**Design and Analysis of Algorithms:  
String Matching**

**Table of Contents**

**Language Used2**

**Libraries and Frameworks2**

Tkinter2

Numpy6

Matplotlib6

Time 7

**Instructions to run code8**

**Instructions to run code on jupyter notebook9**

**Run code via Anaconda10**

**Dependencies10**

time10

os 10

Numpy11

Pandas 11

Matplotlib11

**Algorithms11**

Algorithm of Brute Force12

### Algorithm of Knuth-Morris-Pratt (KMP)12

**Screenshots1**

**References13**

**Task3: Documentation:**

**Language Used:**

Language we used for doing a code is Python. Python is the most popular language for data science and research purposes. Python is high level language. It is easy to use and understand. Its syntax is clear, concise, easy to understand and maintain the code. Python is faster and more productive because of its well-organized object-oriented design, improved process management features, robust text processing and integration capabilities, and built-in unit testing framework. Python also has vast collection of libraries like numpy, pandas, matplotlib etc. By using these tools data manipulation and data visualization became easier.

Python is chosen for this project as it has several important advantages.

1. **Simplicity and Readability:** Python’s syntax is easy to understand which helps in reading and comprehending code. Because of its simplicity, developers can create and debug complex programs without the need for syntax-heavy languages. Python therefore facilitate development and teamwork, which is beneficial for academic projects and maintaining code clarity.
2. **Rapid Development:** Python has so many built in libraries and functions which speeds up development and assist in avoiding the need to construct complex programs from beginning. Searching text across large files is made possible by Python libraries. Python’s flexibility also enables rapid software testing and iteration, guaranteeing that features are added and enhanced more rapidly.
3. **GUI libraries:** As python is user friendly, so it also helps in Graphic User Interface (GUI) development. The libraries it includes are Tkinter, PyQt or Kivy. Tkinter is used for simple and basic interface while on the other hand PyQt or Kivy is used for complex and feature rich applications. These libraries aid developers in rapidly prototyping and testing interfaces while preserving a wide range of customization options. Because of robust community and cross-platform compatibility of python, programmers deign adaptable GUI that function flawlessly on a variety of operating systems with little change.
4. **Easy implementation of Complex Algorithms:** Python can easily implement complex algorithms like Brute Force and Knuth-Morris-Pratt (KMP). Timing libraries of python make it easier to assess and compare performances. This feature helps in evaluating Brute Force and Knuth-Morris-Pratt (KMP) time efficiency.

**Libraries and Frameworks:**

**Tkinter**

**Goal:** The application's graphical user interface (GUI), particularly the bonus section where the GUI permits user interaction, was developed using Tkinter, Python's standard GUI toolkit.

**Role in the Project**: Tkinter is a well-liked option for rapid and efficient interface creation because of its ease of use and seamless connection with Python. In order to create a user-friendly interface, Tkinter offers a variety of widgets, including buttons, labels, entry boxes, and checkboxes.

* Tkinter enabled the creation of an easy-to-use search input field with checkboxes for choices like "Whole Word Match" and "Case Sensitivity" in this project.
* Presenting search results in an orderly, legible format.
* Displaying labels and buttons to let users choose files, start searches, and see performance metrics.

**Why Tkinter?**

Because it strikes a compromise between functionality and simplicity, Tkinter is frequently used for Python GUI development, particularly for simple to intermediate applications. A closer look into why Tkinter was the best option for this project can be found here:

1. **Easily Accessible:**

* **Integrated Library:** Developers don't need to install any other packages or dependencies because Tkinter is already installed with Python. Because Tkinter is ready to use for everyone having Python installed, it simplifies distribution and cuts down on setup time.
* **Minimal System Needs:** Tkinter is appropriate for lower-end computers since it uses fewer resources to operate than more sophisticated frameworks (like PyQt or Kivy). Applications that don't require intricate animations or user interfaces will benefit greatly from this lightweight nature.

1. **Usability:**

* **Simple Syntax:** Even for novices, Tkinter's syntax is simple to grasp and straightforward. With just a few lines of code, developers may construct basic menus, buttons, labels, and entry boxes. This simplifies maintenance and lowers the amount of code required for simple GUIs.
* **Fast Prototyping**: Because Tkinter allows developers to quickly construct functioning interfaces without worrying about complicated setups, it is perfect for fast prototyping. This is helpful for projects where the GUI is a supporting element rather than the main one.

1. **Basic Widget sets:**

* **Minimal but Essential widgets:** Tkinter comes with a minimal yet adequate set of essential widgets that handle the majority of interface requirements for small to medium-sized applications. These include buttons, labels, text fields, checkboxes, radio buttons, and dropdown menus. In this project, Tkinter's widgets are more than enough because the GUI mostly acts as a control panel for inputs and search choices.
* **Variable Layouts:** With the help of Tkinter's support for many layout managers (pack, grid, and put), developers may arrange widgets to create a responsive and understandable user interface. This adaptability makes it possible to create patterns that are both straightforward and rather intricate.

1. **Low Coding Overhead:**

* **Direct Access to Widgets:** Without the need for additional classes or large configuration files, Tkinter's architecture enables developers to access and adjust widgets directly. Tkinter is perfect for applications with simple needs because of this method, which makes the code concise and simple to understand.
* **Instant Feedback and Event Binding:** Tkinter makes interaction handling simple by enabling developers to immediately link events (such as button clicks) to Python routines. As a result, functionality like search initiation and choice selection may be implemented more easily and with less code complexity.

1. **Platform Compatibility and Independence:**

* Cross-Platform Functionality: Without any code modifications, Tkinter programs may operate on Windows, macOS, and Linux. Because of its platform neutrality, users may access the program on any operating system.
* Consistent UI Elements: Tkinter's widgets display consistently across many operating systems since it makes use of the Tk GUI toolkit. Without requiring extra work to modify the UI for every device, this consistency offers a recognizable user experience.

1. **Records and Community Assistance:**
   * + **Rich Documentation**: Tkinter, one of the first Python GUI frameworks, contains a wealth of resources, tutorials, and documentation. This documentation is quite helpful, particularly for novice coders or for resolving frequent problems.
     + **Big Community:** Because of Tkinter's enormous user community, it's not too difficult to obtain help, tutorials, or answers to issues. This strong community support may speed up problem solving and drastically cut down on development time.
2. **Perfect for GUIs of the Basic to Moderate Level:**

* **Suitability for Project Requirements: Tkinter** is a perfect fit for this project because it is intended for simpler applications. It offers all the functionality required without overburdening the program with extraneous features or necessitating significant adjustments.
* **Balanced Approach:** Tkinter allows developers to create GUIs that are simple to create and maintain by providing a good balance between functionality and simplicity. This makes it an excellent option for apps that improve usability but do not prioritize the GUI.

In conclusion, Tkinter is an excellent option for developing GUIs such as the one required for this project due to its ease of use, built-in availability, minimal resource needs, and adequate variety of widgets.

**Numpy:**

**Purpose:**  
A key library for effective scientific and numerical computation in Python is called NumPy (Numerical Python). It offers an efficient framework for managing huge datasets, which is particularly helpful in applications requiring matrix manipulation, array operations, and other mathematical calculations.

1. **Effective Array Management: Compared** to ordinary Python lists, NumPy's ndarray structure is quicker and uses less memory, which makes it perfect for managing large amounts of data.
2. **Mathematical Functions:** NumPy facilitates effective computations without the use of intricate loops by providing a library of mathematical and statistical functions.  
   NumPy, which is designed for large-scale data processing, provides the foundation for scientific libraries such as Pandas, SciPy, and Matplotlib.

**Role:**

Effective array manipulation was made possible by NumPy's role in the project, which was essential for putting sorting and search algorithms into practice and evaluating big datasets.

* **Array Operations:** Repeated data manipulations are frequently necessary for sorting algorithms. These tasks are completed more quickly by NumPy's vectorized operations than by Python lists.
* **Memory Efficiency:** The project can manage big datasets well because to NumPy's contiguous memory blocks and compact data storage.
* **Advanced Calculations:** From organizing data types to computing metrics like average sort time and match frequencies, NumPy's built-in functions handled all essential operations.
* **Library Compatibility**: NumPy allows for easy data storage, manipulation, and display by integrating with Pandas and Matplotlib.

**Why Numpy?**Python lists' non-contiguous storage makes them slower for large datasets, notwithstanding their versatility. Faster processing and memory efficiency are provided by NumPy arrays, which were essential for handling big data handling and algorithm comparisons in this project.

**Matplotlib:**

**Purpose:**  
A popular Python package for creating static, animated, and interactive visualizations is called Matplotlib. Its adaptability and capacity to produce lucid visual comparisons—both crucial for assessing and showcasing algorithm performance—led to its selection for this project.  
  
**Role in the Project:**  
Matplotlib was essential to this study since it allowed for the visualization and comparison of the behavior and performance of the Knuth-Morris-Pratt (KMP) and Brute Force string-searching algorithms. This graphic gave insights into the two algorithms' efficiency and made performance disparities easier to comprehend.

**Features:**

* **Numerous Plot Types**: Matplotlib can create a wide range of plots, such as bar charts, histograms, pie charts, scatter plots, line plots, and more. Because of its adaptability, users may display data whichever best suits their needs.
* **Customizable**: Almost every element of a plot, including colors, markers, line styles, labels, and titles, can be altered by users. This makes it possible to produce figures that are suitable for publication.
* **Integration with Pandas and NumPy:** Matplotlib easily integrates with Pandas DataFrames and NumPy arrays, facilitating data processing and the visualization of complicated datasets.
* **Interactive Plotting**: Users may zoom, pan, and update charts dynamically with Matplotlib's support for interactive charting. This function is very helpful for analyzing exploratory data.

### **Why Use Matplotlib?**

* **Versatility**: Supports a wide range of plot types, including line plots, bar charts, histograms, scatter plots, and 3D plots, making it suitable for various data visualization needs.
* **Customizability**: Offers extensive options for customizing plots, including colors, markers, line styles, axes labels, and titles, allowing for publication-quality graphics.
* **Integration**: Works seamlessly with essential libraries like NumPy and Pandas, enabling easy plotting of numerical and tabular data.
* **Interactive and Animated Plots**: Supports interactive features like zooming and panning, as well as creating animated visualizations to illustrate changes over time.
* **Multiple Output Formats**: Allows saving plots in various formats, including PNG, PDF, SVG, and EPS, facilitating easy sharing and publication.
* **Active Community**: A large, active community ensures extensive documentation and support, helping users troubleshoot and learn.
* **User-Friendly for All Levels**: Suitable for both beginners and advanced users, offering simple syntax for basic tasks and advanced features for expert needs.
* **Open Source**: Free to use, making it accessible to everyone without licensing costs.
* **Jupyter Notebook Compatibility**: Works well within Jupyter Notebooks, enhancing the interactive data analysis experience.

### **Conclusion**

### Matplotlib is a powerful and flexible library for data visualization in Python, offering the versatility, customizability, and integration necessary for effective data representation.

**Time Module**  
  
**Purpose:**

Python's time module offers time-tracking methods, which are crucial for gauging the effectiveness and performance of programming.  
**Role in the Project:**

One of the project's main requirements was to measure the Brute Force and KMP algorithms' performance.

* The time module made it possible to compare the efficiency of several algorithms directly by calculating how long it takes for each algorithm to do its task.
* Keeping track of and showing the beginning and ending timings of searches, which was useful for recording performance in various scenarios.

**Why Time Module?**  
The Reason for the Time Module Without requiring other libraries or setups, the time module offers a straightforward yet precise method of measuring execution time, which is crucial for performance research.

## Instructions to Run the Code

To set up and run the word search application, follow these detailed steps:

1. **Install Python**: Ensure Python (version 3.6 or higher) is installed on your system. You can download it from the official Python website.
2. **Install Required Libraries**: Open a terminal or command prompt and run the following commands to install the required libraries:

bash

Copy code

pip install numpy pandas matplotlib

1. **Clone the Repository**: If the code is hosted on a version control platform (like GitHub), clone the repository using:

bash

Copy code

git clone <repository-url>

1. **Navigate to the Project Directory**:

bash

Copy code

cd <project-directory>

1. **Run the Application**: Execute the main script (replace main.py with your script's filename):

bash

Copy code

python main.py

1. **Input Search Terms**: Use the GUI to input search terms and select options for case sensitivity and whole word matching. Click the search button to initiate the search.

### **Instructions to Run Code on Jupyter Notebook**

1. **Install Jupyter Notebook**:

bash

Copy code

pip install notebook

1. **Start Jupyter Notebook**:
   * Navigate to your desired directory:

bash

Copy code

cd path/to/your/directory

* + Start the notebook:

bash

Copy code

jupyter notebook

1. **Create a New Notebook**:
   * In the dashboard, click **New** and select **Python 3**.
2. **Writing and Running Code**:
   * Enter code in a cell.
   * Run a cell using:
     1. **Run** button in the toolbar.
     2. **Shift + Enter** (runs and moves to the next cell).
     3. **Ctrl + Enter** (runs and stays in the same cell).
3. **Install Required Libraries**:
   * Install libraries directly in a cell:

python

Copy code

!pip install numpy pandas matplotlib

1. **Save Your Notebook**:
   * Click the **Save** icon or press **Ctrl + S**.
2. **Closing the Notebook**:
   * Close the notebook tab. Stop the server by pressing **Ctrl + C** in the terminal.

### **Conclusion**

These steps will help you quickly run code in Jupyter Notebook, making coding and visualization efficient. For issues, check the terminal or consult the Jupyter documentation.

### **Run Jupyter Notebook via Anaconda**

1. **Install Anaconda**:
   * Download and install from the Anaconda website.
2. **Open Anaconda Navigator**:
   * Launch **Anaconda Navigator**.
3. **Launch Jupyter Notebook**:
   * Click **Launch** under **Jupyter Notebook**.
4. **Create a New Notebook**:
   * In the Jupyter dashboard, click **New** and select **Python 3**.
5. **Write and Run Code**:
   * Type code in a cell and run it using:
     + **Run** button.
     + **Shift + Enter** (next cell).
     + **Ctrl + Enter** (same cell).
6. **Install Libraries**:
   * Install required libraries in a cell:

python

Copy code

!pip install numpy pandas matplotlib

1. **Save Your Notebook**:
   * Click the **Save** icon or press **Ctrl + S**.
2. **Close Notebook**:
   * Close the tab when finished. Exit Anaconda Navigator to stop the Jupyter server.

### **Conclusion**

These steps will help you quickly use Jupyter Notebook via Anaconda. For issues, consult the Anaconda documentation.

**Dependencies:**

The libraries used to build the Brute Force and Knuth-Morris-Pratt (KMP) algorithms are mostly those that make simple programming operations like handling strings and input/output management easier. The primary dependencies you may utilize are as follows:

Dependencies for the Python Standard Library used in Brute Force and KMP Implementations:

* **time:** Used to calculate how long each algorithm takes to execute.
* **OS:** Used for path management and file handling (reading text files).
* **NumPy:** NumPy can be useful for managing any array operations or for working with bigger datasets, even if it isn't necessarily required for the fundamental Brute Force and KMP implementations.
* **Pandas:** If you need to show search results in a more structured manner, this is helpful for handling and storing them in a structured format (like DataFrames).
* **Matplotlib:** useful for displaying the algorithms' performance outcomes, particularly when comparing execution durations visually.

**Algorithms:**

**1. Algorithm of Brute Force:**  
A brute force algorithm is a straightforward, all-inclusive search approach that methodically looks at every possibility until the solution to a problem is found. When the problem is little enough to allow for a thorough study, this general method of problem-solving is used. However, brute force methods are ineffective for large-scale problems because to their great temporal complexity.

**Important lesson:**

* **Methodical Listing:** Brute force algorithms look at every possible fix for a problem, often in a systematic and thorough manner. This entails trying each choice in a predetermined order.
* **Relevance:** Brute force is the best approach when the problem space is limited and conveniently explorable in a reasonable amount of time. For bigger problem scenarios, the algorithm's temporal complexity becomes impractical.
* **Absence of heuristics or optimization:** Heuristics and optimization are not used by brute force techniques. They rely on employing astute pruning or heuristics to examine every possible result without ruling out any.

**Brute force algorithm's Features:**

* It is a simple, direct, and intuitive method of problem-solving where all potential approaches or solutions to a certain issue are listed.
* The brute force method is used to solve many issues in daily life, such as determining the quickest route by examining every possible route to a local market.
* maximizing the rack spaces by arranging the books in a rack in every way possible, etc.
* Despite the possibility of optimum algorithms, everyday tasks employ a brute force approach.

**Benefits and drawbacks of the brute force algorithm:  
Advantages:**

* By enumerating every potential candidate solution for the problem, the brute force method provides a sure-fire method of determining the right answer.
* It is a general approach that is not restricted to any particular issue domain.   
  Smaller, easier issues are best solved using the brute force approach.
* It may be used as a standard for comparison and is renowned for its simplicity.

**Disadvantages:**

* The brute force method is not very effective. Algorithm analysis frequently surpasses the O(N!) order of growth for real-time issues.
* This approach depends less on a well-designed algorithm and more on sacrificing a computer system's ability to solve an issue.
* Algorithms using brute force are sluggish.
* Compared to algorithms built utilizing other design paradigms, brute force algorithms lack creativity and constructiveness.

In conclusion   
Although it is inefficient and requires a lot of run time, the brute force approach is a strategy that ensures solutions for issues in any domain. It also helps solve smaller problems and offers a solution that may be used as a benchmark for assessing other design strategies.

### **Algorithm of Knuth-Morris-Pratt (KMP):**

A string-searching method called the Knuth-Morris-Pratt (KMP) algorithm effectively locates instances of a pattern in a text. KMP is especially useful for bigger datasets since it enhances search speed by minimizing pointless comparisons, in contrast to the brute force method. The pattern is preprocessed by the algorithm to produce a partial match table, which enables it to exclude text passages that have previously been matched.

### **Important Lessons:**

* **Efficient Matching**: KMP greatly enhances performance by avoiding reexamining text characters that have already been matched by using previously acquired knowledge about the pattern.
* **Preprocessing**: KMP can decrease the number of comparisons performed during the search by generating a partial match table, sometimes referred to as the "prefix table," which identifies the subsequent spots in the pattern and text to examine.
* **Performance Guarantee**: The KMP technique is appropriate for large-scale string matching problems since it operates in linear time, O(N + M), where N is the text length and M is the pattern length.

### **Features of KMP Algorithm:**

* **Preprocessing Step**: KMP includes a preprocessing step to construct the partial match table, which is essential for efficient searching.
* **Linear Time Complexity**: The overall time complexity of KMP ensures that both the preprocessing and searching phases are linear, which is a significant improvement over brute force method.
* **Robust for Repeated Patterns**: KMP is particularly effective for searching patterns that may have repeating elements, as it can skip redundant comparisons.

### **Benefits and Drawbacks of the KMP Algorithm:**

#### **Advantages:**

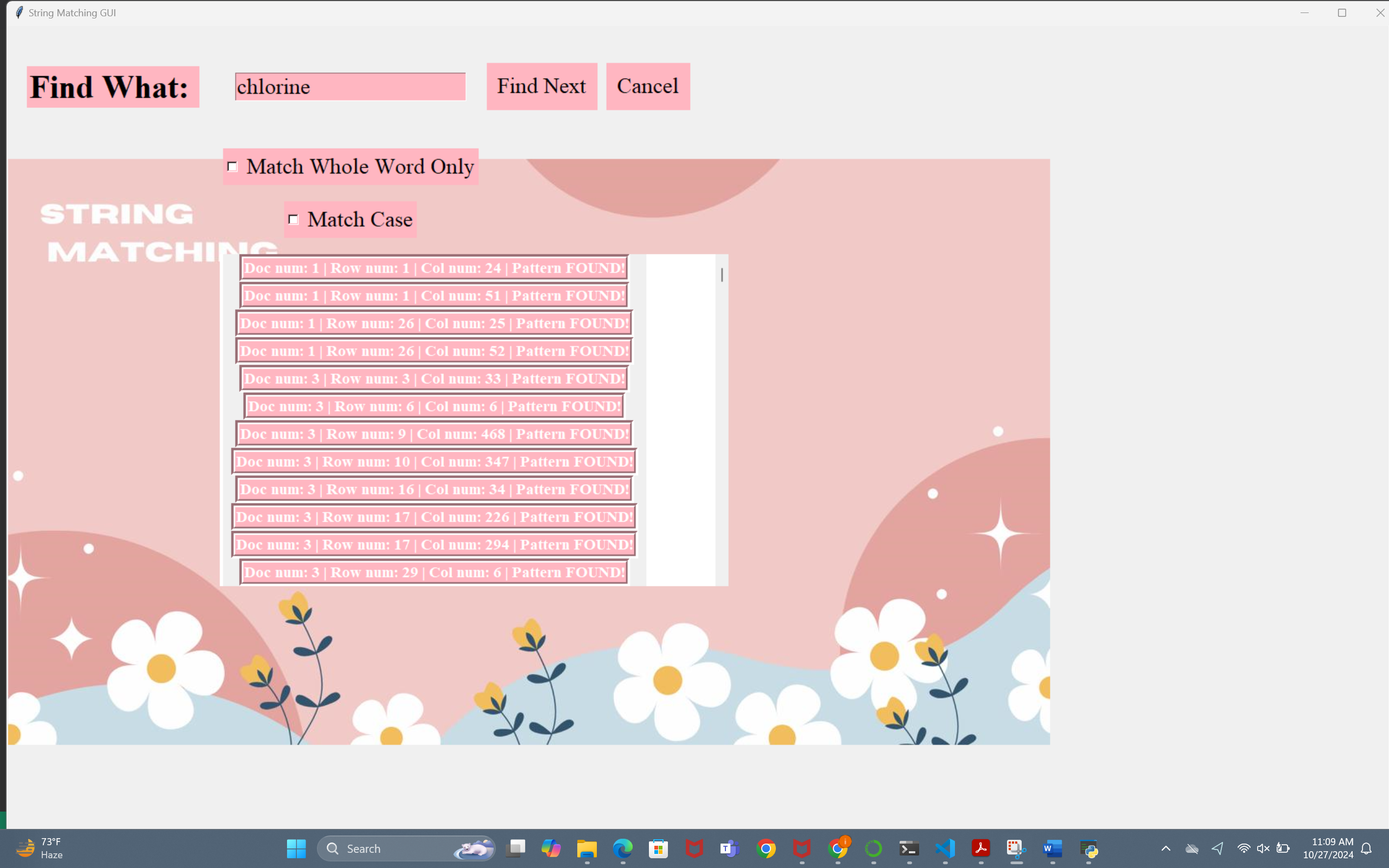
* **Efficiency**: KMP reduces the number of character comparisons needed by utilizing information from previous matches, making it much faster than brute force for larger texts.
* **Linear Complexity**: The O(N + M) complexity allows for efficient searching even in lengthy texts, making it suitable for practical applications like text editors and search engines.
* **No Backtracking**: Unlike the brute force algorithm, KMP does not backtrack in the text, which minimizes unnecessary computations.

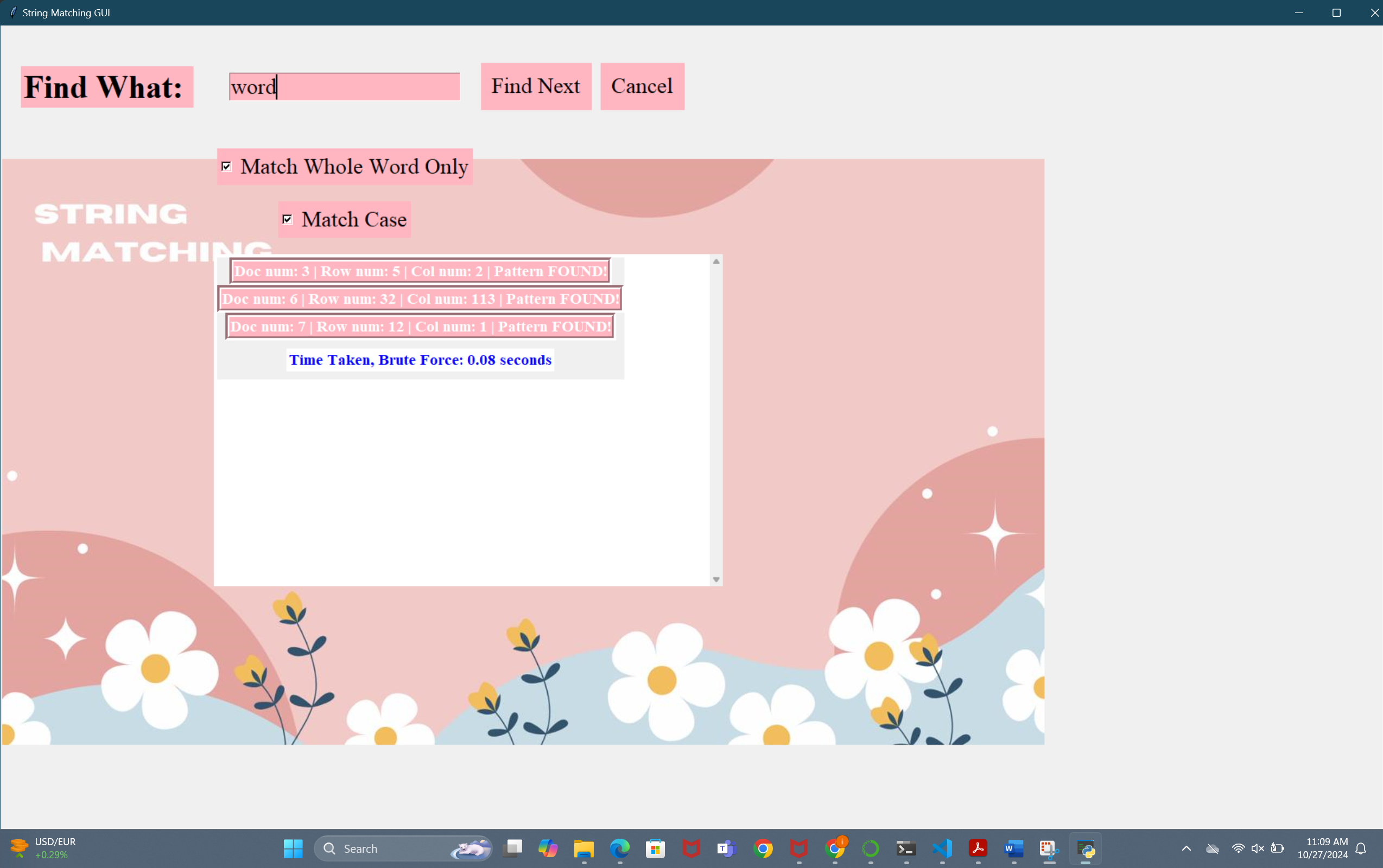
#### **Disadvantages:**

* **Complexity of Implementation**: The preprocessing step and the construction of the partial match table can be complex, making KMP more challenging to implement than brute force methods.
* **Overhead**: For very small datasets, the overhead of preprocessing may not justify the efficiency gains, as the brute force method may perform adequately in such cases.
* **Memory Usage**: The partial match table requires additional memory, which may be a consideration in memory-constrained environments.

In conclusion, the Knuth-Morris-Pratt algorithm is a strong and effective string-matching method that performs better than brute force approaches, particularly when dealing with lengthy texts or recurring patterns. It is a better option for real-world text searching and manipulation applications because to its linear time complexity and avoidance of repeated comparisons, but having a steeper learning curve and maybe requiring more initial setup.

**Screenshots:**





**Work division:**

#### **Member 1: Khadija Haider: Algorithms**

* **Responsibilities:**
  + - Implement the **Brute Force** and **KMP** algorithms for string searching.
    - Ensure both algorithms can handle various matching scenarios as specified.
    - Optimize the algorithms for performance where possible.

**Member 2:Urwah Ejaz: Front-End Development**

* **Responsibilities:**
  + - Design and implement the **Graphical User Interface (GUI)**.
    - Create input fields for users to enter search terms and options (whole word match, case sensitivity).
    - Implement buttons to initiate the search and display results.
    - Display the output clearly, including time taken for each algorithm.
    - Ensure the GUI is intuitive and user-friendly, adhering to usability principles.

#### **Member 3: Fizza Sarfraz: Documentation**

* **Responsibilities:**
  + - Document the overall project, including language used, libraries used, instructions to run the code, dependencies, algorithm’s features, advantages and disadvantages, screenshots and references.
    - Provide detailed user documentation for the GUI, including how to input search terms and interpret results.
    - Create technical documentation on the algorithms used, including their implementation details and performance evaluations.
    - Assist in compiling a comparison report of the performance characteristics of both algorithms.

**Conclusion:**

The goal of facilitating effective text searches across numerous files is successfully achieved by the GUI-based word search program. We gave consumers a powerful tool for string searching and performance comparison by integrating the Brute Force and Knuth-Morris-Pratt (KMP) algorithms. Usability is improved by the user-friendly interface, which makes it simple to enter search phrases and configure matching settings.

Reliability is ensured and user confidence is increased by paying attention to non-functional needs like error management. All things considered, the project provides insightful information about the performance of each algorithm by highlighting its advantages and disadvantages. Future developments may concentrate on adding more search possibilities and taking user input into account to make it even better.

**References**

* Python Documentation: <https://docs.python.org/3/>
* NumPy Documentation: https://numpy.org/doc/
* Pandas Documentation: https://pandas.pydata.org/docs/
* Matplotlib Documentation: https://matplotlib.org/stable/contents.html
* Tkinter Documentation: <https://docs.python.org/3/library/tkinter.html>
* Brute Force Documentation: <https://www.geeksforgeeks.org/brute-force-approach-and-its-pros-and-cons/>
* KMP Documentation: <https://www.geeksforgeeks.org/kmp-algorithm-for-pattern-searching/>