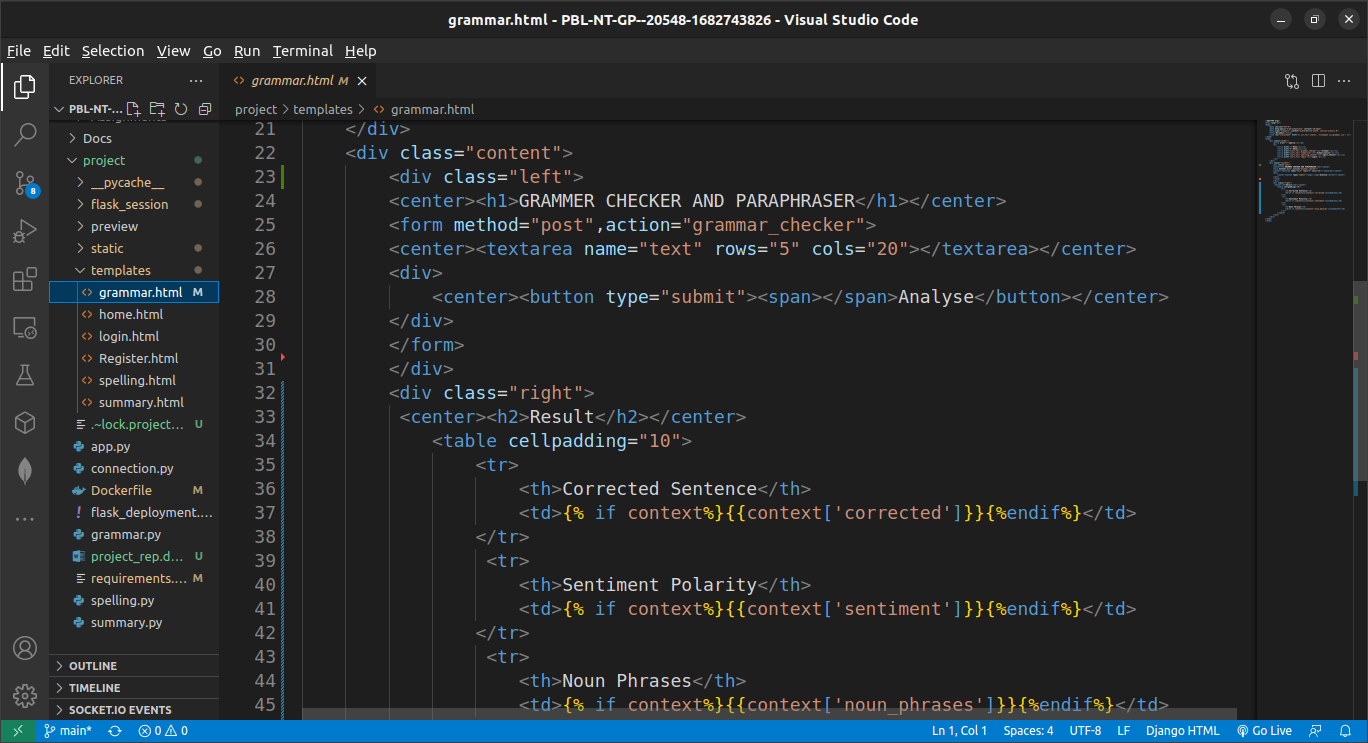
**9. REFERENCES**

**APPENDIX A - SAMPLE CODE**

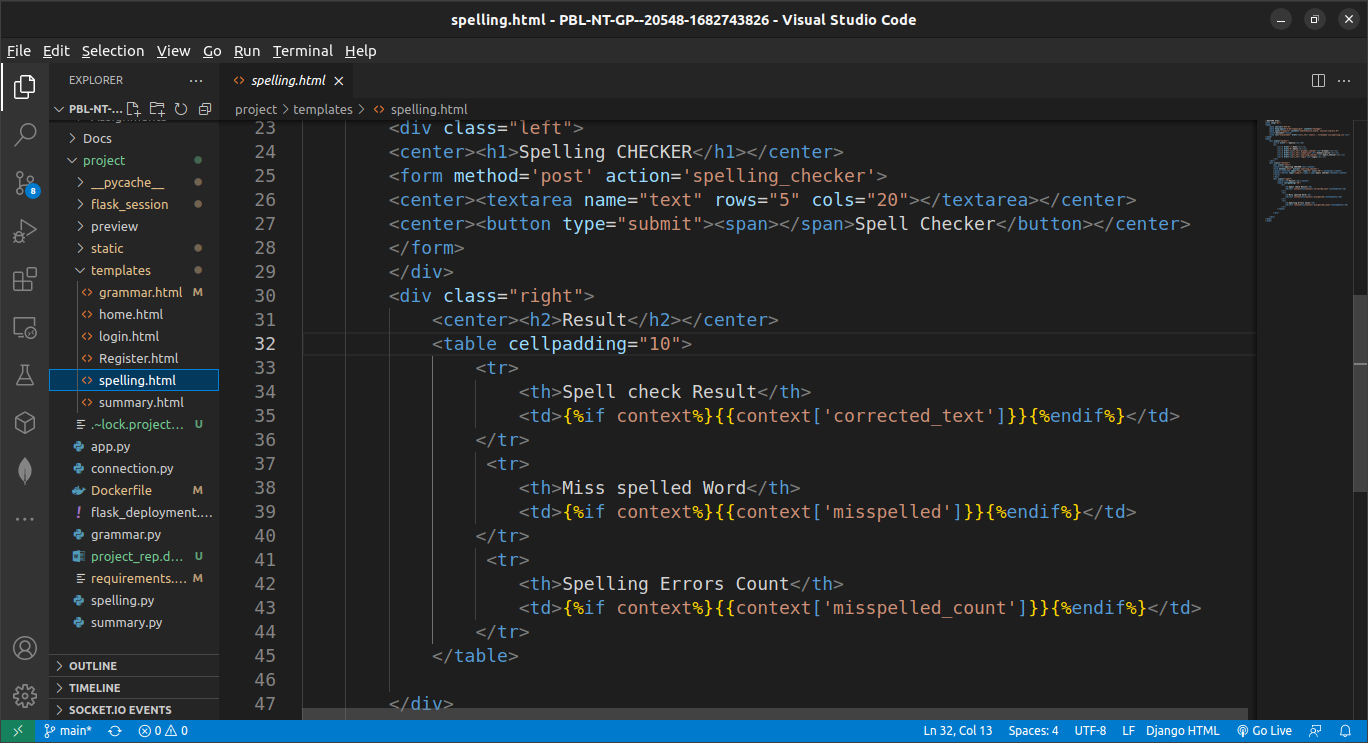
**sample Home.html**

****

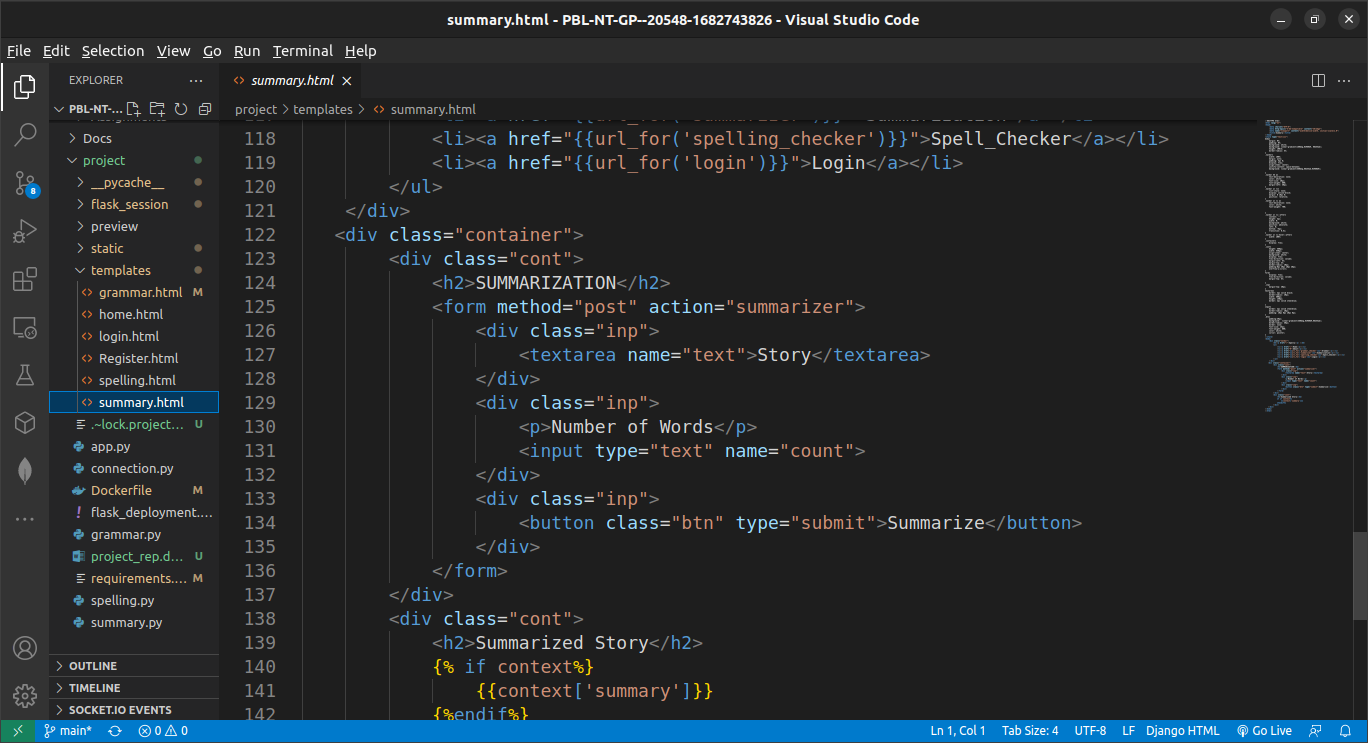
**sample grammar Checker.html**

****

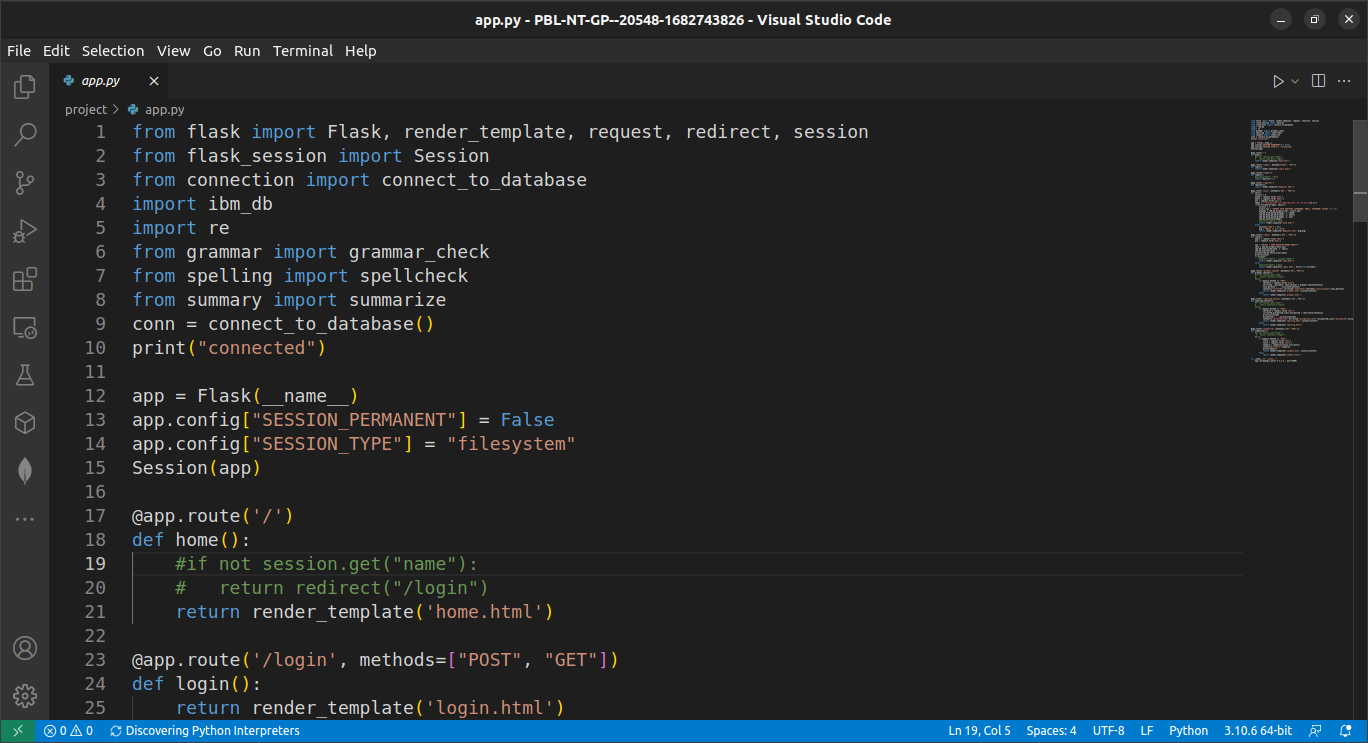
**sample spelling.html**

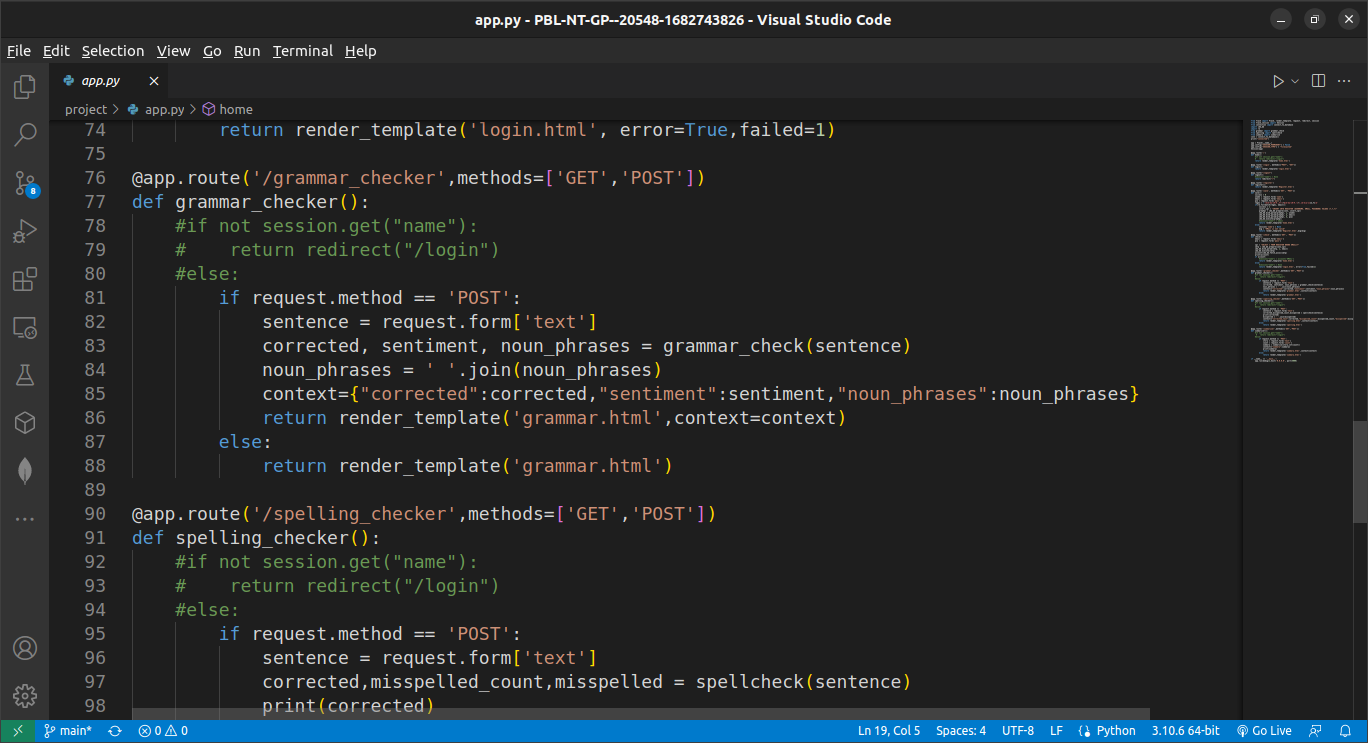
****

**sample summarizer.html**

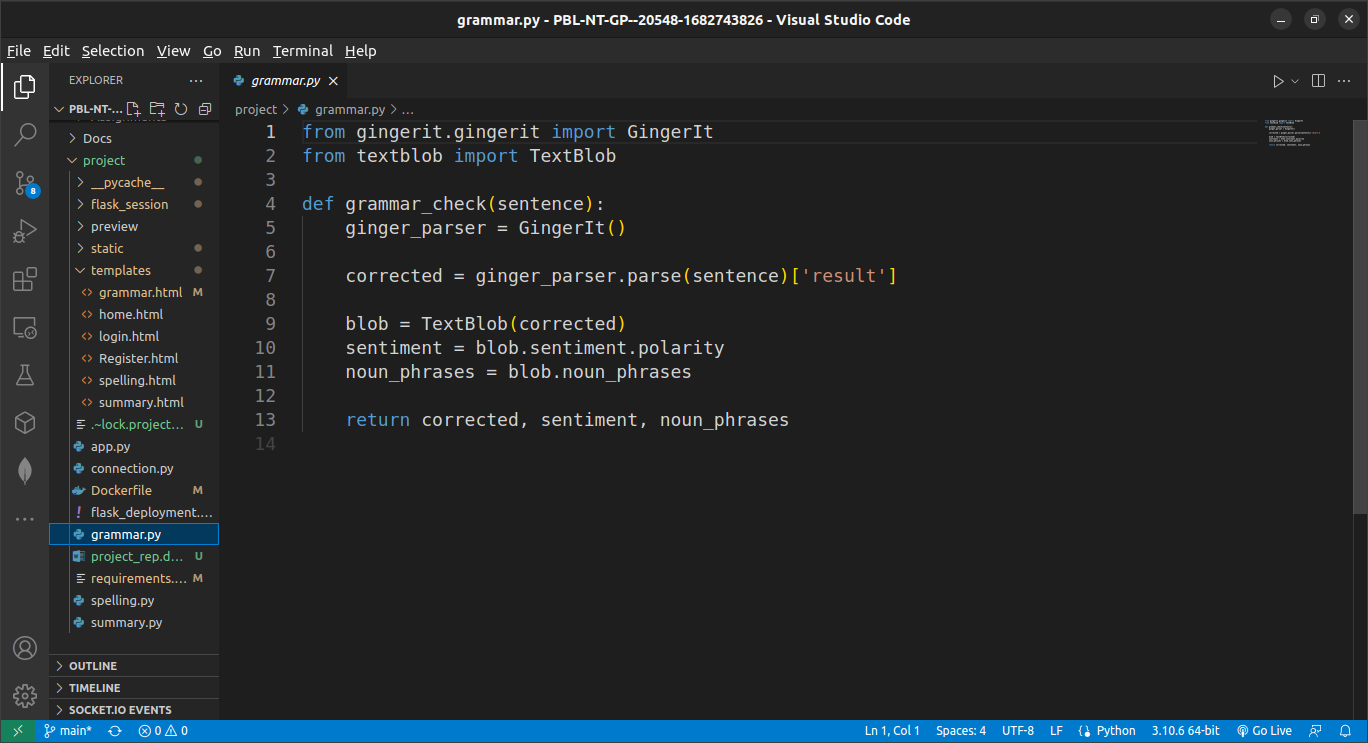
****

**sample app.py**

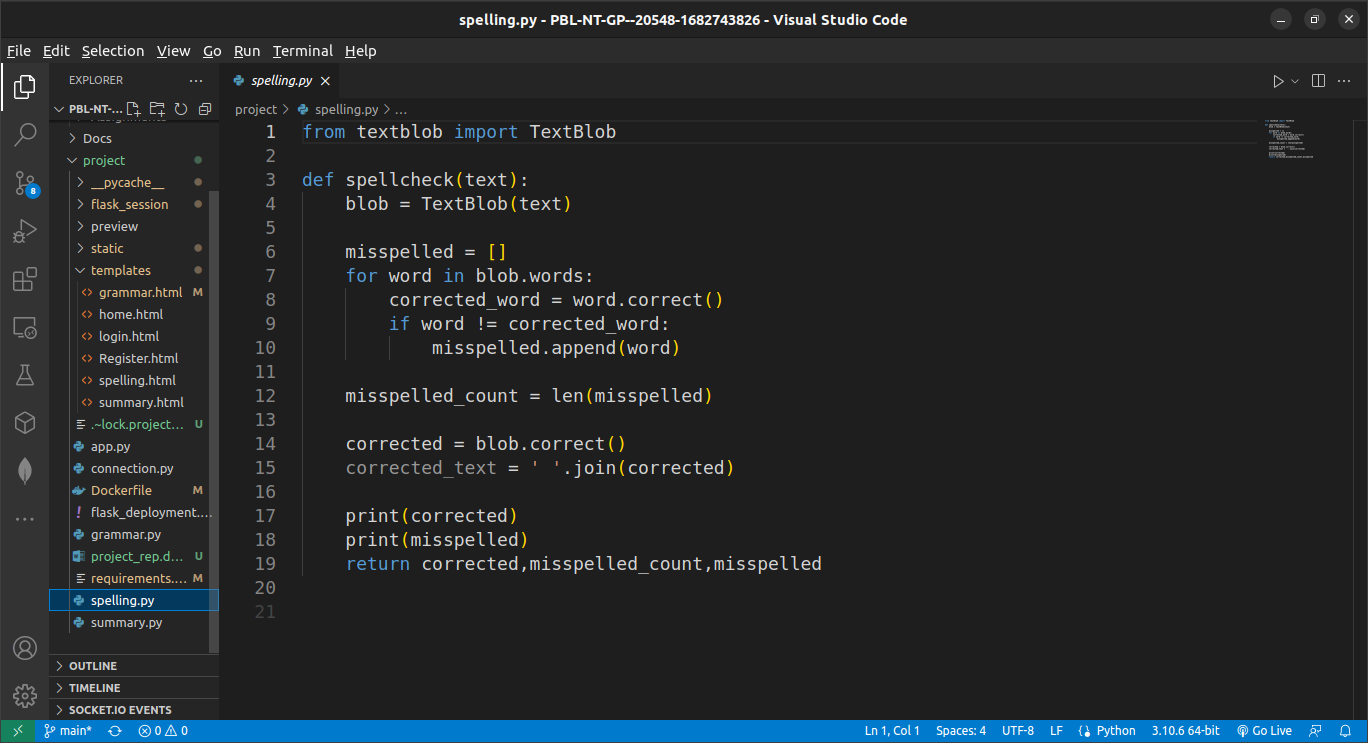
****

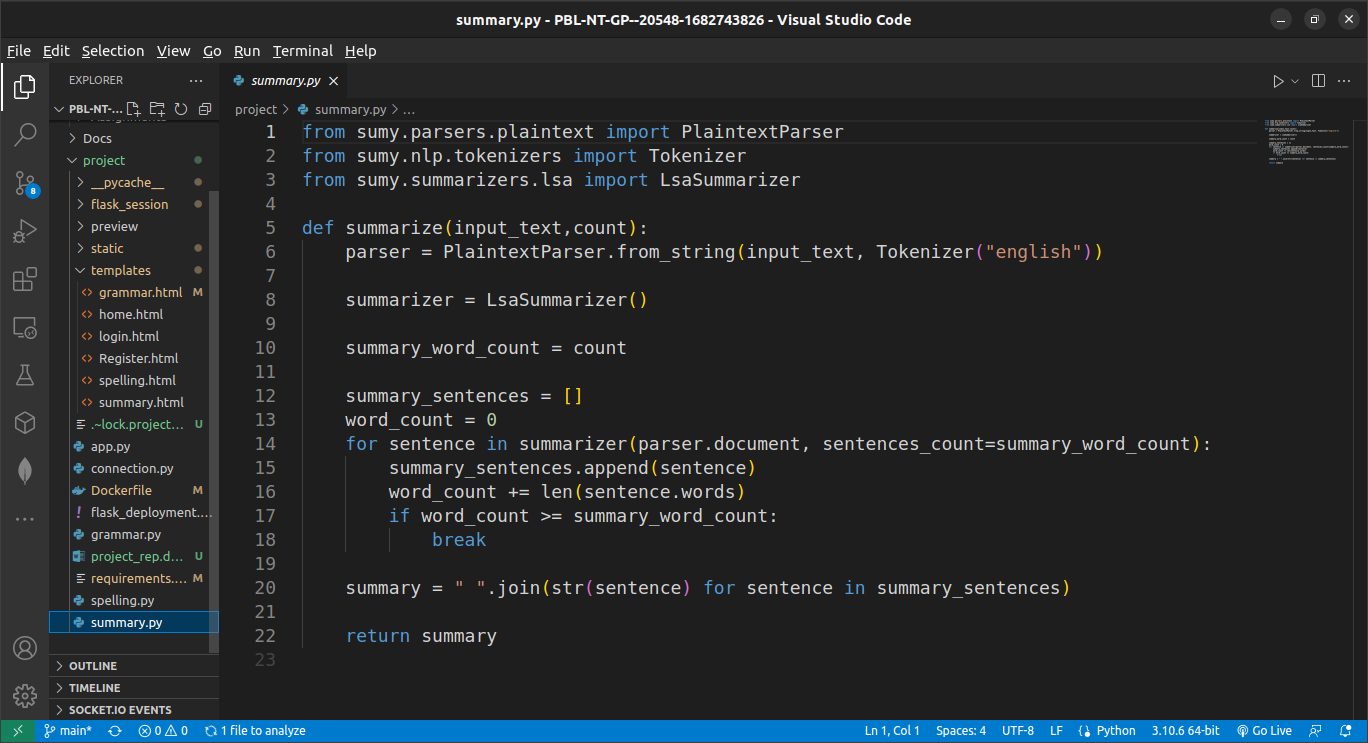
****

**sample grammar.py**

****

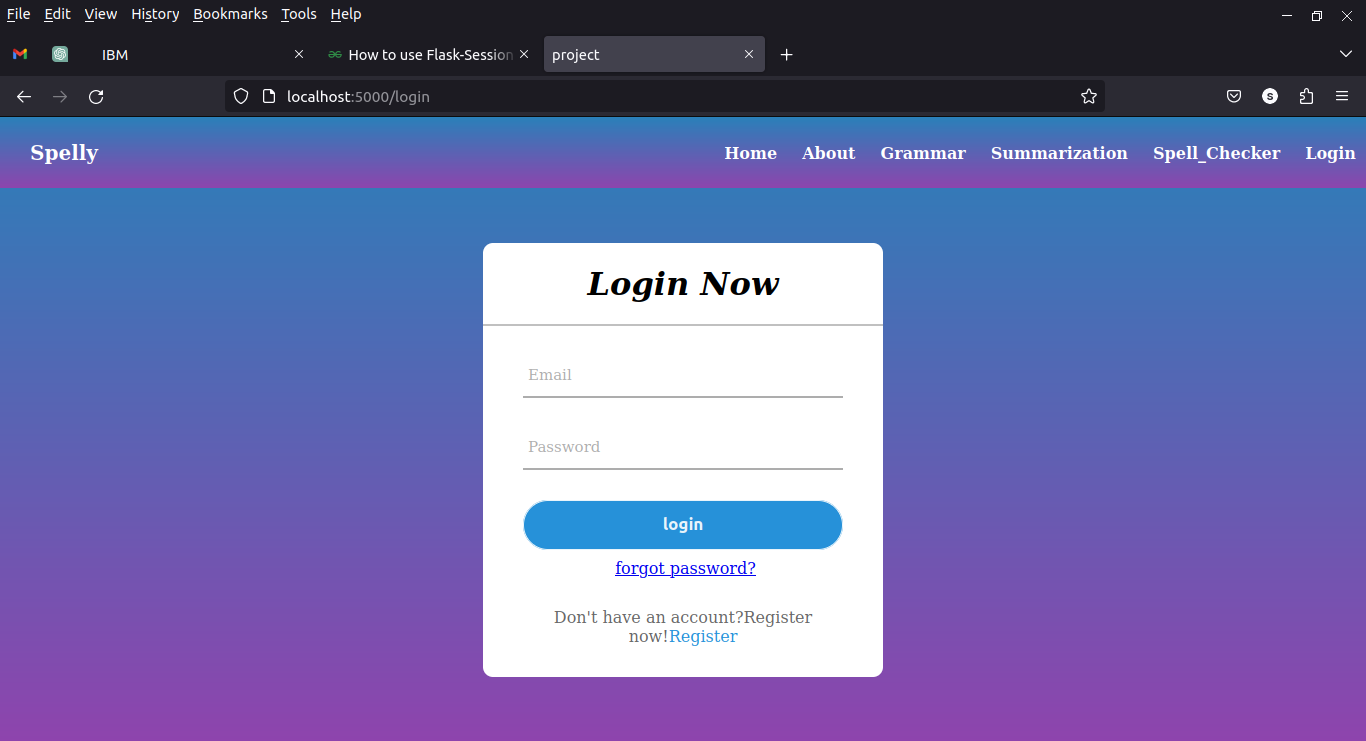
**sample spelling.py**



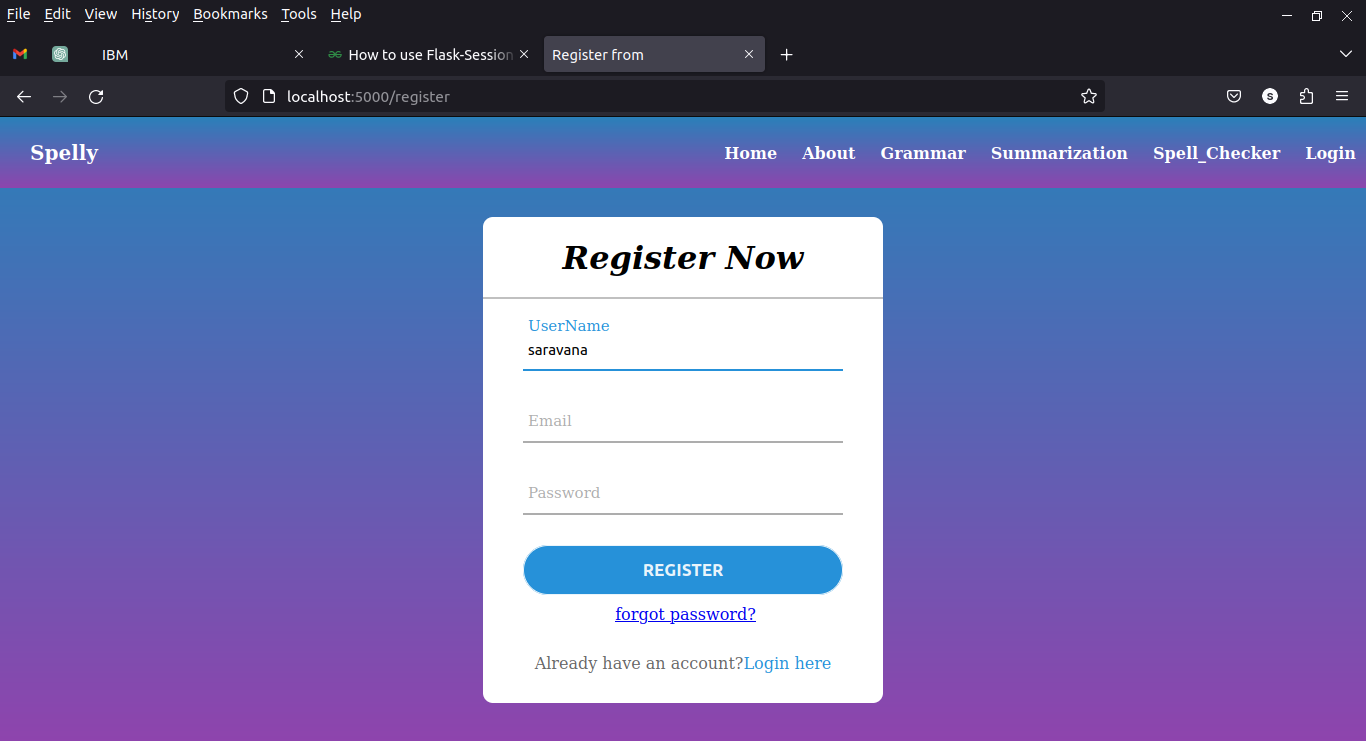
**sample summary.py**

**APPENDIX B – OUTPUT SCREENSHOTS**

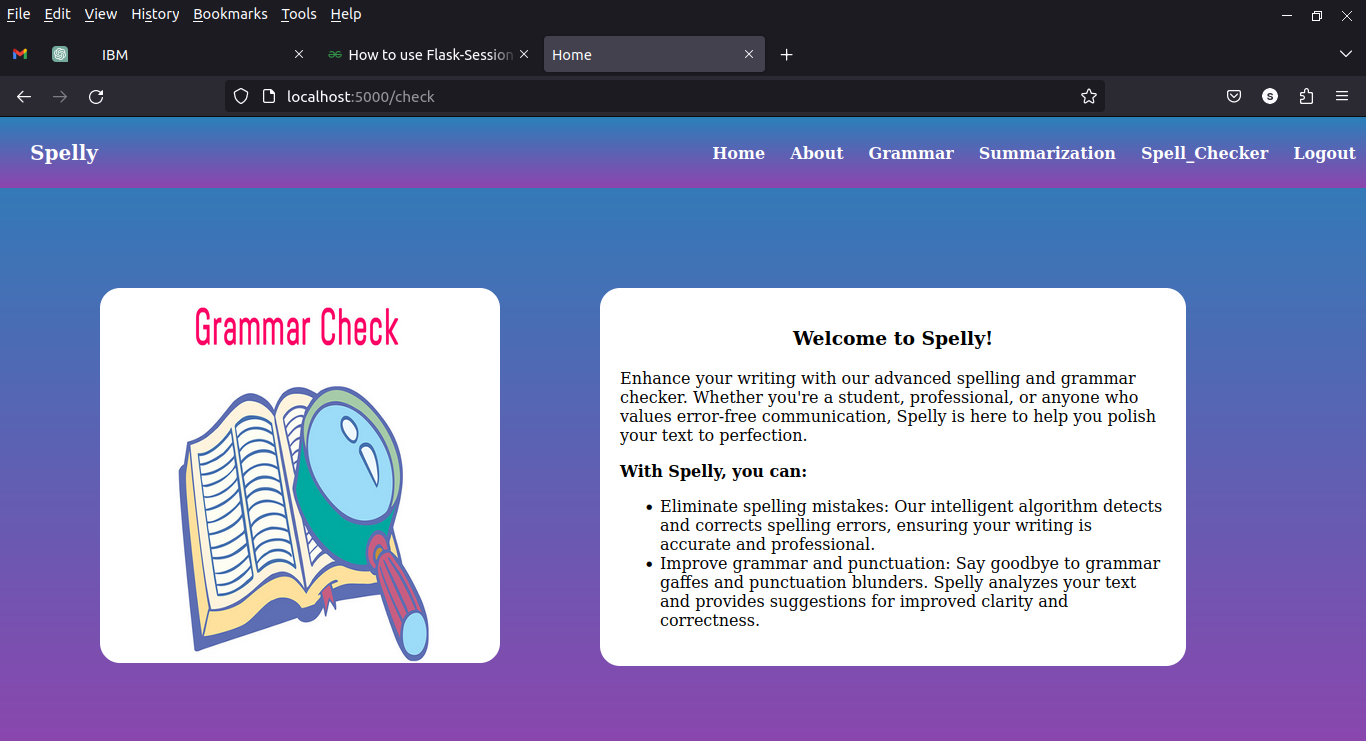
**sample Login page**

****

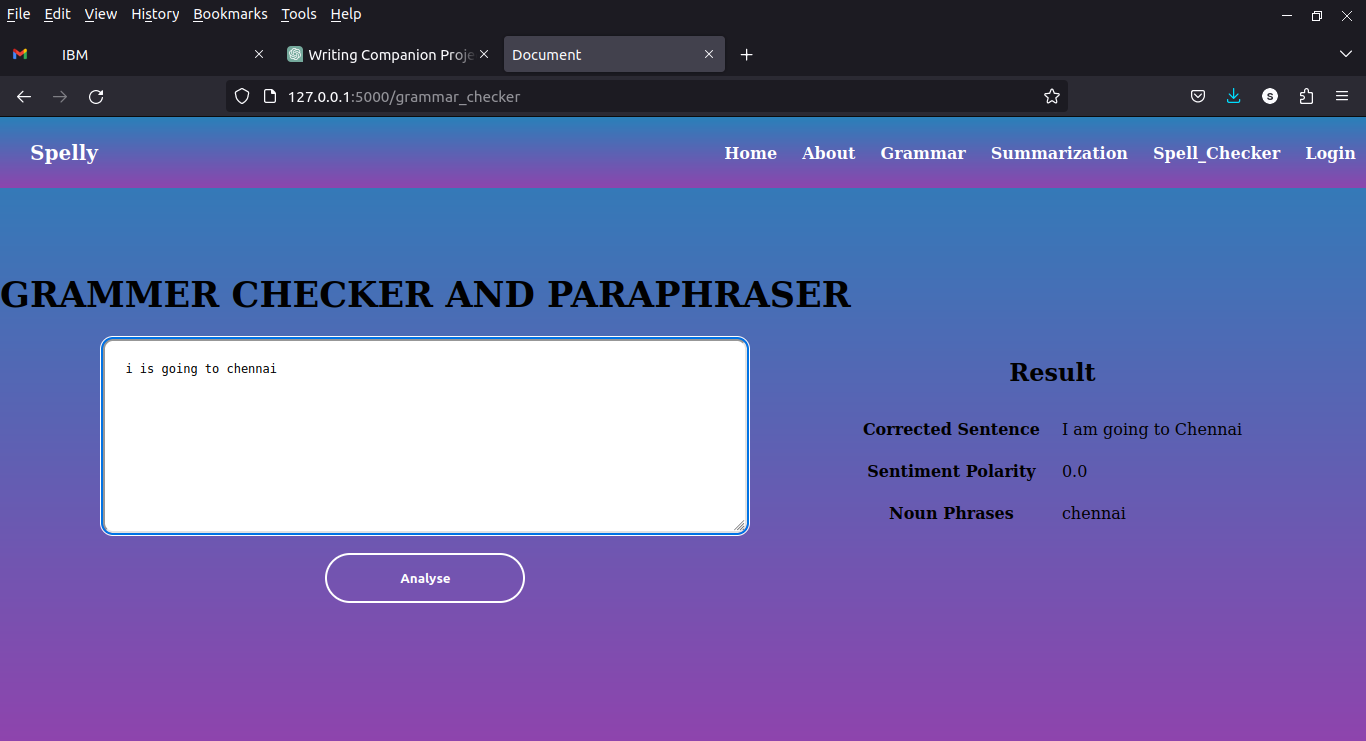
**sample Register page**



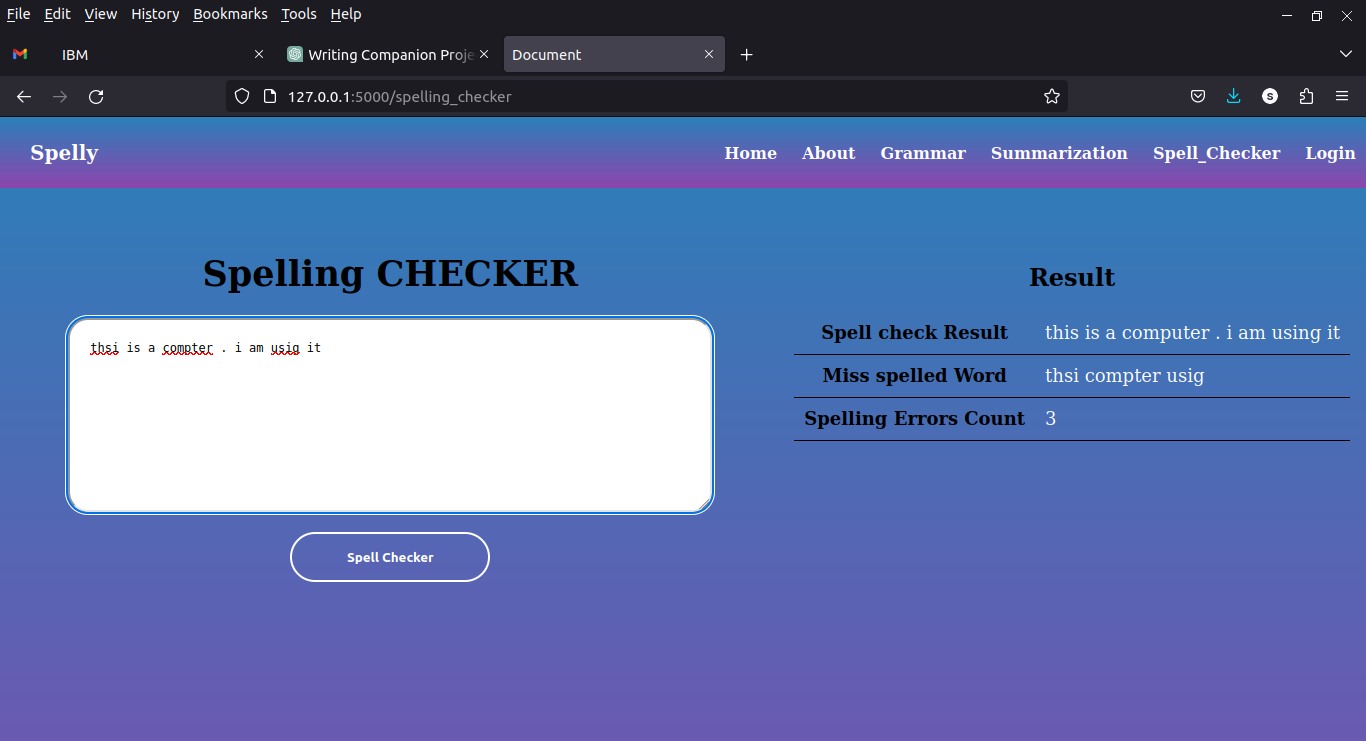
**sample home page**



**sample grammar Checker page**



**sample Spelling Checker page**



**sample summarizing page**

