TEACHER BUDDY

EFFORTLESS GRADING, EXCEPTIONAL TEACHING.

Name: Rishi Ahuja

Roll no:

St. Francis de Sales School

Certificate of Completion

This is to certify that Rishi Ahuja has successfully completed the Computer Science project for the academic year 2023-2024. This project was conducted under the guidance of Ms. Sapna Gupta at St. Francis de Sales School.
Congratulations on successfully completing the Computer Science project!

Ms. Sapna Gupta
HOD Computer Science, SFS

Acknowledgement

I would like to express my sincere appreciation and acknowledgment to all individuals who have played a significant role in the successful completion of this project.

First and foremost, I extend my deepest gratitude to my esteemed teacher, Ms. Sapna Gupta, for her invaluable guidance, patience, and unwavering support throughout this endeavour.

Their expertise and dedication have been instrumental in shaping my understanding and enhancing my skills.
I would like to extend my heartfelt thanks to my parents for their continuous support, understanding, and belief in my abilities.

Finally, I would like to express my gratitude to all the authors, researchers, and experts whose work and insights have influenced and contributed to the development of my project.

Their valuable contributions have broadened my perspective and enriched the scope of this program.

With utmost gratitude and appreciation, I extend my heartfelt thanks to everyone involved.

* * *

Index

Title	Pg. No.
1. Introduction	6
2. Requirements	8
3. System Design • Python Modules • Inbuilt Functions • UDF	9
4. Source Code	20
5. Outputs	49
6. References	57

* * *

TEACHER BUDDY

EFFORTLESS GRADING, EXCEPTIONAL TEACHING.

Introduction

Introducing,

TEACHER BUDDY : Effortless Grading, Exceptional Teaching.

1. **Student Management Made Easy:**
   - Effortlessly add and organize students within the app.
   - Streamline classroom administration for optimal efficiency.

2. **Seamless Assignment Collection:**
   - Simplify the assignment submission process for both teachers and students.
   - Centralize and manage assignments within the app's user-friendly interface.

3. **Deterrent to Delinquency:**
   - Keep track of defaulters and overdue assignments.
   - Enable timely intervention with a comprehensive view of student performance.

4. **Plagiarism Detection with Precision:**
   - Detect plagiarism effortlessly with the app's built-in comparison tool.
   - Pinpoint similarities between assignments and identify potential academic misconduct.

5. **Percentage-Based Plagiarism Analysis:**
    - Gauge the extent of plagiarism by determining the percentage of similarity between assignments.
    - Facilitate fair and objective assessment of academic integrity.

6. **Automated Grading for Efficiency:**
    - Create grading rules tailored to your teaching preferences.
    - Leverage Natural Language Processing (NLP) to automate the grading process, saving time and ensuring consistency.

TEACHER BUDDY is not just an app; it's your ally in creating a well-organized, plagiarism-aware, and efficiently managed classroom environment. Elevate your teaching experience with the power of technology.

***

System Requirements

Minimum System Requirements:

- **Processor:** Dual-core processor, 1.6 GHz or equivalent.
- **RAM:** 2 GB.
- **Storage:** 20 GB of free disk space.
- **Operating System:** Windows 7/8/10, macOS 10.12+, or. modern Linux distribution.
- **Python:** Python 3.x installed.
- **MySQL:** MySQL server installed and accessible.
- **NLTK:** NLTK library installed.
- **Tkinter:** Tkinter library (usually included with Python installations).
- **SQL Connectivity:** Appropriate Python SQL library installed
- **Text File Connectivity:** Standard text file reading/writing capabilities.

Recommended System Requirements:

- **Processor:** Quad-core processor, 2.5 GHz or higher.
- **RAM:** 8 GB or higher.
- **Storage:** 50 GB of free disk space or more.
- **Operating System:** Windows 10, macOS 11+, or a recent Linux distribution.
- **Python:** Latest stable version of Python 3.x.
- **MySQL:** Latest stable version.
- **NLTK:** Latest version for additional features and improvements.
- **Tkinter:** Latest version for potential bug fixes and enhancements.
- **SQL Connectivity:** Latest version of the SQL library for compatibility and security updates.
- **Text File Connectivity:** Reliable and efficient storage with the latest Python capabilities.

***

System Design

python **Modules**

1. **Tkinter:**
    - Tkinter is the de facto standard GUI (Graphical User Interface) toolkit that comes with python. It facilitates the creation of windows, dialogs, buttons, and other GUI elements, allowing developers to build interactive and user-friendly applications. Tkinter is widely used for its simplicity and ease of integration with python applications.

2. **ttk:**
    - The **ttk** module, an extension of Tkinter, introduces themed widgets to enhance the visual aesthetics of the user interface. Themed widgets provide a modern and consistent look across different operating systems. The ttk module includes advanced widgets and styling options, contributing to a more polished and professional appearance for applications.

3. **messagebox:**
    - The **messagebox** module, part of Tkinter, is essential for creating modal dialogs to communicate with users. It offers functions for displaying various types of messages, including information, warnings, errors, and queries for user confirmation. This module streamlines the interaction between the application and the user, improving the overall user experience.

4. **filedialog:**
    - The **filedialog** module, integrated with Tkinter, simplifies file-related operations. It provides dialogs for opening and saving files, enabling users to navigate through their file systems interactively. This module is instrumental in

facilitating file input and output functionality within the application.

5. **csv:**
   - The **csv** module is a built-in python module designed for handling CSV (Comma-Separated Values) files. It streamlines the process of reading from and writing to CSV files, making it easier to work with tabular data in a plain-text format. The csv module is particularly useful for applications dealing with data import/export and manipulation.

6. **mysql.connector:**
   - The **mysql.connector** module serves as a python driver for MySQL databases. It establishes a connection between python applications and MySQL databases, allowing for the execution of SQL queries and the management of database operations. This module is vital for applications requiring database interaction and management.

7. **time:**
   - The **time** module provides functions for working with time-related operations. Developers can utilize this module to measure time intervals, introduce delays in their applications, and work with timestamps. The time module is beneficial in scenarios where precise time-based functionalities are required.

8. **nltk:**
   - NLTK, or the Natural Language Toolkit, is a comprehensive library for natural language processing in python. It encompasses tools for various linguistic tasks, including tokenization, stemming, tagging, parsing, and more. NLTK is widely employed in applications involving text analysis, machine learning, and linguistic research.

9. **re:**
    - The **re** module in Python stands for regular expressions. It provides support for working with regular expressions, enabling developers to define and manipulate string patterns. Regular expressions are powerful tools for searching, matching, and manipulating text, making the re module valuable for text processing tasks.

10. **nltk.tokenize:**
    - The **nltk.tokenize** module within NLTK focuses specifically on tokenization, a fundamental step in natural language processing. Tokenization involves breaking down text into smaller units, such as words or sentences. This module offers various tokenization methods, allowing developers to choose the most suitable approach for their specific application needs.

11. **nltk.corpus:**
    - The **nltk.corpus** module in NLTK provides access to linguistic corpora, which are extensive collections of text used for linguistic analysis and research. These corpora include datasets that cover a wide range of languages and genres, making them valuable resources for training and testing natural language processing models.

These modules collectively empower developers to create sophisticated applications with graphical interfaces, robust database interactions, effective text processing capabilities, and advanced natural language processing functionalities. Their integration enhances the overall functionality and user experience of the application.

***

Inbuilt Functions

Here is a list of built-in functions used in the provided code:

- `print()`
- `input()`
- `open()`
- `len()`
- `time.sleep()`
- `int()`
- `str()`
- `filedialog.askopenfilename()`
- `messagebox.showerror()`
- `messagebox.showinfo()`
- `connect()`
- `cursor()`
- `execute()`
- `commit()`
- `tk.StringVar()`
- `tk.Text()`
- `tk.Entry()`
- `tk.Label()`
- `tk.Button()`
- `tk.Tk()`
- `tk.Frame()`
- `ttk.Progressbar()`

* * *

User-defined Functions

1. `add_student_sql()` function:

- **Description:**
  - The `add_student_sql()` function facilitates the addition of student information to a MySQL database.
- **Purpose:**
  - To create a structured storage system for managing student data.
- **Nested Functions:**
  - `create_table()`:
    - **Description:** Creates a table in the MySQL database for storing student information.
    - **Purpose:** Establishes a structured data format for student records.
  - `add_student()`:
    - **Description:** Adds a student's information to the database.
    - **Purpose:** To insert new student data into the established database structure.
  - `browse_file()`:
    - **Description:** Opens a file dialog for selecting a file.
    - **Purpose:** Allows the user to choose a file, typically for uploading student assignments.
  - `insert(x)`:
    - **Description:** Inserts student information into the database.
    - **Purpose:** To execute the SQL query for inserting student data.

2. `add_student_csv()`:

- **Description***:* The main function that serves as the entry point for the program.
- **Purpose***:* Sets up the GUI for adding student information, handling the interface, and managing data.
- **Nested Functions:**
  - `add_student()`:
    - **Description***:* Validates and adds student information to the lists.
    - **Purpose***:* Takes input from the GUI, validates it, and updates the student information lists.
  - `browse_file()`:
    - **Description***:* Opens a file dialog to browse and select a file path.
    - **Purpose***:* Allows users to select a file for the student's assignment.
  - `save_to_csv()`:
    - **Description***:* Saves student information to a CSV file.
    - **Purpose***:* Writes the student details from the lists to a CSV file for storage.
  - `close_window()`:
    - **Description***:* Closes the GUI window.
    - **Purpose***:* Exits the program when the user clicks the "Close" button.

3. `view_classroom()` SQL:

- **Description***:* The main function that initiates the GUI for viewing classroom information.
- **Purpose***:* Connects to a MySQL database, defines and executes SQL queries to fetch student data, and displays the data in a Tkinter GUI.
- **Nested Functions:**
  - `fetch_students(order_by="roll_num", filters=None)`
    - **Description:** Fetches student data from the database based on specified ordering and filters.
    - **Purpose:** Retrieves student information using SQL queries, allowing for customization of ordering and filtering.
  - `display_classroom(order_by="roll_num")`
    - **Description:** Defines the structure of the Tkinter GUI for displaying classroom information.
    - **Purpose:** Handles the layout, styling, and interactive elements of the GUI, including filtering and ordering options.
  - `update_display(event=None)`
    - **Description:** Updates the displayed student information based on selected filters and ordering.
    - **Purpose:** Invoked when filter values change, ensuring the GUI reflects the updated student data according to specified criteria.
  - `clear_filters()`
    - **Description:** Resets filter values to their default state, clearing any applied filters.
    - **Purpose:** Allows users to easily reset the filters and view the full set of student data.

4. `view_classroom()csv:`

- **Description:** The main function that initializes a Tkinter GUI for displaying classroom information.
- **Purpose:** Connects to a CSV file ("students.csv"), defines ordering and filtering functions, and displays student data in a Tkinter Treeview.
- **Nested Functions:**
  - `fetch_students(order_by="roll_num", filters=None):`
    - **Description:** Reads student data from a CSV file, applies optional filters, and sorts the data based on specified ordering.
    - **Purpose:** Provides a way to retrieve and organize student information for display in the Tkinter GUI.
  - `display_classroom():`
    - **Description:** Defines the structure and layout of the Tkinter GUI for displaying classroom information.
    - **Purpose:** Handles GUI components, such as labels, comboboxes, buttons, and the Treeview, as well as their interactions.
  - `update_display():`
    - **Description:** Updates the displayed student information based on selected filters and ordering.
    - **Purpose:** Triggered by combobox selection events to refresh the displayed data according to specified criteria.

5. `compare_text()`:

- **Description:**
  - The `compare_text()` function compares the similarity between two texts, highlighting common words and displaying a progress bar.
- **Purpose:**
  - To aid in comparing the similarity of text content, potentially for plagiarism detection.
- **Nested Functions:**
  - `load_file_or_display_contents(entry, text_widget, name_var)`:
    - **Description:** Loads a file or displays its contents in a text widget.
    - **Purpose:** Enables the user to input text for comparison, either by loading a file or directly entering text.
  - `compare_text()`:
    - **Description:** Compares the text content and calculates the similarity.
    - **Purpose:** To assess the degree of similarity between two text inputs.
  - `reset_text()`:
    - **Description:** Resets the text in the text widgets and progress bar.
    - **Purpose:** Clears the input fields and progress bar for a new comparison.
  - `close_app()`:
    - **Description:** Closes the application.
    - **Purpose:** Provides a means to exit the text comparison tool.
  - `highlight_text(text_widget, words, color, style)`:
    - **Description:** Highlights common words in the text widgets.
    - **Purpose:** Visually represents words shared between the two texts.

6. `score()` function:

- **Description:**
  - The `score()` function serves as the main assessment tool for evaluating a student's paragraph against specific criteria.
- **Purpose:**
  - To provide a comprehensive evaluation of a student's writing skills by considering factors such as word count, the starting phrase, and the usage of certain words.
- **Nested Functions:**
  - `StopWords(x)`:
    - **Description:** Removes stop words from the given text.
    - **Purpose:** To filter out common words that do not contribute significantly to the content.
  - `WordCount(y)`:
    - **Description:** Calculates the length of a paragraph and awards points based on predefined criteria.
    - **Purpose:** To assess the student's adherence to length requirements.
  - `Regular(o)`:
    - **Description:** Checks if the paragraph starts with the required phrase.
    - **Purpose:** To verify if the student follows the specified starting phrase.
  - `types(v)`:
    - **Description:** Identifies and categorizes words in the paragraph.
    - **Purpose:** To analyze the types of words used by the student.
  - `calculate()`:
    - **Description:** Calculates the final score based on the assessments made by other nested functions.
    - **Purpose:** To provide a comprehensive score for the student's paragraph.

* * *

TEACHER BUDDY:
"Source Code"

#main.py

```
import tkinter as tk
from tkinter import ttk
import main_sql as ms
import main_csv as mc

def open_csv_version():
    mc.main_csv()
    print("Opening CSV version")

def open_sql_version():
    ms.main_sql()
    print("Opening SQL version")

root = tk.Tk()
root.title("Teacher Buddy")

style = ttk.Style()
style.configure("TButton", font=("Arial", 12), padding=10)
style.configure("TLabel", font=("Arial", 14, "bold"))

logo_image = tk.PhotoImage(file="/Users/rishiahuja/Desktop/buddy/
                                logo.png")
logo_label = ttk.Label(root, image=logo_image)
logo_label.grid(row=0, column=0, columnspan=2, pady=20)

csv_button = ttk.Button(root, text="CSV Version",
                            command=open_csv_version)
csv_button.grid(row=1, column=0, padx=20, pady=10)

sql_button = ttk.Button(root, text="SQL Version",
                            command=open_sql_version)
sql_button.grid(row=1, column=1, padx=20, pady=10)

instructions_label = ttk.Label(root, text="Select the version
                            you want to use.", style="TLabel")
instructions_label.grid(row=2, column=0, columnspan=2,
                        pady=10)
root.mainloop()
```

#main_sql.py

```
def main_sql():
    import tkinter as tk
    from tkinter import ttk
    import add_student_sql as add
    import view_classroom_sql as vc
    import compare_text as ct
    import score as sc

    def open_add_student_window():
        add.add_student_sql()

    def open_view_classroom_window():
        vc.view_classroom()

    def open_plagiarism_check_window():
        ct.compare_assignments()

    def open_score_assignment_window():
        sc.score()

    root = tk.Tk()
    root.title("Teacher Buddy")

    window_width = 600
    window_height = 400
    screen_width = root.winfo_screenwidth()
    screen_height = root.winfo_screenheight()
    x = (screen_width - window_width) // 2
    y = (screen_height - window_height) // 2
    root.geometry(f"{window_width}x{window_height}+{x}+{y}")

    heading_label = tk.Label(root, text="Teacher Buddy App",
                                    font=("Arial", 24))
    heading_label.pack(pady=20)

    button_frame = tk.Frame(root)
    button_frame.pack()
```

```
style = ttk.Style()
style.configure("TButton", relief=tk.RAISED, padding=(15, 10))

button_width = 40
button_height = 3

add_student_button = ttk.Button(button_frame, text="Add
                                Student", command=
                                open_add_student_window,
                                width=button_width,
                                style="TButton")
add_student_button.grid(row=0, column=0, padx=10, pady=10,
                        sticky="w")

view_classroom_button = ttk.Button(button_frame, text="View
                                   Classroom", command=
                                   open_view_classroom_window,
                                   width=button_width,
                                   style="TButton")
view_classroom_button.grid(row=1, column=0, padx=10, pady=10,
                           sticky="w")

plagiarism_check_button = ttk.Button(button_frame,
                          text="Plagiarism Check", command=
                          open_plagiarism_check_window,
                          width=button_width, style="TButton")
plagiarism_check_button.grid(row=2, column=0, padx=10,
                             pady=10, sticky="w")

score_assignment_button = ttk.Button(button_frame, text="Score
                          Assignment", command=
                          open_score_assignment_window,
                          width=button_width, style="TButton")
score_assignment_button.grid(row=3, column=0, padx=10,
                             pady=10, sticky="w")

root.mainloop()
```

#main_csv.py

```
def main_csv():
    import tkinter as tk
    from tkinter import ttk
    import add_student_csv as add
    import compare_text as ct
    import view_classroom_csv as vc
    import score as sc

    def open_add_student_window():
        add.add_student_csv()

    def open_view_classroom_window():
        vc.view_classroom()

    def open_plagiarism_check_window():
        ct.compare_assignments()

    def open_score_assignment_window():
        sc.score()

    root = tk.Tk()
    root.title("Teacher Buddy")

    window_width = 600
    window_height = 400
    screen_width = root.winfo_screenwidth()
    screen_height = root.winfo_screenheight()
    x = (screen_width - window_width) // 2
    y = (screen_height - window_height) // 2
    root.geometry(f"{window_width}x{window_height}+{x}+{y}")

    heading_label = tk.Label(root, text="Teacher Buddy App",
                             font=("Arial", 24))
    heading_label.pack(pady=20)

    button_frame = tk.Frame(root)
    button_frame.pack()
```

```
style = ttk.Style()
style.configure("TButton", relief=tk.RAISED, padding=(15, 10))

button_width = 40
button_height = 3

add_student_button = ttk.Button(button_frame, text="Add
                                Student", command=
                                open_add_student_window,
                                width=button_width,
                                style="TButton")
add_student_button.grid(row=0, column=0, padx=10, pady=10,
                        sticky="w")

view_classroom_button = ttk.Button(button_frame, text="View
                                   Classroom", command=
                                   open_view_classroom_window,
                                   width=button_width,
                                   style="TButton")
view_classroom_button.grid(row=1, column=0, padx=10, pady=10,
                           sticky="w")

plagiarism_check_button = ttk.Button(button_frame,
                          text="Plagiarism Check", command=
                          open_plagiarism_check_window,
                          width=button_width, style="TButton")
plagiarism_check_button.grid(row=2, column=0, padx=10,
                             pady=10, sticky="w")

score_assignment_button = ttk.Button(button_frame, text="Score
                          Assignment", command=
                          open_score_assignment_window,
                          width=button_width, style="TButton")
score_assignment_button.grid(row=3, column=0, padx=10,
                             pady=10, sticky="w")

root.mainloop()
```

#add_student_sql.py

```
def add_student_sql():
    import mysql.connector as ms
    import tkinter as tk
    from tkinter import filedialog
    from tkinter import messagebox
    from tkinter import ttk

    connect = ms.connect(host = "localhost", user = "root", passwd
                         = “***************”, database= "sca_003")
    cursor = connect.cursor()

    def create_table():
        cursor.execute('CREATE TABLE student(\
            admission_num int NOT NULL,\
            name varchar(20) NOT NULL UNIQUE,\
            roll_num int NOT NULL,\
            class int(2) NOT NULL,\
            section varchar(1) NOT NULL,\
            grade varchar(3) NOT NULL,\
            assignment varchar(max) NOT NULL,\
            email varchar(255) NOT NULL UNIQUE,\
            PRIMARY KEY(admission_num)\
            )')


    student_admission_numbers = []   #admission number
    student_names = []               #name
    student_roll_numbers = []        #roll number
    student_classes = []             #class
    student_sections = []            #section
    student_grades = []              #grade
    student_assignments = []         #assignment path
    student_emails = []              #email
```

```
def add_student():
    try:
        admission_number = int(admission_number_entry.get())
    except ValueError:
        messagebox.showerror("Error", "Admission Number must
                             be a unique integer.")
        return

    name = name_entry.get()
    roll_number = roll_number_entry.get()
    class_name = class_entry.get()
    section = section_entry.get()
    grade = grade_entry.get()
    assignment = file_path_label["text"]
    email = email_entry.get()

    if admission_number in student_admission_numbers:
        messagebox.showerror("Error", "Admission Number must
                             be unique.")
        return

    student_admission_numbers.append(admission_number)
    student_names.append(name)
    student_roll_numbers.append(roll_number)
    student_classes.append(class_name)
    student_sections.append(section)
    student_grades.append(grade)
    student_assignments.append(assignment)
    student_emails.append(email)

    admission_number_entry.delete(0, tk.END)
    name_entry.delete(0, tk.END)
    roll_number_entry.delete(0, tk.END)
    class_entry.delete(0, tk.END)
    section_entry.delete(0, tk.END)
    grade_entry.delete(0, tk.END)
    email_entry.delete(0, tk.END)
    file_path_label.config(text="")
```

```
        messagebox.showinfo("Success", "Student information added
                                successfully.")

    def browse_file():
        file_path = filedialog.askopenfilename()
        file_path_label.config(text=file_path)

    def close_window():
        root.destroy()

    root = tk.Tk()
    root.title("Student Information")

    style = ttk.Style()
    style.configure("TLabel", font=("Arial", 12))
    style.configure("TButton", font=("Arial", 12))
    style.configure("TEntry", font=("Arial", 12))

    fields = [
        ("Admission Number:", "admission_number_entry"),
        ("Name:", "name_entry"),
        ("Roll Number:", "roll_number_entry"),
        ("Class:", "class_entry"),
        ("Section:", "section_entry"),
        ("Grade:", "grade_entry"),
        ("Upload Assignment:", "file_path_label"),
        ("Email:", "email_entry"),
    ]

    for i, (label_text, widget_name) in enumerate(fields):
        label = ttk.Label(root, text=label_text)
        label.grid(row=i, column=0, padx=10, pady=5, sticky="e")

        if widget_name == "file_path_label":
            widget = ttk.Label(root, text="")
            browse_button = ttk.Button(root, text="Browse",
                                       command=browse_file)
            widget.grid(row=i, column=1, padx=10, pady=5,
                        sticky="w")
            browse_button.grid(row=i, column=2, padx=10, pady=5)
```

```
    else:
        widget = ttk.Entry(root)
        widget.grid(row=i, column=1, columnspan=2, padx=10,
                    pady=5)

    globals()[widget_name] = widget

submit_button = ttk.Button(root, text="Add Student",
                           command=add_student)
submit_button.grid(row=len(fields), column=0, columnspan=3,
                   padx=10, pady=10)

close_button = ttk.Button(root, text="Close",
                          command=close_window)
close_button.grid(row=len(fields) + 1, column=0, columnspan=3,
                  padx=10, pady=10)

root.mainloop()


# x = student_admission_numbers
def insert(x):
    for i in range(0, len(x)):
        cursor.execute('INSERT INTO student VALUES (%s, %s,
                       %s, %s, %s, %s, %s, %s)',
            (student_admission_numbers[i], student_names[i],
             student_roll_numbers[i], student_classes[i],
             student_sections[i], student_grades[i],
             student_assignments[i], student_emails[i]))

        connect.commit()


insert(student_admission_numbers)
```

#add_student_sql.py

```
def add_student_csv():
    import csv
    import tkinter as tk
    from tkinter import filedialog
    from tkinter import messagebox
    from tkinter import ttk

    student_data_csv = "students.csv"

    student_admission_numbers = []   # Admission number
    student_names = []               # Name
    student_roll_numbers = []        # Roll number
    student_classes = []             # Class
    student_sections = []            # Section
    student_grades = []              # Grade
    student_assignments = []         # Assignment path
    student_emails = []              # Email


    def add_student():
        try:
            admission_number = int(admission_number_entry.get())
        except ValueError:
            messagebox.showerror("Error", "Admission Number must
                                 be a unique integer.")
            return

        name = name_entry.get()
        roll_number = roll_number_entry.get()
        class_name = class_entry.get()
        section = section_entry.get()
        grade = grade_entry.get()
        assignment = file_path_label["text"] or "missing"
        email = email_entry.get() or "missing"

        if admission_number in student_admission_numbers:
            messagebox.showerror("Error", "Admission Number must
                                 be unique.")
            return
```

```
    student_admission_numbers.append(admission_number)
    student_names.append(name)
    student_roll_numbers.append(roll_number)
    student_classes.append(class_name)
    student_sections.append(section)
    student_grades.append(grade)
    student_assignments.append(assignment)
    student_emails.append(email)

    admission_number_entry.delete(0, tk.END)
    name_entry.delete(0, tk.END)
    roll_number_entry.delete(0, tk.END)
    class_entry.delete(0, tk.END)
    section_entry.delete(0, tk.END)
    grade_entry.delete(0, tk.END)
    email_entry.delete(0, tk.END)
    file_path_label.config(text="")

    messagebox.showinfo("Success", "Student information added
                        successfully.")


def browse_file():
    file_path = filedialog.askopenfilename()
    file_path_label.config(text=file_path)


def save_to_csv():
    with open(student_data_csv, mode='a', newline='') as file:
        writer = csv.writer(file)
        for i in range(len(student_admission_numbers)):
            writer.writerow([
                student_admission_numbers[i],
                student_names[i],
                student_roll_numbers[i],
                student_classes[i],
                student_sections[i],
                student_grades[i],
                student_assignments[i],
                student_emails[i]
            ])
```

```
        messagebox.showinfo("Success", f"Data saved to
                            {student_data_csv} successfully.")


def close_window():
    root.destroy()


root = tk.Tk()
root.title("Student Information")

style = ttk.Style()
style.configure("TLabel", font=("Arial", 12))
style.configure("TButton", font=("Arial", 12))
style.configure("TEntry", font=("Arial", 12))

fields = [
    ("Admission Number:", "admission_number_entry"),
    ("Name:", "name_entry"),
    ("Roll Number:", "roll_number_entry"),
    ("Class:", "class_entry"),
    ("Section:", "section_entry"),
    ("Grade:", "grade_entry"),
    ("Upload Assignment:", "file_path_label"),
    ("Email:", "email_entry"),
]

for i, (label_text, widget_name) in enumerate(fields):
    label = ttk.Label(root, text=label_text)
    label.grid(row=i, column=0, padx=10, pady=5, sticky="e")

    if widget_name == "file_path_label":
        widget = ttk.Label(root, text="")
        browse_button = ttk.Button(root, text="Browse",
                                   command=browse_file)
        widget.grid(row=i, column=1, padx=10, pady=5,
                    sticky="w")
```

```
            browse_button.grid(row=i, column=2, padx=10, pady=5)
        else:

            widget = ttk.Entry(root)
            widget.grid(row=i, column=1, columnspan=2, padx=10,
                        pady=5)

        globals()[widget_name] = widget

    submit_button = ttk.Button(root, text="Add Student",
                               command=add_student)
    submit_button.grid(row=len(fields), column=0, columnspan=3,
                       padx=10, pady=10)

    save_csv_button = ttk.Button(root, text="Save to CSV",
                                 command=save_to_csv)
    save_csv_button.grid(row=len(fields) + 1, column=0,
                         columnspan=3, padx=10, pady=10)

    close_button = ttk.Button(root, text="Close",
                              command=close_window)
    close_button.grid(row=len(fields) + 2, column=0, columnspan=3,
                      padx=10, pady=10)

    root.mainloop()
```

#view_classroom_sql.py

```
def view_classroom():
    import csv
    import tkinter as tk
    from tkinter import ttk

    student_data_csv = "students.csv"

    def fetch_students(order_by="roll_num", filters=None):
        with open(student_data_csv, mode='r') as file:
            reader = csv.reader(file)
            students = [row for row in reader]

        if filters:
            students = [student for student in students if
            all(filter_func(student) for filter_func in filters)]

        order_by_index = {"admission_num": 0, "name": 1,
                          "roll_num": 2, "class": 3, "section": 4,
                          "grade": 5, "assignment": 6, "email": 7}
        students.sort(key=lambda x: x[order_by_index.get(order_by, 2)])
        return students

    def display_classroom():
        def update_display():
            order_by_value = order_by_var.get()
            grade_filter_value = grade_filter_var.get()
            class_filter_value = class_filter_var.get()
            assignment_filter_value = assignment_filter_var.get()

            filters = []
            if grade_filter_value:
                filters.append(lambda student: student[5] ==
                                               grade_filter_value)
            if class_filter_value:
                filters.append(lambda student: student[3] ==
                                               class_filter_value)
```

```
    if assignment_filter_value:
        if assignment_filter_value == 'Not Missing':
            filters.append(lambda student: student[6]!='')
        else:
            filters.append(lambda student: student[6] == '')

    tree.delete(*tree.get_children())
    students = fetch_students(order_by=order_by_value,
                              filters=filters)
    for student in students:
        tree.insert("", "end", values=student)

root = tk.Tk()
root.title("Classroom Information")

style = ttk.Style()
style.configure("TLabel", font=("Arial", 12), padding=5)
style.configure("TButton", font=("Arial", 12), padding=5)
style.configure("TEntry", font=("Arial", 12), padding=5)
style.configure("Treeview.Heading", font=("Arial", 12, "bold"))
style.configure("Treeview", font=("Arial", 12),
                rowheight=25, padding=5)

order_by_var = tk.StringVar()
order_by_var.set("roll_num")
order_by_menu = ttk.Combobox(root,
                             textvariable=order_by_var,
                             values=("admission_num",
                             "name", "roll_num", "class",
                             "section", "grade",
                             "assignment", "email"))
order_by_menu.grid(row=0, column=1, padx=10, pady=10,
                   sticky="w")
order_by_menu.bind("<<ComboboxSelected>>", lambda event:
                   update_display())

grade_filter_var = tk.StringVar()
```

```
grade_filter_menu = ttk.Combobox(root,
                    textvariable=grade_filter_var,
                    values=("A", "B", "C", "D", "F", ""))
grade_filter_menu.set("")
grade_filter_menu.grid(row=0, column=3, padx=10, pady=10,
                       sticky="w")
grade_filter_menu.bind("<<ComboboxSelected>>", lambda
                       event: update_display())

class_filter_var = tk.StringVar()
class_filter_menu = ttk.Combobox(root,
                    textvariable=class_filter_var,
                    values=("1", "2", "3", "4", "5", "6",
                    "7", "8", "9", "10", "11", "12", ""))
class_filter_menu.set("")
class_filter_menu.grid(row=0, column=5, padx=10, pady=10,
                       sticky="w")
class_filter_menu.bind("<<ComboboxSelected>>", lambda
                       event: update_display())

assignment_filter_var = tk.StringVar()
assignment_filter_menu = ttk.Combobox(root,
                         textvariable=assignment_filter_var,
                         values=("Missing", "Not Missing", ""))
assignment_filter_menu.set("")
assignment_filter_menu.grid(row=0, column=7, padx=10,
                            pady=10, sticky="w")
assignment_filter_menu.bind("<<ComboboxSelected>>", lambda
                            event: update_display())

tree = ttk.Treeview(root, columns=(1, 2, 3, 4, 5, 6, 7,
                    8), show="headings", height=10)
tree.grid(row=1, column=0, columnspan=8, padx=10, pady=10,
          sticky="nsew")

tree.heading(1, text="Admission Number")
tree.heading(2, text="Name")
tree.heading(3, text="Roll Number")
tree.heading(4, text="Class")
tree.heading(5, text="Section")
tree.heading(6, text="Grade")
```

```
        tree.heading(7, text="Assignment Path")
        tree.heading(8, text="Email")

        for student in fetch_students(order_by=order_by_var.get()):
            tree.insert("", "end", values=student)

        for col in (1, 2, 3, 4, 5, 6, 7, 8):
            tree.column(col, anchor="center")

        scrollbar = ttk.Scrollbar(root, orient="vertical",
                                  command=tree.yview)
        scrollbar.grid(row=1, column=8, sticky="ns")
        tree.configure(yscrollcommand=scrollbar.set)

        hscrollbar = ttk.Scrollbar(root, orient="horizontal",
                                   command=tree.xview)
        hscrollbar.grid(row=2, column=0, columnspan=8,
                        sticky="ew")
        tree.configure(xscrollcommand=hscrollbar.set)

        ttk.Label(root, text="Order By:").grid(row=0, column=0,
                  padx=10, pady=10, sticky="e")
        ttk.Label(root, text="Grade Filter:").grid(row=0,
                  column=2, padx=10, pady=10, sticky="e")
        ttk.Label(root, text="Class Filter:").grid(row=0,
                  column=4, padx=10, pady=10, sticky="e")
        ttk.Label(root, text="Assignment Filter:").grid(row=0,
                  column=6, padx=10, pady=10, sticky="e")

        clear_filters_button = ttk.Button(root, text="Clear
                                 Filters", command=update_display)
        clear_filters_button.grid(row=3, column=0, columnspan=8,
                                  pady=10)

        close_button = ttk.Button(root, text="Close",
                                  command=root.destroy)
        close_button.grid(row=4, column=0, columnspan=8, pady=10)

        root.mainloop()

    display_classroom()
```

#view_classroom_csv.py

```
def view_classroom():
    import csv
    import tkinter as tk
    from tkinter import ttk

    student_data_csv = "students.csv"

    def fetch_students(order_by="roll_num", filters=None):
        with open(student_data_csv, mode='r') as file:
            reader = csv.reader(file)
            students = [row for row in reader]

        if filters:
            students = [student for student in students if
            all(filter_func(student) for filter_func in filters)]

        order_by_index = {"admission_num": 0, "name": 1,
                          "roll_num": 2, "class": 3, "section": 4,
                          "grade": 5, "assignment": 6, "email": 7}
        students.sort(key=lambda x: x[order_by_index.get(order_by, 2)])
        return students

    def display_classroom():
        def update_display():
            order_by_value = order_by_var.get()
            grade_filter_value = grade_filter_var.get()
            class_filter_value = class_filter_var.get()
            assignment_filter_value = assignment_filter_var.get()

            filters = []

            if grade_filter_value:
                filters.append(lambda student: student[5] ==
                grade_filter_value)

            if class_filter_value:
                filters.append(lambda student: student[3] ==
                class_filter_value)
```

```
    if assignment_filter_value:
        if assignment_filter_value == 'Not Missing':
            filters.append(lambda student: student[6] != '')
        else:
            filters.append(lambda student: student[6] == '')

    tree.delete(*tree.get_children())
    students = fetch_students(order_by=order_by_value,
                              filters=filters)
    for student in students:
        tree.insert("", "end", values=student)

root = tk.Tk()
root.title("Classroom Information")

style = ttk.Style()
style.configure("TLabel", font=("Arial", 12), padding=5)
style.configure("TButton", font=("Arial", 12), padding=5)
style.configure("TEntry", font=("Arial", 12), padding=5)
style.configure("Treeview.Heading", font=("Arial", 12,
                "bold"))
style.configure("Treeview", font=("Arial", 12),
                rowheight=25, padding=5)

order_by_var = tk.StringVar()
order_by_var.set("roll_num")
order_by_menu = ttk.Combobox(root,
                textvariable=order_by_var,
                values=("admission_num", “name",
                        "roll_num", "class", “section",
                        "grade", "assignment", "email"))
order_by_menu.grid(row=0, column=1, padx=10, pady=10,
                   sticky="w")
order_by_menu.bind("<<ComboboxSelected>>", lambda event:
                   update_display())

grade_filter_var = tk.StringVar()
grade_filter_menu = ttk.Combobox(root,
                   textvariable=grade_filter_var,
                   values=("A", "B", "C", "D", "F", ""))
```

```
grade_filter_menu.set("")
grade_filter_menu.grid(row=0, column=3, padx=10, pady=10,
                       sticky="w")
grade_filter_menu.bind("<<ComboboxSelected>>", λ
                       event: update_display())

class_filter_var = tk.StringVar()
class_filter_menu = ttk.Combobox(root,
                    textvariable=class_filter_var,
                    values=("1", "2", "3", "4", "5", “6",
                    "7", "8", "9", "10", "11", "12", ""))
class_filter_menu.set("")
class_filter_menu.grid(row=0, column=5, padx=10, pady=10,
                       sticky="w")
class_filter_menu.bind("<<ComboboxSelected>>", λ
                        event: update_display())

assignment_filter_var = tk.StringVar()
assignment_filter_menu = ttk.Combobox(root,
                       textvariable=assignment_filter_var,
                       values=("Missing", "Not Missing”, ""))
assignment_filter_menu.set("")
assignment_filter_menu.grid(row=0, column=7, padx=10,
                            pady=10, sticky="w")
assignment_filter_menu.bind("<<ComboboxSelected>>", λ
                             event: update_display())

tree = ttk.Treeview(root, columns=(1, 2, 3, 4, 5, 6, 7, 8),
                           show="headings", height=10)
tree.grid(row=1, column=0, columnspan=8, padx=10, pady=10,
          sticky="nsew")

tree.heading(1, text="Admission Number")
tree.heading(2, text="Name")
tree.heading(3, text="Roll Number")
tree.heading(4, text="Class")
tree.heading(5, text="Section")
tree.heading(6, text="Grade")
tree.heading(7, text="Assignment Path")
tree.heading(8, text="Email")
```

```
    for student in
                fetch_students(order_by=order_by_var.get()):
        tree.insert("", "end", values=student)

    for col in (1, 2, 3, 4, 5, 6, 7, 8):
        tree.column(col, anchor="center")

    scrollbar = ttk.Scrollbar(root, orient="vertical",
                              command=tree.yview)
    scrollbar.grid(row=1, column=8, sticky="ns")
    tree.configure(yscrollcommand=scrollbar.set)

    hscrollbar = ttk.Scrollbar(root, orient="horizontal",
                               command=tree.xview)
    hscrollbar.grid(row=2, column=0, columnspan=8,
                    sticky="ew")
    tree.configure(xscrollcommand=hscrollbar.set)
    ttk.Label(root, text="Order By:").grid(row=0, column=0,
              padx=10, pady=10, sticky="e")
    ttk.Label(root, text="Grade Filter:").grid(row=0,
              column=2, padx=10, pady=10, sticky="e")
    ttk.Label(root, text="Class Filter:").grid(row=0,
              column=4, padx=10, pady=10, sticky="e")
    ttk.Label(root, text="Assignment Filter:").grid(row=0,
              column=6, padx=10, pady=10, sticky="e")

    clear_filters_button = ttk.Button(root, text="Clear
                           Filters", command=update_display)
    clear_filters_button.grid(row=3, column=0, columnspan=8,
                              pady=10)

    close_button = ttk.Button(root, text="Close",
                              command=root.destroy)
    close_button.grid(row=4, column=0, columnspan=8, pady=10)

    root.mainloop()

display_classroom()
```

Compare Texts

```
def compare_assignments():
    import tkinter as tk
    from tkinter import filedialog
    from tkinter import ttk

    def load_file_or_display_contents(entry, text_widget, name_var):
        file_path = entry.get()

        if not file_path:
            file_path = filedialog.askopenfilename()

        if file_path:
            with open(file_path, 'r') as file:
                file_contents = file.read()
                text_widget.delete("1.0", tk.END)
                text_widget.insert(tk.END, file_contents)
                name_var.set(file_contents)

    def compare_text():
        text1 = text_textbox1.get("1.0", tk.END)
        text2 = text_textbox2.get("1.0", tk.END)
        words1 = set(text1.split())
        words2 = set(text2.split())
        common_words = words1.intersection(words2)

        similarity_percentage = len(common_words)/len(words1)*100

        highlight_text(text_textbox1, common_words, "red", "bold")
        highlight_text(text_textbox2, common_words, "red", "bold")

        progress_bar['value'] = similarity_percentage

    def reset_text():
        text_textbox1.delete("1.0", tk.END)
        text_textbox2.delete("1.0", tk.END)
        progress_bar['value'] = 0
        name_var1.set("")
        name_var2.set("")
```

```
def close_app():
    root.destroy()

def highlight_text(text_widget, words, color, style):
    text_widget.tag_configure("highlight", foreground=color,
                              font=("Helvetica", 10, style))

    for word in words:
        start = "1.0"
        while True:
            start = text_widget.search(word, start,
                                       stopindex=tk.END)
            if not start:
                break

            end = f"{start}+{len(word)}c"
            text_widget.tag_add("highlight", start, end)
            start = end

root = tk.Tk()
root.title("Text Comparison Tool")

frame = tk.Frame(root)
frame.pack(padx=10, pady=10)

text_label1 = tk.Label(frame, text="Student 1 Name:")
text_label1.grid(row=0, column=0, padx=5, pady=5)
name_var1 = tk.StringVar()
name_entry1 = tk.Entry(frame, width=20,
                       textvariable=name_var1)
name_entry1.grid(row=0, column=1, padx=5, pady=5)
text_textbox1 = tk.Text(frame, wrap=tk.WORD, width=30,
                        height=10)
text_textbox1.grid(row=0, column=2, padx=5, pady=5)

text_label2 = tk.Label(frame, text="Student 2 Name:")
text_label2.grid(row=1, column=0, padx=5, pady=5)
name_var2 = tk.StringVar()
```

```
name_entry2 = tk.Entry(frame, width=20,
                       textvariable=name_var2)
name_entry2.grid(row=1, column=1, padx=5, pady=5)
text_textbox2 = tk.Text(frame, wrap=tk.WORD, width=30,
                        height=10)
text_textbox2.grid(row=1, column=2, padx=5, pady=5)

load_button1 = tk.Button(frame, text="Load Text 1",
               command=lambda:
               load_file_or_display_contents(name_entry1,
                                             text_textbox1, name_var1))
load_button1.grid(row=0, column=3, padx=5, pady=5)
load_button2 = tk.Button(frame, text="Load Text 2",
               command=lambda:
               load_file_or_display_contents(name_entry2,
                                             text_textbox2, name_var2))
load_button2.grid(row=1, column=3, padx=5, pady=5)

compare_button = tk.Button(root, text="Compare",
                           command=compare_text)
compare_button.pack(pady=5)
reset_button = tk.Button(root, text="Reset",
                         command=reset_text)
reset_button.pack(pady=5)
close_button = tk.Button(root, text="Close",
                         command=close_app)
close_button.pack(pady=5)

progress_frame = tk.Frame(root)
progress_frame.pack(pady=10)
progress_label = tk.Label(progress_frame, text="Similarity:")
progress_label.pack(side=tk.LEFT, padx=5)
progress_bar = ttk.Progressbar(progress_frame, orient=
               "horizontal", length=200, mode="determinate")
progress_bar.pack(side=tk.LEFT, padx=5)

root.mainloop()
```

Score_Assignments

```
def score():
    import tkinter as tk
    from tkinter import filedialog, messagebox, ttk
    import re
    from nltk.tokenize import word_tokenize
    from nltk.corpus import stopwords
    import nltk
    import time
    from io import StringIO
    import sys

    nltk.download('stopwords')
    nltk.download('averaged_perceptron_tagger')

    def browse_file():
        file_path = filedialog.askopenfilename(filetypes=[("Text
                                                files", "*.txt")])
        if file_path:
            file_entry.configure(state="normal")
            file_entry.delete(0, tk.END)
            file_entry.insert(0, file_path)
            file_entry.configure(state="readonly")

    def create_checkboxes():
        rules = [
            "Minimum 150 words should be used. The limit is capped
             at 200 words.",
            "Stop words will not be counted as words.",
            "The paragraph should always begin with the phrase
             'Once upon a time.'",
        ]

        for idx, rule in enumerate(rules, start=1):
            var = tk.BooleanVar()
            checkbox = ttk.Checkbutton(root, text=rule,
                                       variable=var)
            checkbox.grid(row=idx, column=0, columnspan=3,
                          padx=10, pady=5, sticky="w")
```

```
            checkboxes[rule] = var

    def check_paragraph():
        paragraph_path = file_entry.get()

        try:
            sys.stdout = StringIO()

            execute_paragraph_checker(paragraph_path)

            output = sys.stdout.getvalue()

            messagebox.showinfo("Output", output)

        finally:
            sys.stdout = sys.__stdout__

    def execute_paragraph_checker(paragraph_path):
        with open('tags.txt', 'r') as file:
            l = file.read()
        lines = l.split('\n')

        user = []
        score = 1

        with open(paragraph_path, 'r') as file:
            paragraph = file.read()

        def StopWords(x):
            nonlocal user
            x = re.sub(r"[^a-zA-Z0-9]", " ", x.lower())
            token = word_tokenize(x)
            for j in range(0, len(token)):
                match = 'rishi'
                for z in range(0, len(stop_words)):
                    if token[j] == stop_words[z]:
                        print("it was a stop word")
                        match = 'nitin'
                        break
                if match == 'rishi':
                    user.append(token[j])

        def WordCount(y):
```

```
        nonlocal score
        length = len(y)
        if length < 150:
            print('The length of the paragraph is less than
                   150.\n +1 denied.')
        elif length >= 150:
            score = score + 1
            print('The length of the paragraph is greater than
                   150 words.\n +1 awarded')
        else:
            print('You exceeded the word limit of the
                   paragraph.\n +1 denied')

    def Regular(o):
        nonlocal score
        if re.search('^Once upon a time', o).group():
            score = score + 1
            print('Correct starting phrase.\n +1 awarded')
        else:
            print('Did not start with the required phrase.\n
                   +1 denied.')

    def types(v):
        tags = {}
        for e in range(0, len(lines)):
            line = lines[e]
            a = line.split(':')
            tags[a[0]] = a[1]

        for words in v:
            j = (nltk.pos_tag([words]))
            p = j[0][1]
            ok = list(j[0])
            if p in tags:
                ok[1] = tags[p]
                print(ok)

        fd = nltk.FreqDist(user)
        print(fd)
        fd.plot(20)


    def calculate():
```

```
        print('calculating marks.....')
        time.sleep(1.8)
        print('marks = ', score)

    StopWords(paragraph)
    WordCount(user)
    Regular(paragraph)
    types(user)
    calculate()
    score_value_label.config(text=f"{score}/3")

root = tk.Tk()
root.title("Paragraph Checker")
root.geometry("600x400")

stop_words = stopwords.words('english')
checkboxes = {}

file_label = ttk.Label(root, text="Upload File:")
file_label.grid(row=0, column=0, padx=10, pady=10, sticky="e")

file_entry = ttk.Entry(root, width=40, state="readonly")
file_entry.grid(row=0, column=1, padx=10, pady=10, sticky="w")

browse_button = ttk.Button(root, text="Browse",
                           command=browse_file)
browse_button.grid(row=0, column=2, padx=10, pady=10)

create_checkboxes()

score_label = ttk.Label(root, text="Final Score:")
score_label.grid(row=6, column=0, columnspan=2, padx=10,
                 pady=10, sticky="e")

score_value_label = ttk.Label(root, text="")
score_value_label.grid(row=6, column=2, padx=10, pady=10,
                       sticky="w")

check_button = ttk.Button(root, text="Check Paragraph",
                          command=check_paragraph)
check_button.grid(row=7, column=0, columnspan=3, pady=10)

root.mainloop()
```

TEACHER BUDDY

Execution Walkthrough

Welcome to Teacher Buddy! On the main screen, you have two options:

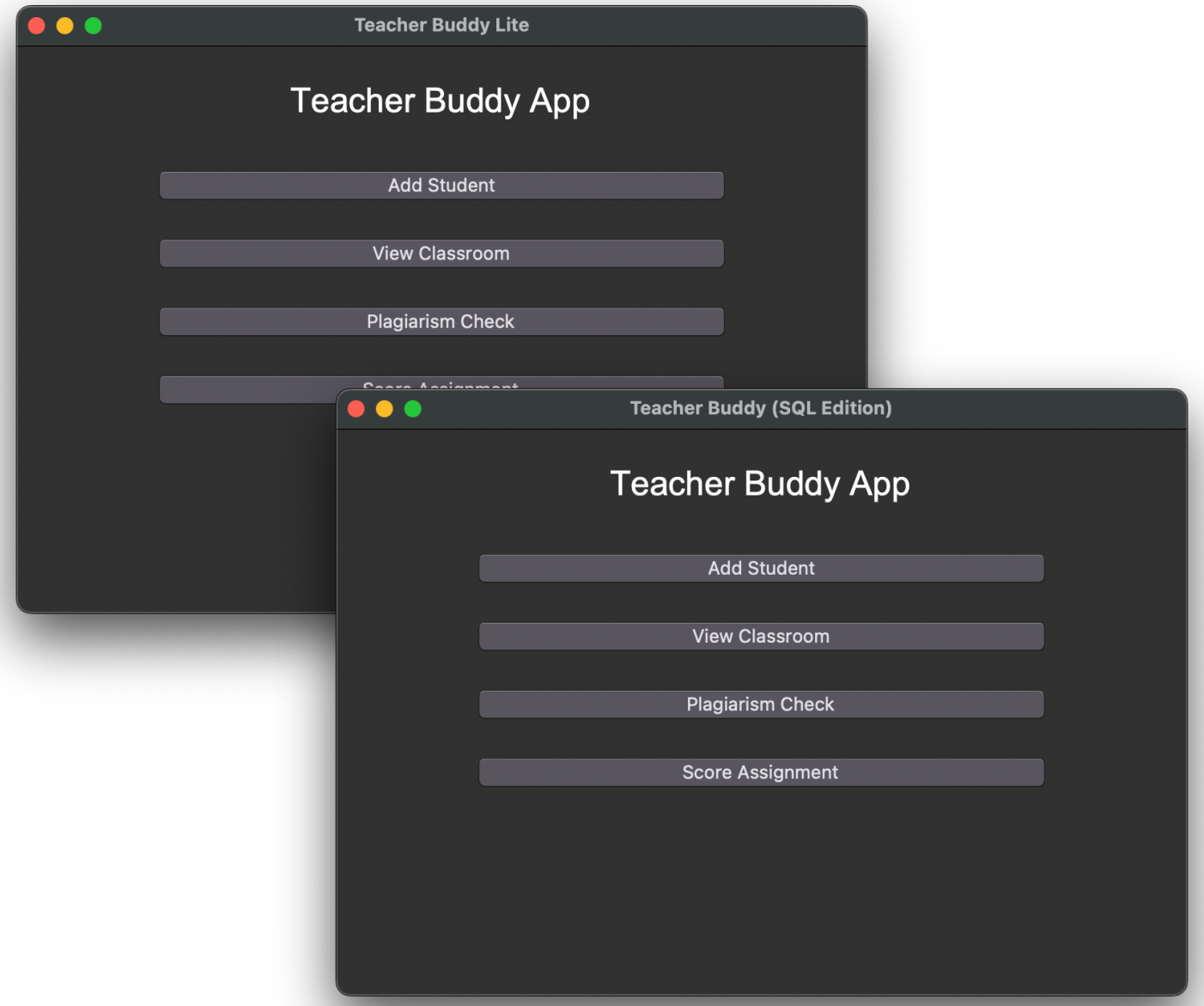
1. Access Teacher Buddy Lite (CSV Version)
2. Access Teacher Buddy (SQL Version)

Choose the option that best suits your needs.

On the subsequent page, you may find two distinct outputs, in accordance with the version you selected* (CSV or SQL):

For* TEACHER BUDDY/TEACHER BUDDY *Lite :

- Add Students
- View Classroom
- Check Assignment for Plagiarism
- Score Assignment

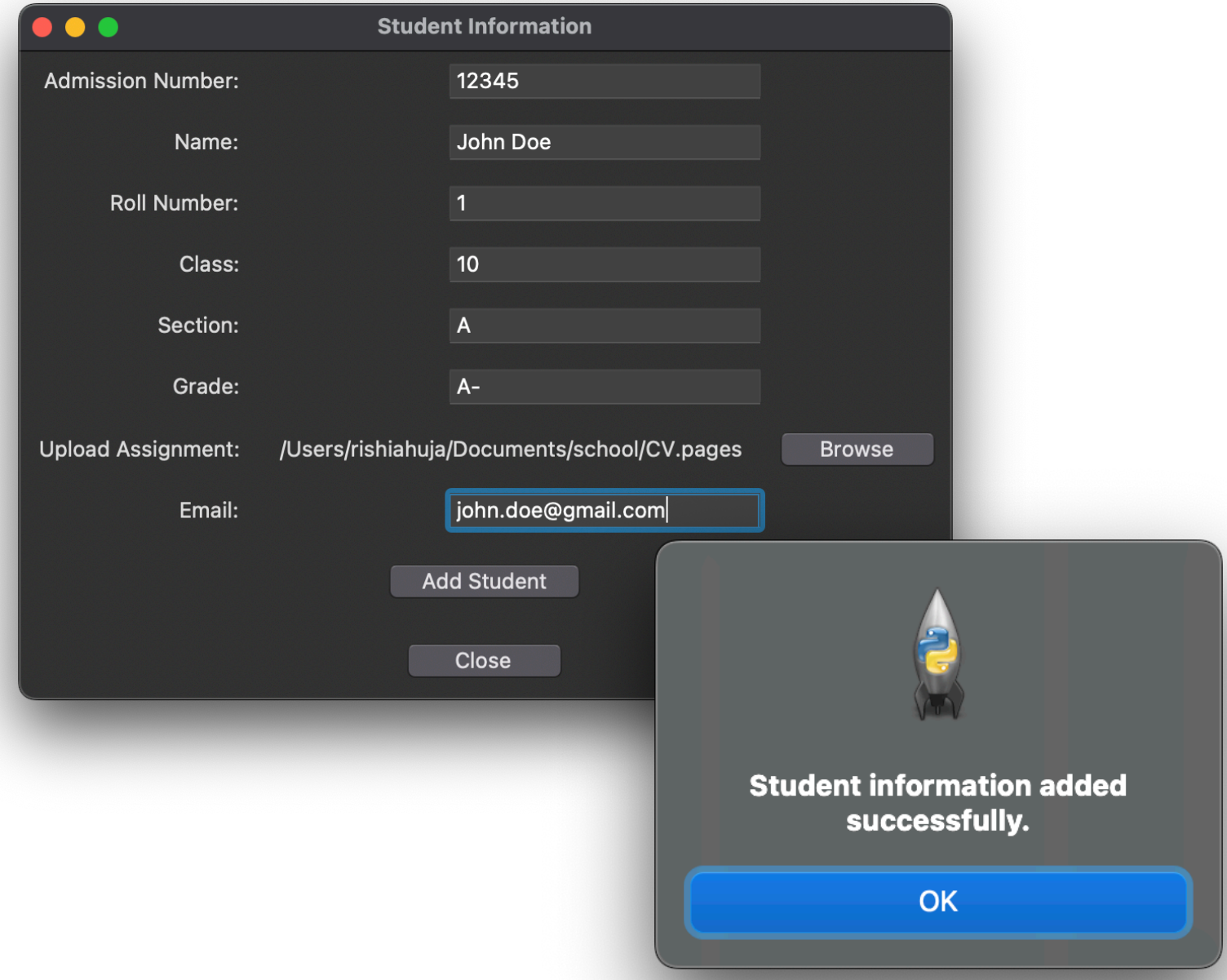


**The functionality remains consistent, with the only difference being the underlying data storage format.*

On the third screen, the focus is on adding students to the system. Please provide all necessary details for each student.

TEACHER BUDDY :

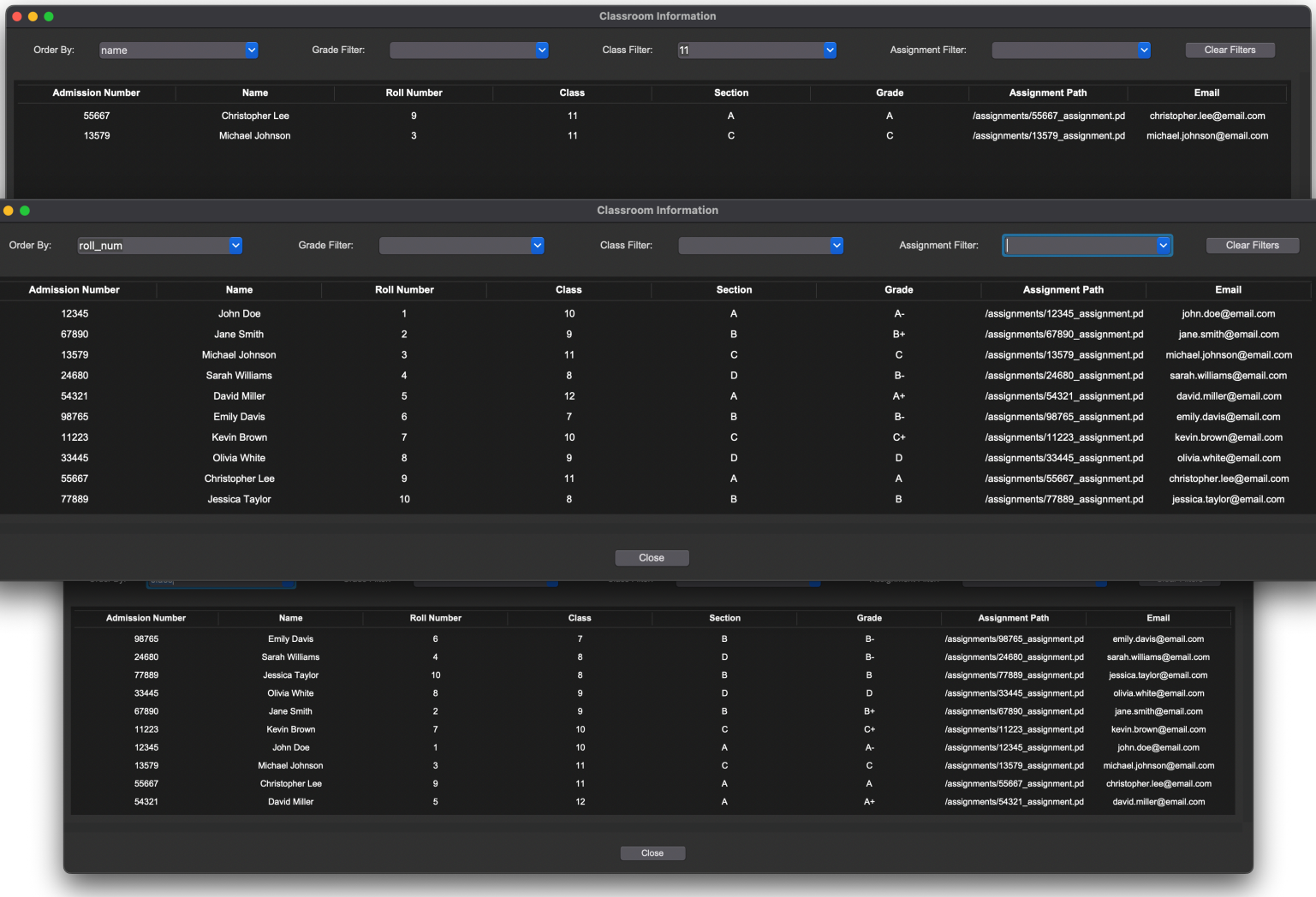
- Ensures admission numbers are unique
- Custom-tailored to enforce uniqueness
- Validates entered details for accuracy
- Saves the student information
- Stores assignmentsfile path to CSV/SQL for later use



Classroom Overview:

In the Classroom section, you can explore and manage your students effortlessly. Here, you'll find all the details you added in the "Add Students" screen. Take a look at the various features designed to make classroom management a breeze:

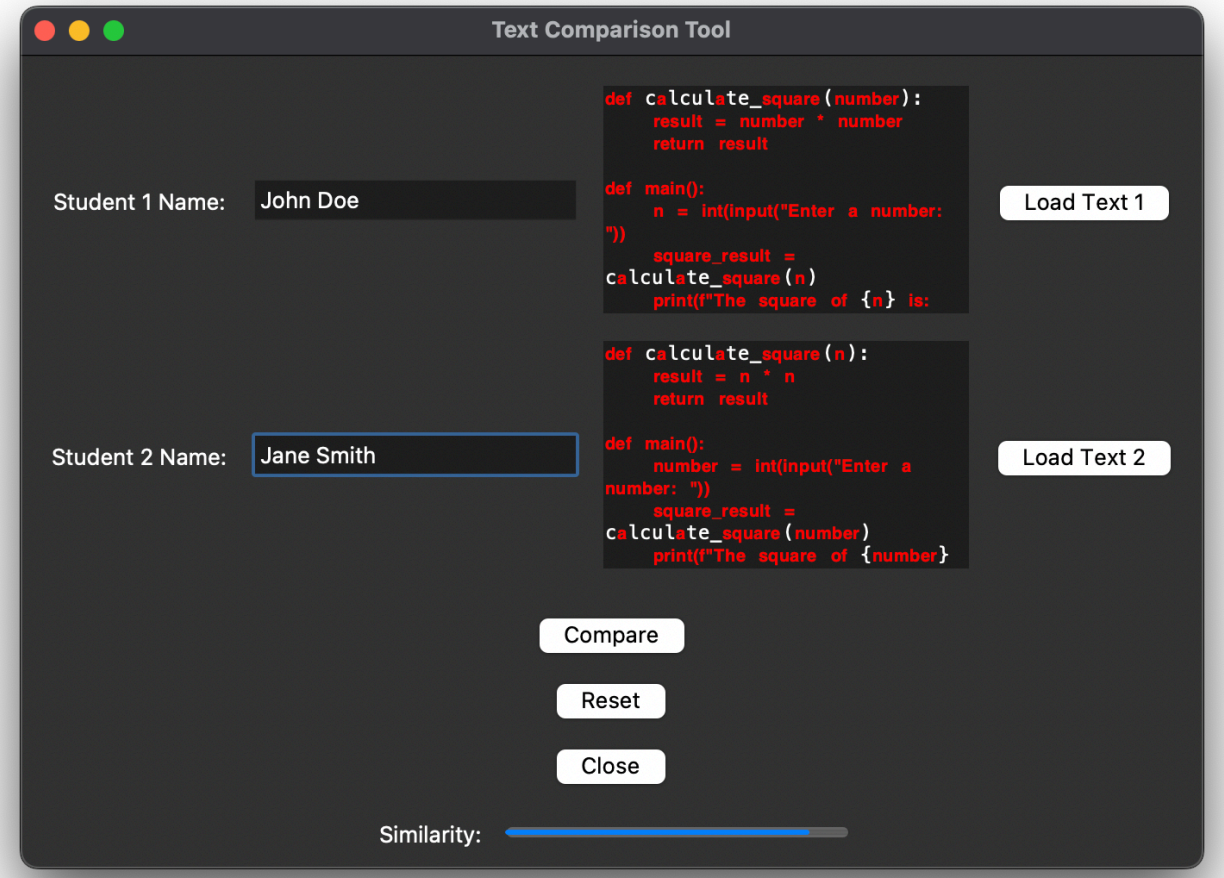
- Navigate to your classroom to review all students and their details from the previous "Add Students" screen.
- Arrange student data based on preferences, including options like roll number, admission number, and name.
- Apply filters for focused viewing, such as displaying only Class 10 students or sorting by names.
- Enjoy flexibility in customizing your view of the classroom according to your needs.
- Conveniently clear all applied filters with a single button click



Check for plagiarism:

Explore the Plagiarism Checker screen to effortlessly evaluate assignment similarity between any two students. By entering their names, load assignments and discover shared content with identical words highlighted in red. This feature ensures fair assessment and upholds academic integrity.

- Efficiently compare assignments of two students.
- Visual indication with red-colored highlighting for identical words.
- Promotes fair evaluation and encourages academic honesty.

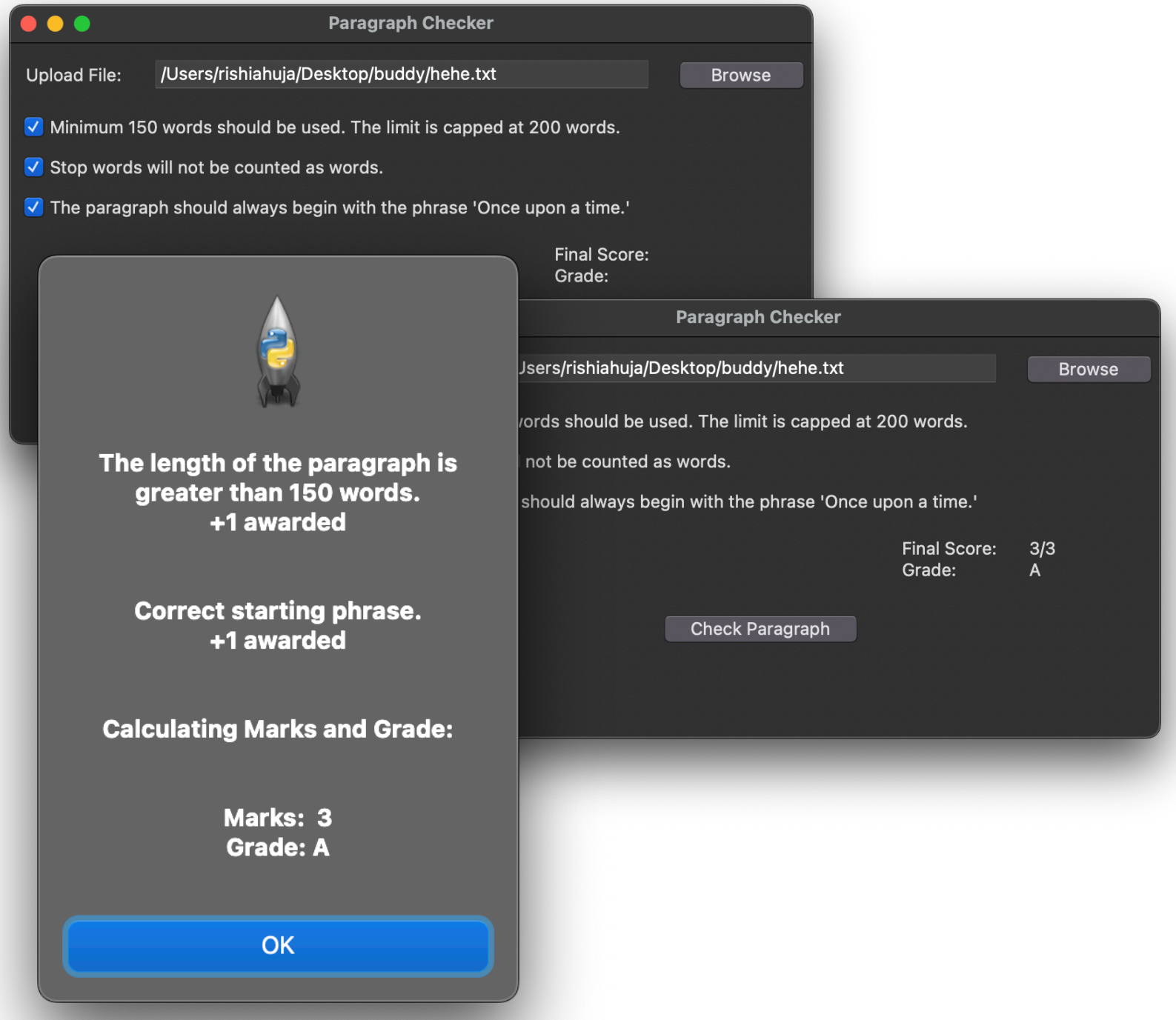


**The displayed output demonstrates two Python programs that are over 50% similar, for illustrative purposes only. Actual results may vary.*

Assignment Grading Tool:

Effortlessly evaluate student assignments with the Grade Assignment tool. Designed to simplify the evaluation process. Load a student's assignment and benefit from the following features:

- Automatically grade a student's work based on preset rules.
- Choose which rules to apply or ignore, tailoring the grading process.
- Receive an automated final score and corresponding grade.
- Explore calculated counts of nouns, pronouns, and other linguistic elements and gain insights into the frequency of each word, enhancing the evaluation process.



◉ SQL Table student :

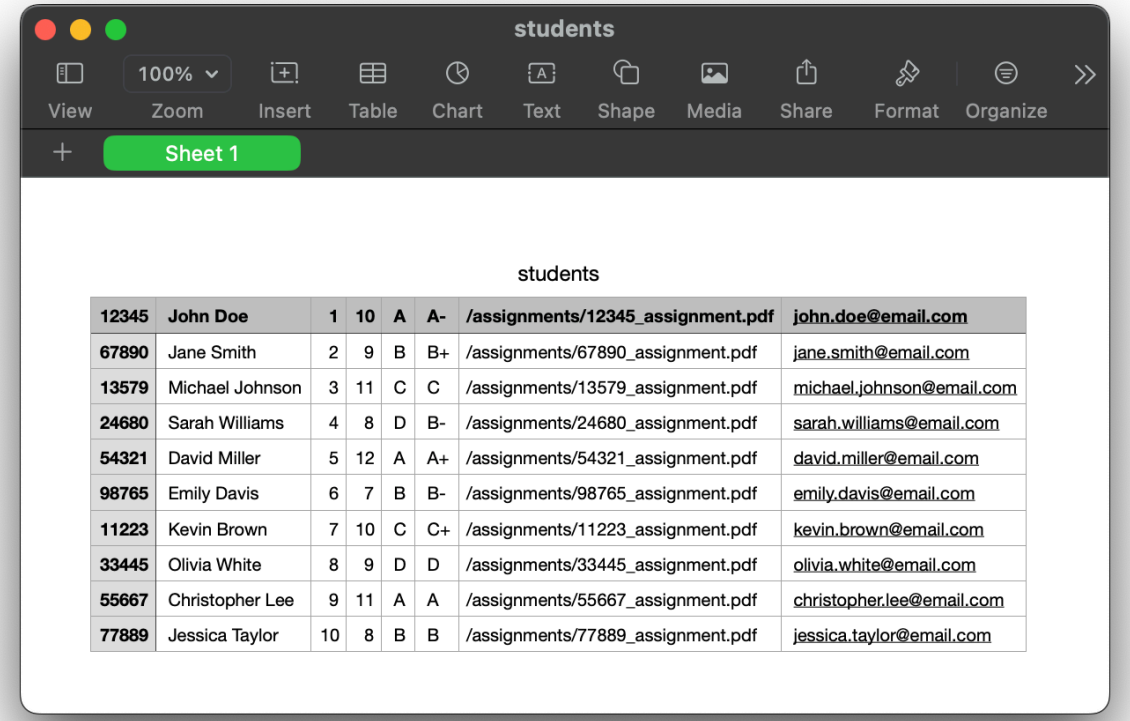
rishiahuja — mysql -u root -p — 137×24

```
mysql> use sca_003;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from student;
+---------------+-----------------+----------+-------+---------+-------+--------------------------------+-------------------------+
| admission_num | name            | roll_num | class | section | grade | assignment                     | email                   |
+---------------+-----------------+----------+-------+---------+-------+--------------------------------+-------------------------+
|         11223 | Kevin Brown     |        7 |    10 | C       | C+    | /assignments/11223_assignment.pdf | kevin.brown@email.com     |
|         12345 | John Doe        |        1 |    10 | A       | A-    | /assignments/12345_assignment.pdf | john.doe@email.com        |
|         13579 | Michael Johnson |        3 |    11 | C       | C     | /assignments/13579_assignment.pdf | michael.johnson@email.com |
|         24680 | Sarah Williams  |        4 |     8 | D       | B-    | /assignments/24680_assignment.pdf | sarah.williams@email.com  |
|         33445 | Olivia White    |        8 |     9 | D       | D     | /assignments/33445_assignment.pdf | olivia.white@email.com    |
|         54321 | David Miller    |        5 |    12 | A       | A+    | /assignments/54321_assignment.pdf | david.miller@email.com    |
|         55667 | Christopher Lee |        9 |    11 | A       | A     | /assignments/55667_assignment.pdf | christopher.lee@email.com |
|         67890 | Jane Smith      |        2 |     9 | B       | B+    | /assignments/67890_assignment.pdf | jane.smith@email.com      |
|         77889 | Jessica Taylor  |       10 |     8 | B       | B     | /assignments/77889_assignment.pdf | jessica.taylor@email.com  |
|         98765 | Emily Davis     |        6 |     7 | B       | B-    | /assignments/98765_assignment.pdf | emily.davis@email.com     |
+---------------+-----------------+----------+-------+---------+-------+--------------------------------+-------------------------+
10 rows in set (0.00 sec)

mysql>
```

◉ CSV File “students.csv”:



students

12345	John Doe	1	10	A	A-	/assignments/12345_assignment.pdf	john.doe@email.com
67890	Jane Smith	2	9	B	B+	/assignments/67890_assignment.pdf	jane.smith@email.com
13579	Michael Johnson	3	11	C	C	/assignments/13579_assignment.pdf	michael.johnson@email.com
24680	Sarah Williams	4	8	D	B-	/assignments/24680_assignment.pdf	sarah.williams@email.com
54321	David Miller	5	12	A	A+	/assignments/54321_assignment.pdf	david.miller@email.com
98765	Emily Davis	6	7	B	B-	/assignments/98765_assignment.pdf	emily.davis@email.com
11223	Kevin Brown	7	10	C	C+	/assignments/11223_assignment.pdf	kevin.brown@email.com
33445	Olivia White	8	9	D	D	/assignments/33445_assignment.pdf	olivia.white@email.com
55667	Christopher Lee	9	11	A	A	/assignments/55667_assignment.pdf	christopher.lee@email.com
77889	Jessica Taylor	10	8	B	B	/assignments/77889_assignment.pdf	jessica.taylor@email.com

Project References

Tools and Frameworks

- **Python:**
  - Website: https://www.python.org/
- **Tkinter:**
  - Documentation: https://docs.python.org/3/library/tkinter.html
- **MySQL Connector/Python:**
  - Documentation: https://dev.mysql.com/doc/connector-python/en/
- **NLTK (Natural Language Toolkit):**
  - Website: https://www.nltk.org/

External Libraries

- **NLTK (Natural Language Toolkit):**
  - Installation: `pip install nltk`
  - Documentation: https://www.nltk.org/
- **MySQL Connector/Python:**
  - Installation: `pip install mysql-connector-python`
  - Documentation: https://dev.mysql.com/doc/connector-python/en/
- **Tkinter (Included with Python):**
  - Documentation: https://docs.python.org/3/library/tkinter.html

Online Resources

- **Stack Overflow:**
  - Website: https://stackoverflow.com/
- **GitHub:**
  - Repository: https://github.com/Rishi-Ahuja/
- **W3Schools:**
  - Website: https://www.w3schools.com/
- **Khan Academy:**
  - Website: https://www.khanacademy.org/

Project Contributors

- **Rishi Ahuja:**
  - Role: Project Developer
  - Contact: rishiahuja.1404@gmail.com

* * *