|  |
| --- |
| Photo displaying partial image of two pie charts on a canvas-textured page |
| Custom Student Records System  Final Project |
| |  |  |  | | --- | --- | --- | | Muhammad Sultan | 12/9/23 | Document Automation Python | |

**Group Members (Group-1)**

Hasti Rajnikant Donga – 200556287

Sarina Poudel – 200556414

Karan Gill – 200550743

Katsuya Luchi – 200538257

**Content**

**1.** [**Introduction**](#Introduction)………………………………………………………………………………………………..…..3

**1.1** [**Description of Title**](#Description)**……………………………………………………………………………………………..………..3**

**1.2** [**Python and Excel: A Dynamic Duo**](#python)**………………………………………………………..……………..……….3**

**1.3** [**Smart Functions for Smooth Sailing**](#Smart)**…………………………………………..…………………………..…....3**

**[2. Main Content…………………………………………………………………………………………...…4](#main)**

[**2.1 Opening the Record Spreadsheet……………….……….……………………………………..……………..…4**](#opening)

[**2.2 User-Defined Functions for Validation and Formatting……………………………………………….…4**](#user)

[**2.3 Interactive Menu and User Choices……………………………….………………..……………………….…...5**](#menu)

[**2.4 Adding a New Student Record ……………………….……………………………………………….…………...5**](#adding)

[**2.5 Updating and Deleting Existing Record of Student ………………………………………….…………….6**](#updating)

[**2.6 View the list of Student Records ……………….……………………………………………….…….………….7**](#viewing)

**[2.7](#Searching)****[Searching and Displaying a Specific Student Record …………….……………………………………….7](#Searching)**

[**2.8** **Exiting the Program ……………………………………………………….………………………………..…………..7**](#exting)

[**3. Technical Details…………………………………………………………………………………………..8**](#technical)

[**3.1 Data Storage and Retrieval…………………………………………………………………….………………………8**](#storage)

[**3.2 Input Validation…………………………………………………………………………………………………………….8**](#input)

[**3.3 User Interface……………………………………………………………………………………………..………………..8**](#Interface)

[**3.4 Error Handling……………………………………………………………………………………………………………….8**](#error)

**[4. Conclusion…………………………………………………………………………………………………….9](#conclusion)**

[**5. Appendix…………………………………………………………………………………………………..10**](#appendix)

Introduction

**1.1** **Description of Title**

This program in Python uses openpyxl for Excel connection to construct a basic student records system. It has features for viewing, updating, removing, and adding student records. The application assures data integrity by validating student IDs, semester dates, and CGPA forms. An additional security layer is added by using a password authentication method. The main loop offers an easy-to-use interface for manipulating student records in an Excel workbook. It's a flexible tool for organizing and preserving student data.

**1.2** **Python and Excel: A Dynamic Duo**

Our solution uses Python, a simple computer language, to act as a useful companion for establishments. It integrates smoothly with Excel, providing people with easy access to well-organized data. It's like having a proficient technology friend who ensures that everyone, including the less knowledgeable about technology, can utilize it. When it comes to utilizing technology to improve customer satisfaction in establishments, the above technological team is in the lead.

**1.3** **Smart Functions for Smooth Sailing**

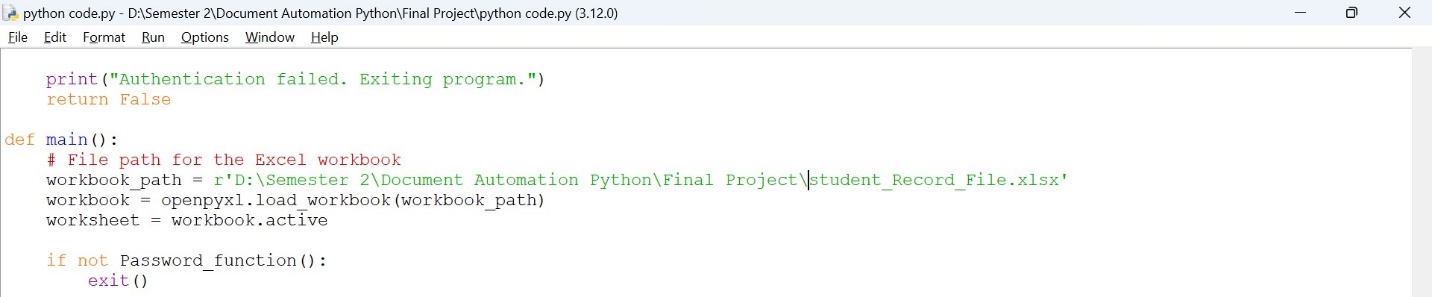
We use clever techniques in our programming to ensure that everything functions as it should. The accuracy of item names is checked, pricing and quantity are assessed using fundamental concepts, and prices are even formatted well. There is also an option to find an object within the Excel file.

Main Content

This Python script features a rudimentary student records system that is both straightforward and efficient. Let's look at its main characteristics and discover how it aids in maintaining an organized record by school/college operators.

**2.1 Opening the Record Spreadsheet**

The script imports the required libraries (re for regular expressions and openpyxl for Excel interaction). After that, the workbook and active worksheet are initialized by opening the designated Excel file (student\_Record\_File.xlsx).



**2.2 User-Defined Functions for Validation and Formatting**

Several user-defined functions are created to validate user inputs and manipulate data:

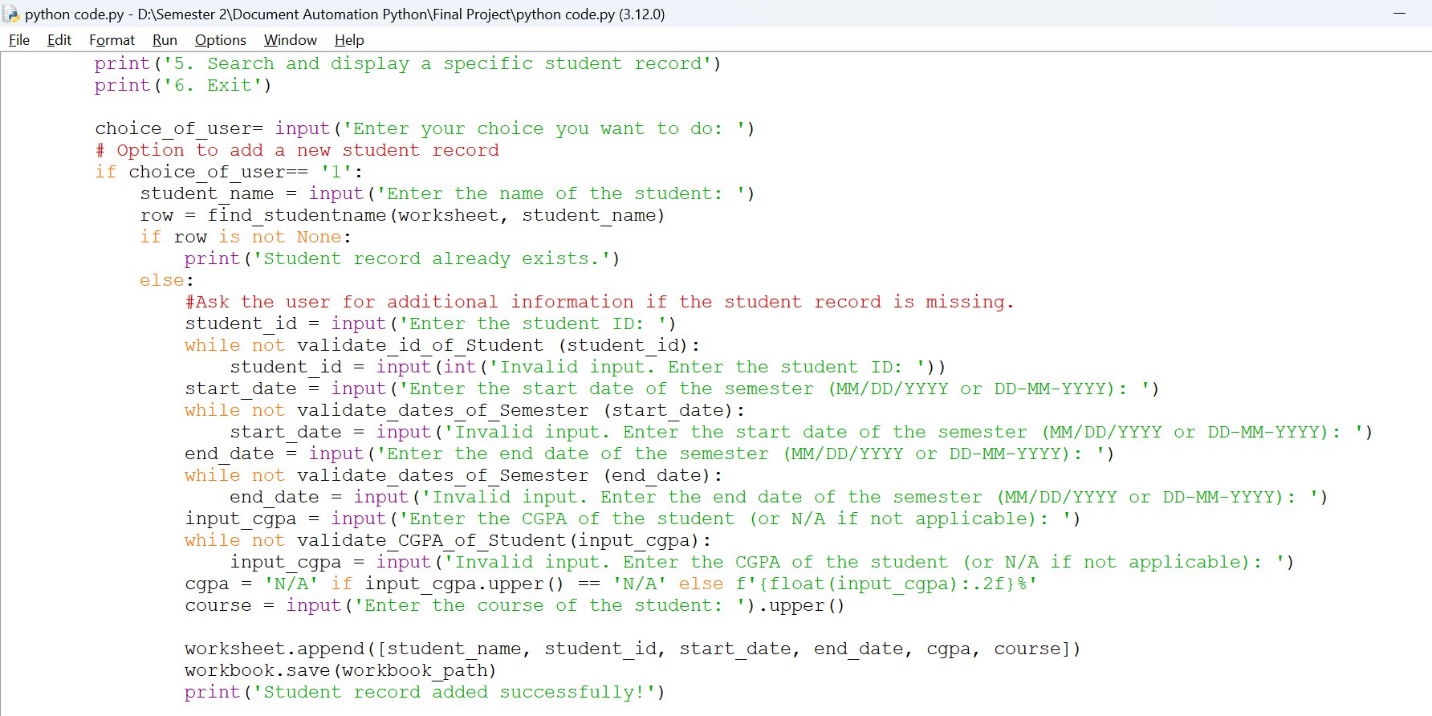
* **validate\_id\_of\_Student(input\_id):** This function verifies that a provided student ID only consists of letters and digits and follows an alphanumeric pattern.
* **validate\_dates\_of\_Semester(input\_date):** In charge of verifying if the start and finish dates of the semester are formatted in "MM/DD/YYYY" or "DD-MM-YYYY".
* **validate\_CGPA\_of\_Student(input\_cgpa):** Verifies that a student's CGPA is formatted correctly by accepting numbers with up to two decimal places or the string "N/A" in situations when the CGPA is irrelevant.
* **find\_studentname(sheet, name):** Depending on the student's name, finds the row number in the Excel worksheet, making it simple to get or edit student records.
* **display\_students\_records(sheet):** Presents the current student data in a tabular format, highlighting information such the student's name, ID, start and end dates, CGPA, course, and length of semester.
* **calculate\_semester\_duration(start\_date, end\_date):** Determines the number of days in a semester based on the start and end dates while accounting for various date formats. When a date input is invalid, it raises a ValueError.

**2.3 Interactive Menu and User Choices**

The code makes use of an interactive menu system that offers users a range of options for handling student information. To create a new student record, update an existing record, delete an entry, browse the complete list, search for a specific record, or quit the software, users are prompted to input their intended action. A sequence of conditional statements processes the user's selection and points the program to the appropriate action or function. By offering an intuitive and well-organized interface for organizing student data in the Excel spreadsheet, this interactive menu design improves the user experience.

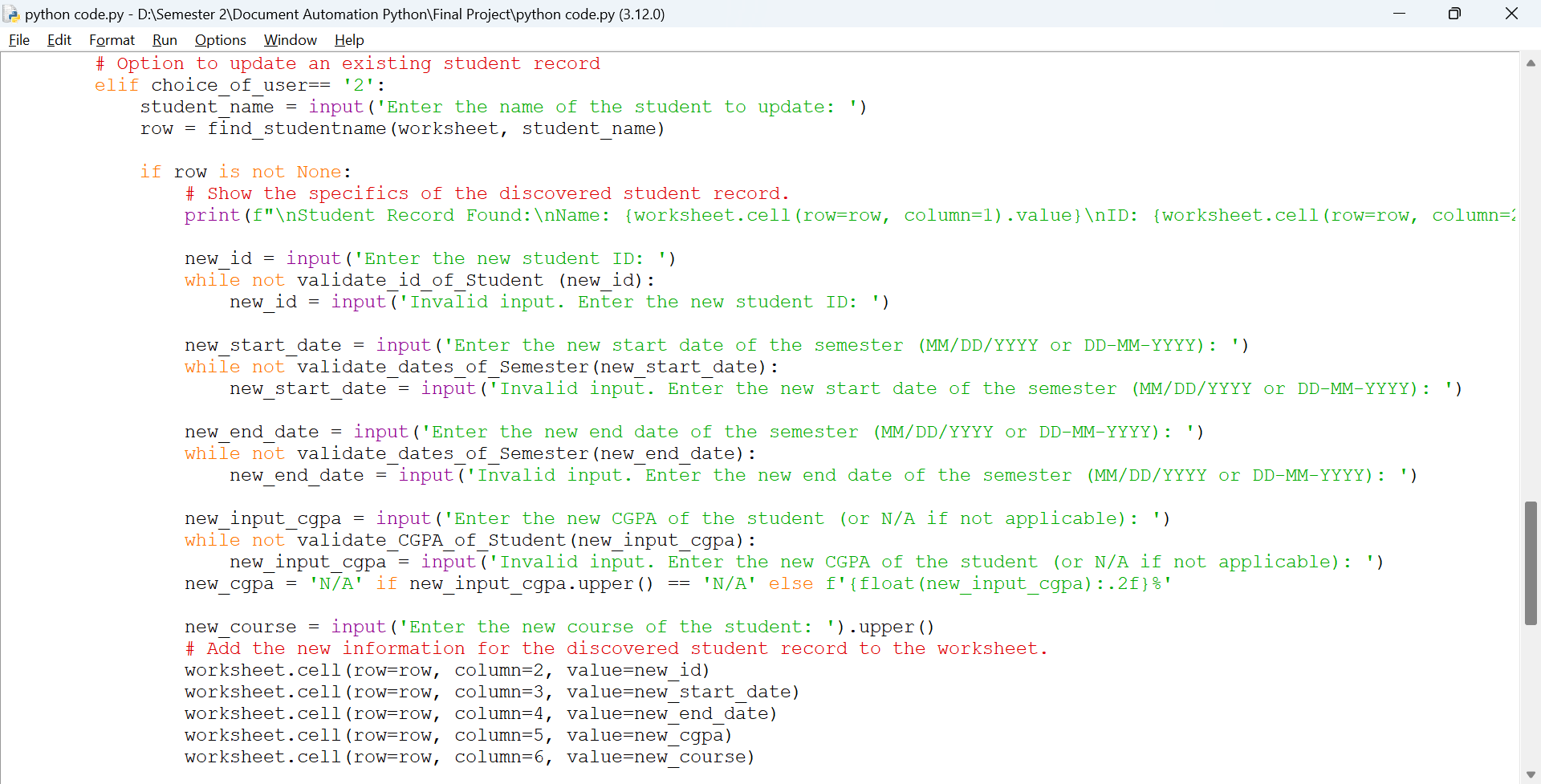
**2.4** **Adding a New Student Record**

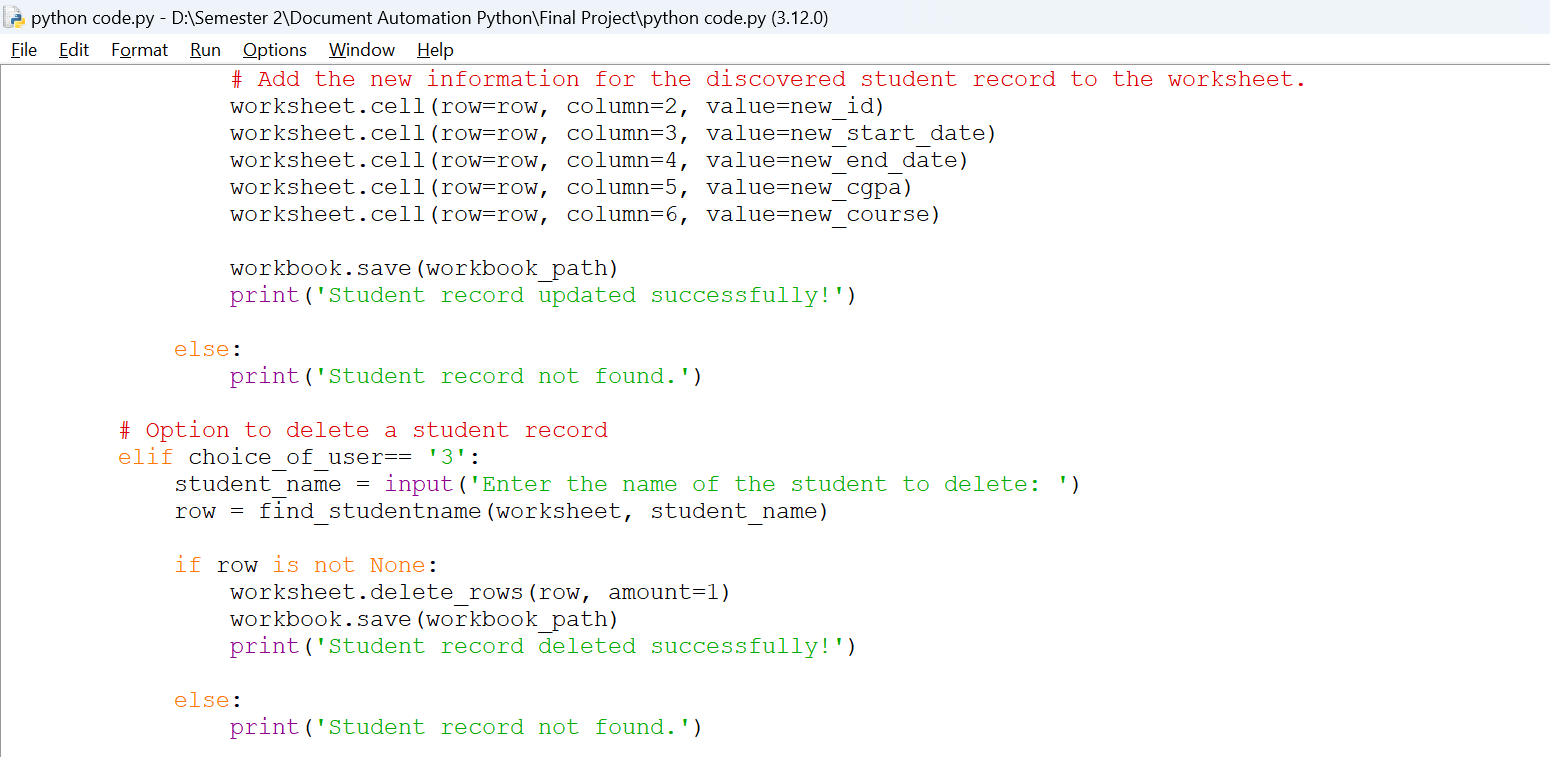
User input is used in the code to implement the process of adding a new student record. The user is prompted by the program to enter the student's ID, name, course, CGPA, start and end dates of the semester. Validation checks are carried out on inputs, including the format of student IDs, semester dates, and CGPA. The user is notified of a successful addition to the student records when the workbook is saved and the new student information is attached to the Excel worksheet if the entered student name is not found in the current records.



**2.5** **Updating and Deleting Existing Record of Student**

Features for adding and removing current student records are included in the code. When a user selects to update a record, new data entry fields such as the student ID, start and end dates, course, and CGPA are displayed. Before updating the entry in the Excel spreadsheet, the function makes sure that the data entered is validated. Similarly, users are required to provide the name of the student whose record they wish to erase when they choose to do so. After that, the code looks up the student's name and removes the entire record if it is discovered. This gives administrators a comprehensive way to manage and keep up-to-date student data.





**2.6** **View the list of Student Records**

The code includes a function called display\_students\_records whose purpose is to provide an organized list of the most recent student records. It collects data from the Excel spreadsheet using the openpyxl library and presents it in a tabular style, including student names, IDs, start and end dates, CGPA, course, and semester duration. This feature provides users with a thorough overview of all student data that is saved in the application.

**2.7** **Searching and Displaying a Specific Student Record**

By providing the student's name, the code enables users to search for and show particular student records. The application verifies that the supplied name is present in the spreadsheet after receiving input from the user. If a match is discovered, the student's full profile is displayed, along with their name, ID, semester start and end dates, CGPA, and course. With the help of this search feature, users can quickly and precisely access certain student records stored within the system.

**2.8** **Exiting the Program**

The user can choose to exit the application in the given code by making a selection in the interactive menu. A farewell message is displayed by the software and it exits the main loop when the user chooses option '6' to end the run. Furthermore, the user may receive an authentication failure warning and application termination if they are unable to authenticate with the right password after several tries.

Technical Details

**3.1 Data Storage and Retrieval**

To store and arrange student records, the code makes use of an Excel spreadsheet as a data storage mechanism. The spreadsheet stores the information in columns, containing the names, IDs, semester dates, CGPA, and courses of the students. With the help of the OpenPyXL library, the application effectively retrieves and manipulates this data, making it easier to store and retrieve data for maintaining student information.

**3.2 Input Validation**

Strong input validation procedures are implemented by the code to guarantee data integrity. For example, functions like validate\_id\_of\_Student, validate\_dates\_of\_Semester, and validate\_CGPA\_of\_Student are used to check the format and accuracy of student IDs, semester dates, and CGPA values, respectively, while adding or modifying student information. To guarantee that only correct and correctly formatted data is entered into the student records spreadsheet, these validation features ask users to re-enter the information if any input is deemed invalid.

**3.3 User Interface**

The given code has a text-based, console-driven user interface that provides a simple paradigm of interaction. An easy-to-navigate menu with available alternatives is displayed to users, making input and navigation effortless. By assisting users in adding, editing, and viewing student information in the Excel spreadsheet, the interface improves user experience and makes the system more approachable and user-friendly.

**3.4 Error Handling**

Strong error-handling techniques are integrated into the code to guarantee a seamless and error-free user experience. It has validation features to check CGPA submissions, semester date formats, and student ID formats. To ensure that wrong data inputs are handled graciously, the program also uses try-except blocks to handle potential issues while parsing date formats. Proactively managing errors improves the program's dependability and averts unforeseen problems when users interact with the student records system.

Conclusion

To sum up, the Custom Student Records System offers an example of how to use Python and Excel to manage student information in a real-world setting. Users may easily add, update, remove, and browse student records thanks to the code's user-friendly interface and interactive menu. User-defined functions are integrated into the code to improve readability and maintainability and to provide effective validation for formats such as student ID, semester dates, and CGPA. Furthermore, the code prevents anomalies in the spreadsheet by using functions to validate and format user inputs. This maintains data integrity.

By preventing unauthorized access to student records, the program's usage of a password authentication feature offers an extra degree of protection. The dynamic pair of Excel and Python demonstrates how flexible these tools are for automating and simplifying data management chores. The code's modular design, which includes functions for a range of activities, aids in its scalability and adaptability to possible future improvements. All things considered, the Custom Student Records System is a strong example of efficient document automation, providing a reliable way to manage student data in an educational environment.

Appendix

import openpyxl

from datetime import datetime

import re

# Verification function for student ID format

def validate\_id\_of\_Student (input\_id):

# Alphanumeric characters are matched by a regular expression pattern

pattern = r'^[A-Za-z0-9]+$'

return bool(re.match(pattern, input\_id))

# Validation function for semester date format

def validate\_dates\_of\_Semester(input\_date):

try:

datetime.strptime(input\_date, "%m/%d/%Y")

except ValueError:

try:

datetime.strptime(input\_date, "%d-%m-%Y")

except ValueError:

return False

return True

#Function to verify the student's CGPA format

def validate\_CGPA\_of\_Student(input\_cgpa):

if input\_cgpa.upper() == 'N/A':

return True

pattern = r'^\d+(\.\d{1,2})?$'

return bool(re.match(pattern, input\_cgpa))

# Function to determine a student's row number by name

def find\_studentname(sheet, name):

for i in range(2, sheet.max\_row + 1):

if sheet.cell(row=i, column=1).value == name:

return i

return None

# Capability to show every student record

def display\_students\_records(sheet):

print('\nCurrent Student Records:\n')

print(f'{"Name":<20}{"ID":<10}{"Start Date":<15}{"End Date":<15}{"CGPA":<10}{"Course":<20}{"Duration":<15}')

for i in range(2, sheet.max\_row + 1):

# Get values from the current row's cells.

name = sheet.cell(row=i, column=1).value

student\_id = sheet.cell(row=i, column=2).value

start\_date = sheet.cell(row=i, column=3).value

end\_date = sheet.cell(row=i, column=4).value

cgpa = sheet.cell(row=i, column=5).value

course = sheet.cell(row=i, column=6).value

# Use the calculate\_semester\_duration function to determine the duration.

duration = calculate\_semester\_duration(start\_date, end\_date)

# Publish information from student records in format.

print(f'{name:<20}{student\_id:<10}{start\_date:<15}{end\_date:<15}{cgpa:<10}{course:<20}{duration:<15}')

# Function to determine the length of a semester

def calculate\_semester\_duration(start\_date, end\_date):

if start\_date is None or end\_date is None:

raise ValueError("Start date and end date cannot be None.")

# Experiment with parsing the start\_date in various date formats.

try:

start\_date\_obj = datetime.strptime(start\_date, "%m/%d/%Y")

except ValueError:

try:

start\_date\_obj = datetime.strptime(start\_date, "%d-%m-%Y")

except ValueError:

raise ValueError("Invalid start date format. Please use MM/DD/YYYY or DD-MM-YYYY.")

# Experiment with parsing the end\_date in various date formats.

try:

end\_date\_obj = datetime.strptime(end\_date, "%m/%d/%Y")

except ValueError:

try:

end\_date\_obj = datetime.strptime(end\_date, "%d-%m-%Y")

except ValueError:

raise ValueError("Invalid end date format. Please use MM/DD/YYYY or DD-MM-YYYY.")

duration = end\_date\_obj - start\_date\_obj

return f'{duration.days} days'

# Password authentication function

def Password\_function():

correct\_password = "7070"

attempts\_left = 4

while attempts\_left > 0:

user\_password = input("Enter password: ")

if user\_password == correct\_password:

return True

else:

attempts\_left -= 1

print(f'Incorrect password. Attempts left: {attempts\_left}')

print("Authentication failed. Exiting program.")

return False

def main():

# File path for the Excel workbook

workbook\_path = r'D:\Semester 2\Document Automation Python\Final Project\student\_Record\_File.xlsx'

workbook = openpyxl.load\_workbook(workbook\_path)

worksheet = workbook.active

if not Password\_function():

exit()

print('\n=== Welcome to the Custom Student Records System ===')

while True:

print('\nOptions:')

print('1. Add a new student record')

print('2. Update an existing student record')

print('3. Delete a student record')

print('4. View the list of student records')

print('5. Search and display a specific student record')

print('6. Exit')

choice\_of\_user= input('Enter your choice you want to do: ')

# Option to add a new student record

if choice\_of\_user== '1':

student\_name = input('Enter the name of the student: ')

row = find\_studentname(worksheet, student\_name)

if row is not None:

print('Student record already exists.')

else:

#Ask the user for additional information if the student record is missing.

student\_id = input('Enter the student ID: ')

while not validate\_id\_of\_Student (student\_id):

student\_id = input(int('Invalid input. Enter the student ID: '))

start\_date = input('Enter the start date of the semester (MM/DD/YYYY or DD-MM-YYYY): ')

while not validate\_dates\_of\_Semester (start\_date):

start\_date = input('Invalid input. Enter the start date of the semester (MM/DD/YYYY or DD-MM-YYYY): ')

end\_date = input('Enter the end date of the semester (MM/DD/YYYY or DD-MM-YYYY): ')

while not validate\_dates\_of\_Semester (end\_date):

end\_date = input('Invalid input. Enter the end date of the semester (MM/DD/YYYY or DD-MM-YYYY): ')

input\_cgpa = input('Enter the CGPA of the student (or N/A if not applicable): ')

while not validate\_CGPA\_of\_Student(input\_cgpa):

input\_cgpa = input('Invalid input. Enter the CGPA of the student (or N/A if not applicable): ')

cgpa = 'N/A' if input\_cgpa.upper() == 'N/A' else f'{float(input\_cgpa):.2f}%'

course = input('Enter the course of the student: ').upper()

worksheet.append([student\_name, student\_id, start\_date, end\_date, cgpa, course])

workbook.save(workbook\_path)

print('Student record added successfully!')

# Option to update an existing student record

elif choice\_of\_user== '2':

student\_name = input('Enter the name of the student to update: ')

row = find\_studentname(worksheet, student\_name)

if row is not None:

# Show the specifics of the discovered student record.

print(f"\nStudent Record Found:\nName: {worksheet.cell(row=row, column=1).value}\nID: {worksheet.cell(row=row, column=2).value}\nStart Date: {worksheet.cell(row=row, column=3).value}\nEnd Date: {worksheet.cell(row=row, column=4).value}\nCGPA: {worksheet.cell(row=row, column=5).value}\nCourse: {worksheet.cell(row=row, column=6).value}")

new\_id = input('Enter the new student ID: ')

while not validate\_id\_of\_Student (new\_id):

new\_id = input('Invalid input. Enter the new student ID: ')

new\_start\_date = input('Enter the new start date of the semester (MM/DD/YYYY or DD-MM-YYYY): ')

while not validate\_dates\_of\_Semester(new\_start\_date):

new\_start\_date = input('Invalid input. Enter the new start date of the semester (MM/DD/YYYY or DD-MM-YYYY): ')

new\_end\_date = input('Enter the new end date of the semester (MM/DD/YYYY or DD-MM-YYYY): ')

while not validate\_dates\_of\_Semester(new\_end\_date):

new\_end\_date = input('Invalid input. Enter the new end date of the semester (MM/DD/YYYY or DD-MM-YYYY): ')

new\_input\_cgpa = input('Enter the new CGPA of the student (or N/A if not applicable): ')

while not validate\_CGPA\_of\_Student(new\_input\_cgpa):

new\_input\_cgpa = input('Invalid input. Enter the new CGPA of the student (or N/A if not applicable): ')

new\_cgpa = 'N/A' if new\_input\_cgpa.upper() == 'N/A' else f'{float(new\_input\_cgpa):.2f}%'

new\_course = input('Enter the new course of the student: ').upper()

# Add the new information for the discovered student record to the worksheet.

worksheet.cell(row=row, column=2, value=new\_id)

worksheet.cell(row=row, column=3, value=new\_start\_date)

worksheet.cell(row=row, column=4, value=new\_end\_date)

worksheet.cell(row=row, column=5, value=new\_cgpa)

worksheet.cell(row=row, column=6, value=new\_course)

workbook.save(workbook\_path)

print('Student record updated successfully!')

else:

print('Student record not found.')

# Option to delete a student record

elif choice\_of\_user== '3':

student\_name = input('Enter the name of the student to delete: ')

row = find\_studentname(worksheet, student\_name)

if row is not None:

worksheet.delete\_rows(row, amount=1)

workbook.save(workbook\_path)

print('Student record deleted successfully!')

else:

print('Student record not found.')

# Option to view the list of student records

elif choice\_of\_user== '4':

display\_students\_records (worksheet)

# Option to search and display a specific student record

elif choice\_of\_user== '5':

print('\nSearch and display a specific student record: ')

student\_name = input('Enter the name of the student: ')

row = find\_studentname(worksheet, student\_name)

if row is not None:

print(f'\nStudent Record Found:\nName: {worksheet.cell(row=row, column=1).value}\nID: {worksheet.cell(row=row, column=2).value}\nStart Date: {worksheet.cell(row=row, column=3).value}\nEnd Date: {worksheet.cell(row=row, column=4).value}\nCGPA: {worksheet.cell(row=row, column=5).value}\nCourse: {worksheet.cell(row=row, column=6).value}')

else:

print('Student record not found.')

# Option to exit the program

elif choice\_of\_user== '6':

print('\nExiting program of Final Project...')

break

else:

print('\nInvalid choice. Please try again.')

if \_\_name\_\_ == "\_\_main\_\_":

main()