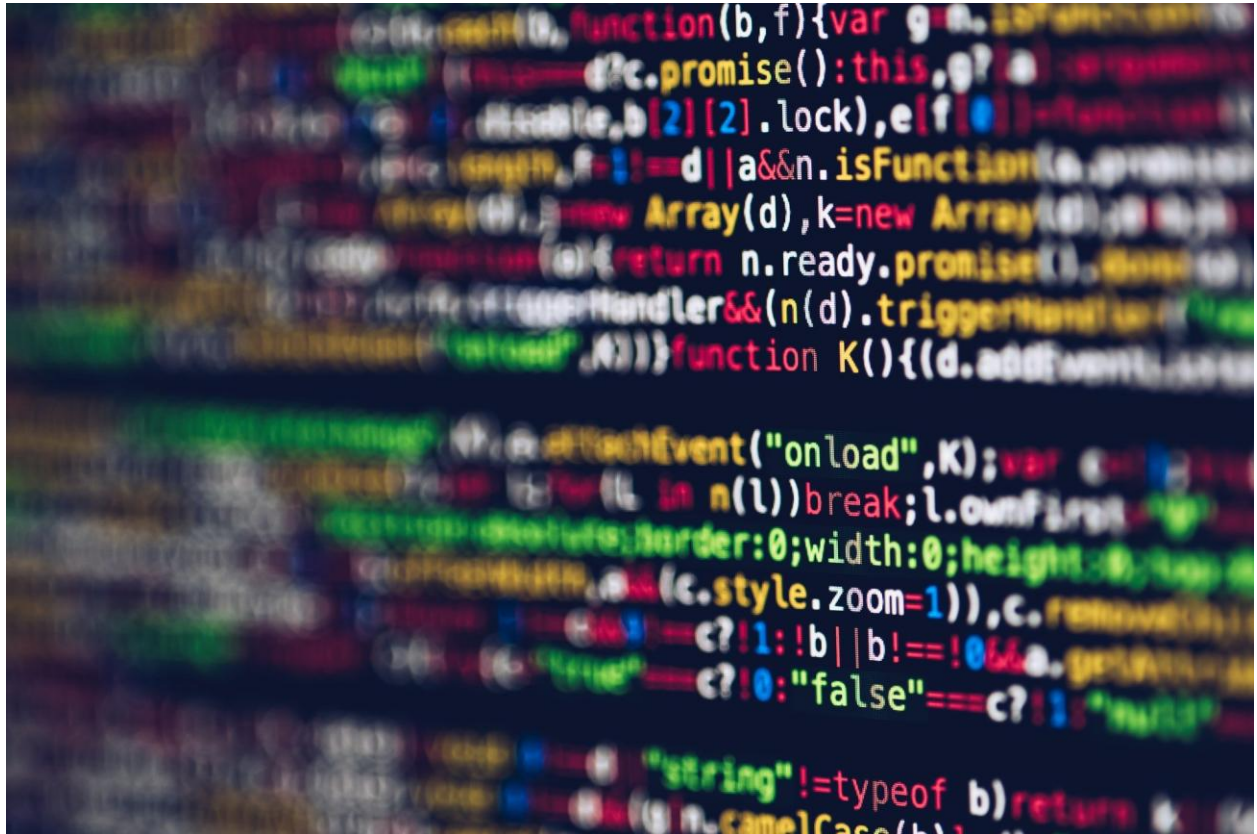


LBYCPA1-EQ3

Programming Logic and Design Laboratory



Final Project Documentation

DLSU Computer Engineering Student Progress Tracker

John Carlo E. Cheng Roa

ABSTRACT

Many students fail to monitor their academic standing due to delayed or missing grade updates on platforms like Instructure Canvas. This project aims to develop a Python-based console application that helps DLSU Computer Engineering students track their academic progress. The program uses the official ID124 flowchart to monitor subjects, compute GPA, and assess dean's list eligibility. It allows input of each course and GPA. Results show the system provides accurate and flexible tracking. This tool encourages early academic monitoring, helping students make informed decisions and avoid unexpected academic outcomes.

Keywords: Academic Tracking, GPA Computation, Dean's List Evaluation.

INTRODUCTION

Students often fail classes because they haven't been keeping track of their grades that often leads to unexpected failures in their classes. This issue is frequently caused by the grade not being updated in platforms like Instructure Canvas. This results in students to forget to regularly monitor their progress or miscalculate their standing overall. To address this, the DLSU Computer Engineering Student Progress Tracker aims to help students to actively monitor their academic journey throughout their course. This project tackles the common challenges of manually calculating and storing grades, computing the term GPA, and determining the eligibility for the dean's list in the specific term. A study by the ACT Organization (2012) emphasizes that early monitoring of student progress significantly boosts college performance and career readiness.

The design of the Student Progress Tracker came from several key computing and academic principles. The program implements file handling via .json for structured data management to manage the student's record and course information, and basic algorithm design for GPA computation. GPA calculation is based on the weighted average method used by many institutions, including DLSU, where each subject grade is multiplied by its respective unit count, summed, and then divided by the total number of units. The system also makes use of data validation and exception handling to ensure reliable user input and prevent program crashes.

The primary objectives of the DLSU Computer Engineering Student Progress Tracker are as follows:

- To assist students in computing their standings in their courses.
- To track courses based on the ID124 Computer Engineering Flowchart.

The following core features have been implemented in the program at this state:

- Save File Creation, Deletion, and Save Management
- Integration of the Official Computer Engineering ID124 flowchart.
- Automated Term GPA Computation and Dean's List Eligibility Calculation.
- Real-Time tracking and editing of courses in the save file.
- Create a save file from scratch or from a template based on the flowchart.
- Error handling throughout the program, including the save file.

PROGRAM OVERVIEW AND FLOWCHART

a. Input-Process-Output

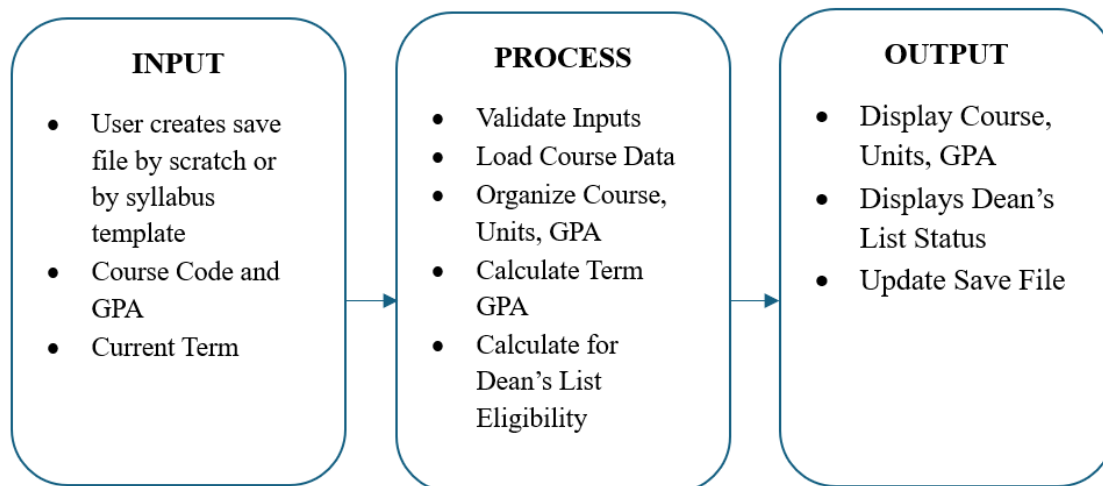


Fig. 1. The IPO flowchart.

The DLSU Computer Engineering Student Progress Tracker is a console-based Python application that helps students monitor their academic standing. It allows users to create a student profile, input subject grades, track progress based on the Computer Engineering course flowchart, and compute their GPA. It also checks for dean's list eligibility.

The IPO structure was gathered directly from the academic issues the project is designed to solve:

- The Input reflects the need for a complete academic information management system (course, GPA, units) that students often overlook or miscalculate due to manual tracking/ineffective tracking.
- The Process reflects the core functions that address the problem, such as GPA computation and checking for Dean's List eligibility. Tasks that students usually do manually or inefficiently.
- The Output delivers the exact academic statistics that students need but don't get instantly from platforms like Canvas, such as Term GPA calculations, eligibility status, and an effective academic progress system that helps them stay informed about their grades and make better decisions for their future.

b. Flowchart

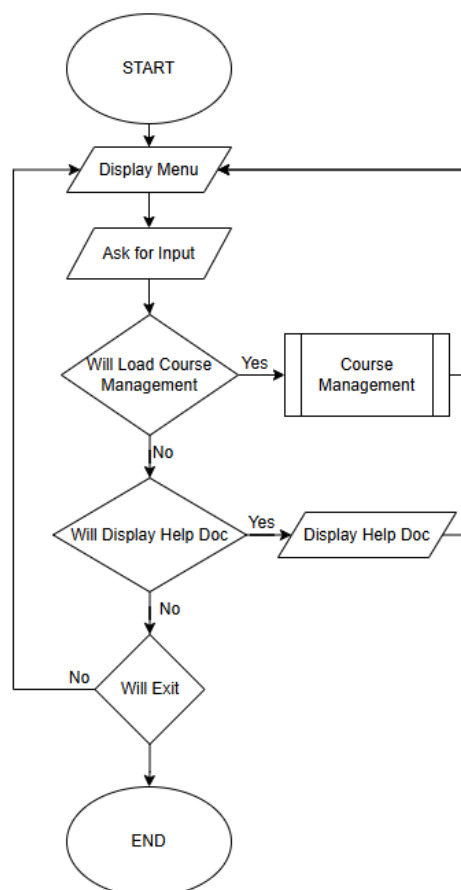


Fig. 2. Main Menu Flowchart.

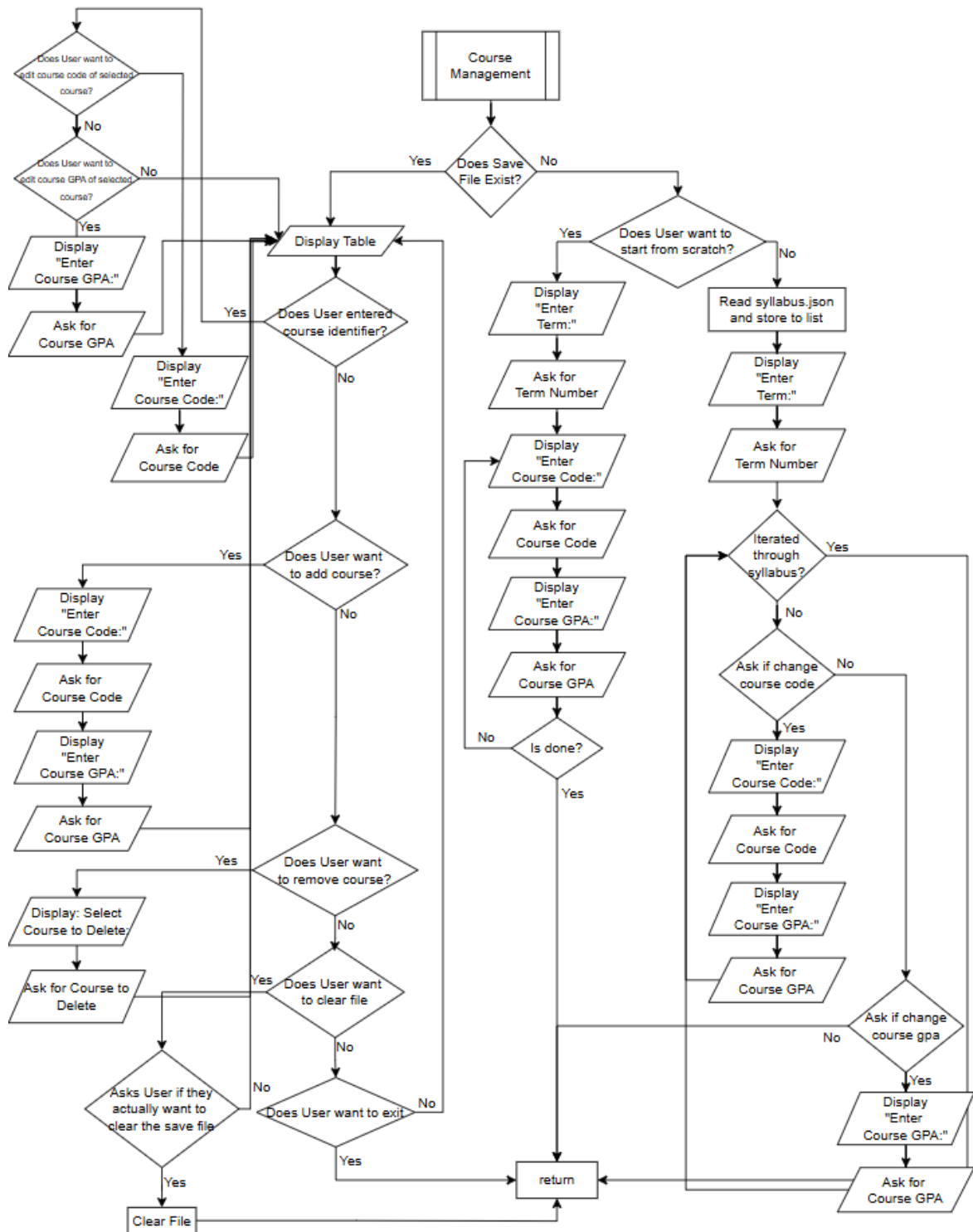
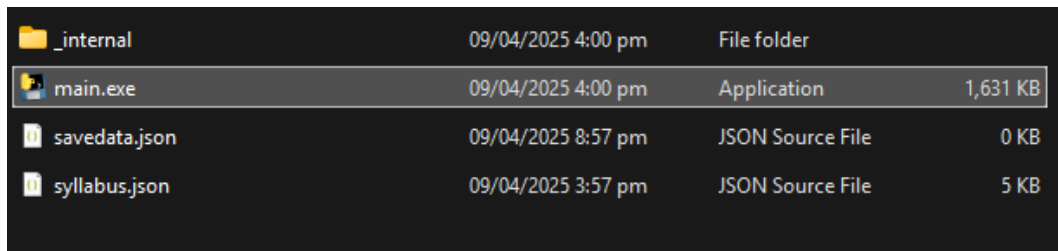


Fig. 3. Course Management Flowchart.

PROGRAM FEATURES AND SCREENSHOTS

(a) Running the Program



_internal	09/04/2025 4:00 pm	File folder	
main.exe	09/04/2025 4:00 pm	Application	1,631 KB
savedata.json	09/04/2025 8:57 pm	JSON Source File	0 KB
syllabus.json	09/04/2025 3:57 pm	JSON Source File	5 KB

Fig. 4. Running the Program.

To begin using the DLSU Computer Engineering Student Progress Tracker, users can run the program either by executing the .py file through a Python environment like Anaconda Prompt or by launching the compiled .exe file provided for easier access without additional setup. But take note that this program is intended for computers running Microsoft Windows with Python 3.12 installed.

(b) Main Menu

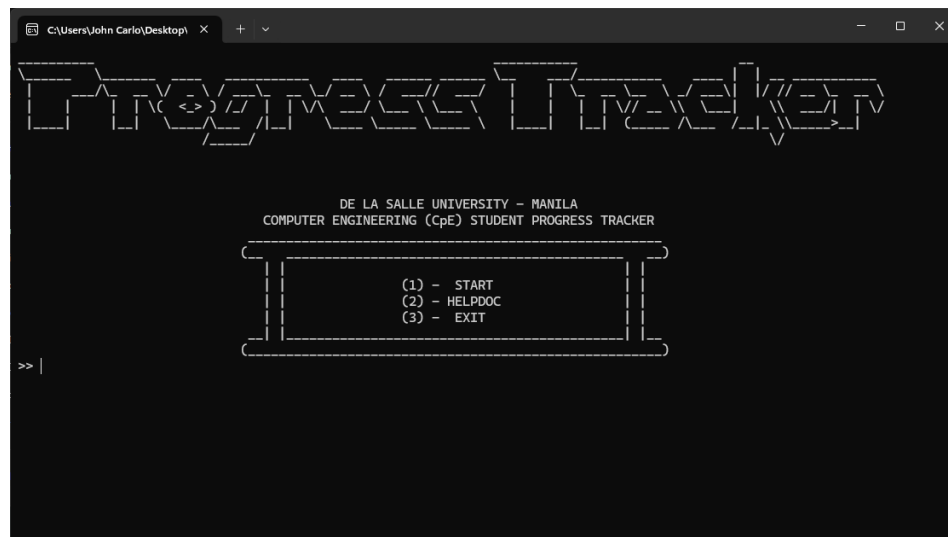


Fig. 5. Main Menu.

Upon the startup of the program, the user is presented a main menu that allows users to start, view the help doc, or exit. The user must enter a number ranging from 1-3, otherwise the program will recognize it as an invalid input. If it is the user's first time using the program, entering 1 will start the create a save file menu, otherwise it will start the course management menu.

(c) Create New Save Menu

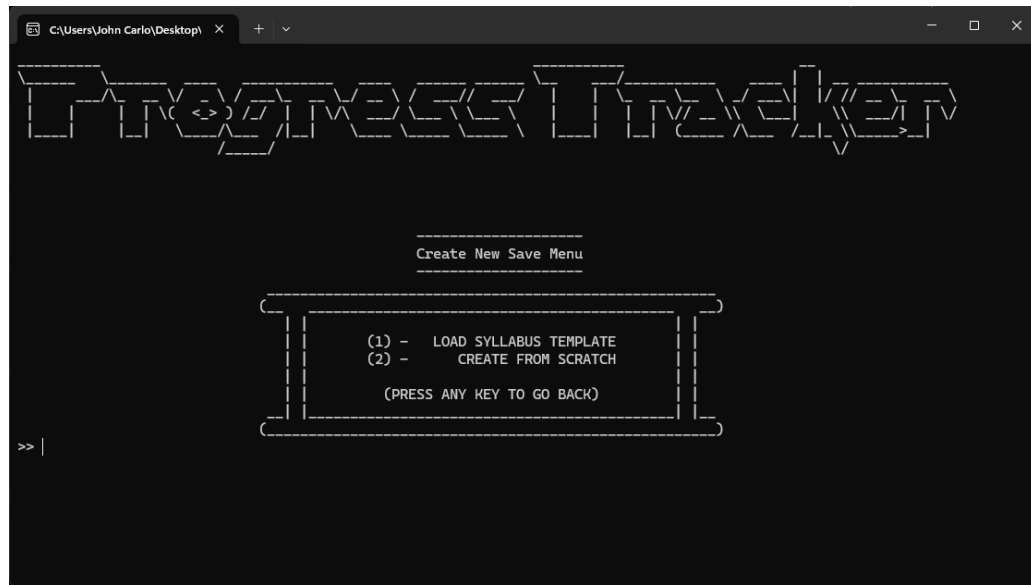


Fig. 6. Create Save Menu

When creating a new save, the user is prompted to choose between Load Syllabus Template and Create from Scratch with their respective number identifier. The user must pick between 1 or 2, any other input will send the user back to the main menu without saving any changes.

(d) Create a Save File from Scratch

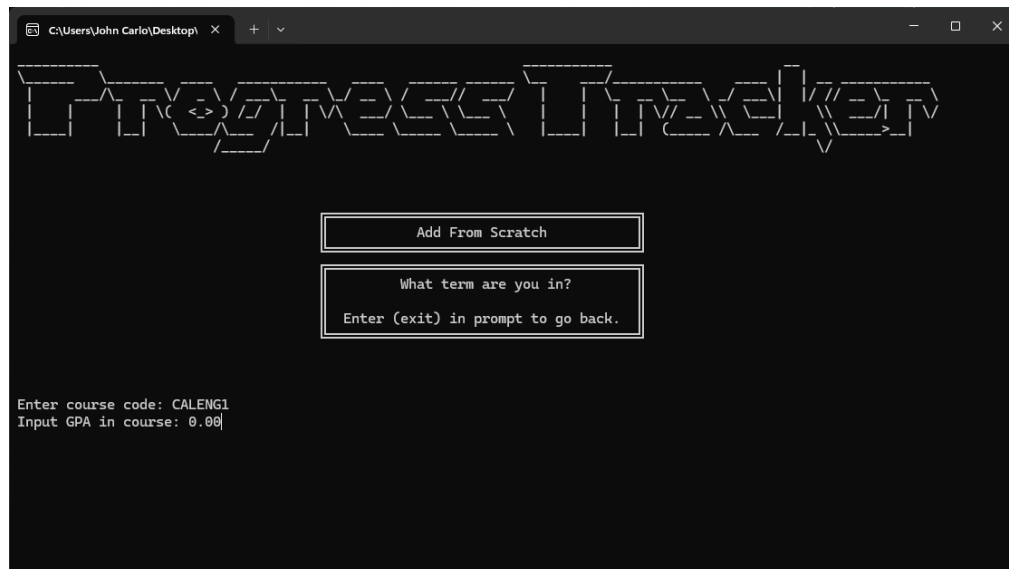


Fig. 7. Add From Scratch Menu

In this part of the create new save feature, the user is first prompted the term they are in, then they are prompted to enter the course code (the seven-digit course code) and the GPA they received in the course. This will continue in a loop until the user enters the word “exit” to the prompt when asked for the course code and it will return to the main menu.

(e) Create a Save File from a Syllabus Template

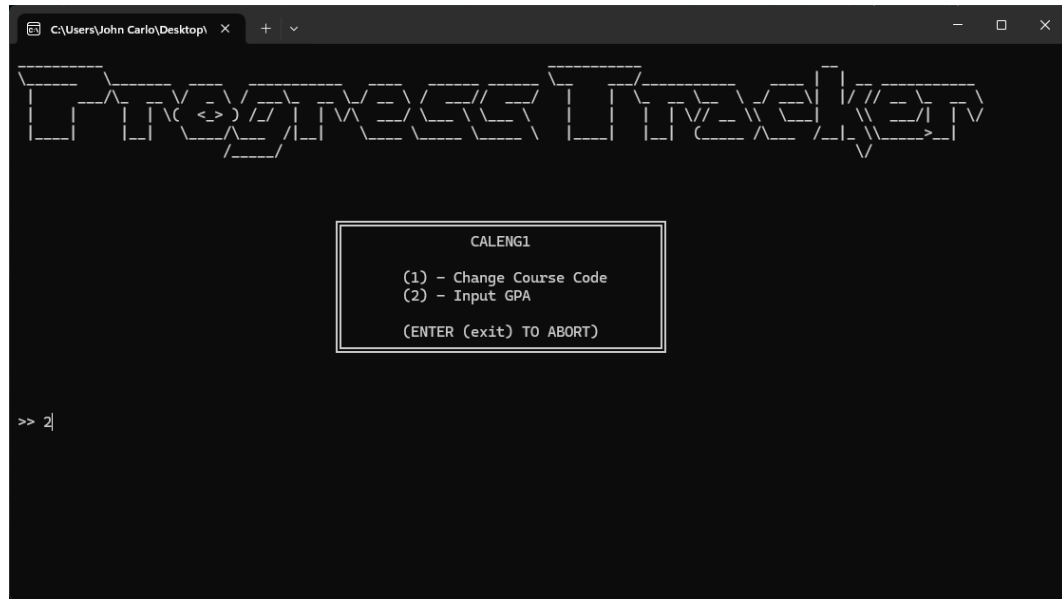


Fig. 8. Load Syllabus Template Menu

In this part of the create new save feature, the user is first prompted the term they are in, in which it will determine the set of courses that the program will list in the menu. The player here has two choices, either to change the course code of the current course selected or Input the GPA if the current course code is correct. The change course code is most useful for GE or LC courses that is not defined in the flowchart, or if you have deviated from the flowchart. Changing the course code will also prompt the user to input the GPA of the new course, so the user will have to enter a GPA for that highlighted course regardless. Entering exit when in the prompt in Fig. 8., will cause the feature to abort early, which means all of the courses the user has imported before will be saved into the save file.

(f) Course Management Menu

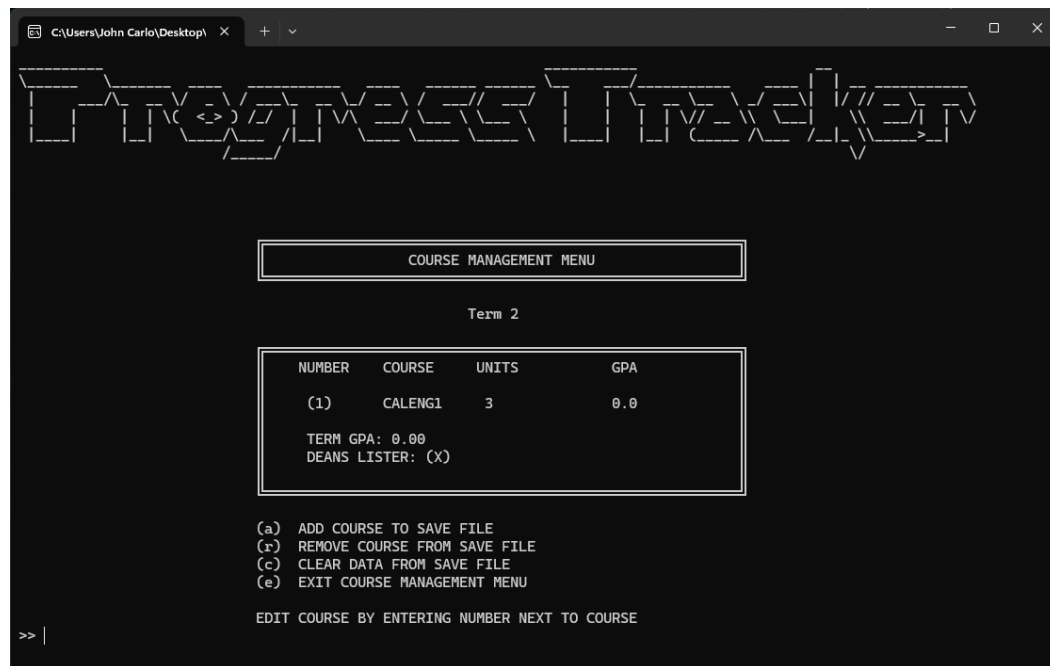


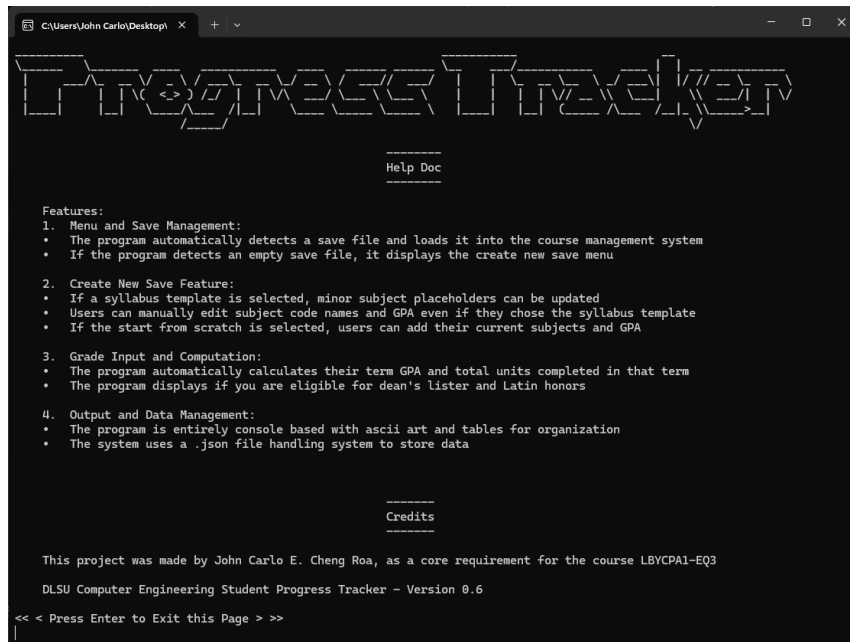
Fig. 9. Course Management Menu

This is the main feature of the program, wherein it is accessible if there is data detected in the save file. It displays the current term that the user entered and displays the course code, the unit of the respective course, and the GPA of the course. Below the list also displays the Term GPA and the eligibility of the user for dean's list.

The user has multiple choices, which includes entering "a" to add a course to the save file, which includes entering the course code and the GPA of the course, which will be added to the bottom of the list. Entering "r" to remove a course from the save file, which the user selects a course identifier number to delete the course without changing the order of the list. Entering "c" clears the data from the save file, but has a warning which the user must input either "Y" or "yes" to continue. And "e" to exit the course management menu and returning back to the main menu.

Entering a course identifier number like "1" selects the course, which in this instant is CALENG1, users can choose to replace the course code or replace the GPA of the selected course. Entering anything else will be ignored. This menu is updated real time, which means every input the user makes overwrites the save file.

(g) Help Doc



```
C:\Users\John Carlo\Desktop>
DLSU Computer Engineering Student Progress Tracker

Help Doc

Features:
1. Menu and Save Management:
  • The program automatically detects a save file and loads it into the course management system
  • If the program detects an empty save file, it displays the create new save menu
2. Create New Save Feature:
  • If a syllabus template is selected, minor subject placeholders can be updated
  • Users can manually edit subject code names and GPA even if they chose the syllabus template
  • If the start from scratch is selected, users can add their current subjects and GPA
3. Grade Input and Computation:
  • The program automatically calculates their term GPA and total units completed in that term
  • The program displays if you are eligible for dean's lister and Latin honors
4. Output and Data Management:
  • The program is entirely console based with ascii art and tables for organization
  • The system uses a .json file handling system to store data

Credits

This project was made by John Carlo E. Cheng Roa, as a core requirement for the course LBYCPA1-EQ3
DLSU Computer Engineering Student Progress Tracker - Version 0.6

<< Press Enter to Exit this Page >>
```

Fig. 10. Help Doc Menu

This part of the program is accessible by entering 3 in the main menu. It displays the features of the program and what the user can expect to do with the program, as well as the credits and version number. The user presses enter to exit the page and return back to the main menu.

(h) Exit



```
C:\Users\John Carlo\Desktop>
Are you sure?
[1] - Yes
[2] - No

>> 1
```

Fig. 11. Exit Confirmation Menu

This part of the program is the only way the user can exit the program through the program itself. This is accessible by entering 4 in the main menu. The user is expected to enter either 1 to exit the program completely or 2 to return to the main menu.

CONCLUSION AND FUTURE IMPROVEMENTS

In summary, the DLSU Computer Engineering Student Progress Tracker solves the problem of students losing track of their grades due to slow or incomplete updates on platforms like Instructure Canvas. It helps students easily monitor their grades, track subjects, calculate GPA, and check dean's list eligibility. The program met its goals by allowing save file management, course tracking using the official flowchart, and flexible subject adjustments. Overall, it makes academic progress clearer and easier to manage.

In the future, I plan to reorganize the entire project with the new features I'm going to implement after LBYCPA1. For starters, I would implement an option to add up to three save files, each with multiple and viewable terms where users can build their own course flowchart or use the ID124 flowchart. I would make the GUI inside of the console application simpler and more consistent. I may not have much time to implement these changes for LBYCPA1, but you can look up this project's source code here: <https://github.com/jchengroa/dlsucpespt>.

REFERENCES

ACT. (2009). Staying on target: The importance of monitoring student progress to ensure college and career readiness.

<https://www.act.org/content/dam/act/unsecured/documents/Staying-on-Target.pdf>

Python Software Foundation. (n.d.). json — JSON encoder and decoder.

<https://docs.python.org/3/library/json.html>

W3Schools. (n.d.). Python dictionaries.

https://www.w3schools.com/python/python_dictionaries.asp

Python Software Foundation. (n.d.). Data structures.

<https://docs.python.org/3/tutorial/datastructures.html>

Programiz. (n.d.). Python JSON: Read, write, parse JSON (with examples).

<https://www.programiz.com/python-programming/json>

Real Python. (2024, December 3). Understanding Python exceptions and error handling.

<https://realpython.com/python-exceptions/>

Python Software Foundation. (n.d.). Errors and exceptions. In Python 3.13.2 documentation.

<https://docs.python.org/3/tutorial/errors.html>

APPENDIX

main.py

```
# Import Project Files
from menu import *
import coursemanagement as cm

# Import System Files
import os

# Main Functions
def clr():
    """
    <DOCSTRING: CLR>
    This function clears the screen, Adaptive to any OS.
    """
    try:
        os.system('cls') # Windows based system
    except Exception:
        os.system('clear') # macOS and Linux based system

# The Main Function
def main():
    """
    <DOCSTRING: MAIN>
    This Function is where the program starts!
    """

    # Local Variable
    options = "0"
    errorhandline1 = int()

    while True:
        clr()
        title_menu()
        if errorhandline1 == 1:
            print("Invalid Input, Try Again!")

        try:
            options = input(">> ")
        except KeyboardInterrupt:
            options = " "

        if options == "1":
            errorhandline1 = 0
            cm.createorsave()
            pass
        elif options == "2":
            errorhandline1 = 0
```



```

def menubox():
    """
    <DOCSTRING: MENUBOX>
    This Function displays user options in the main menu
    """
    print(r"_____".center(114))
    print(r"(_ _ _ _ _)".center(114))
    print(r" | | | | ".center(114))
    print(r" | | (1) - START | | ".center(114))
    print(r" | | (2) - HELPDOC | | ".center(114))
    print(r" | | (3) - EXIT | | ".center(114))
    print(r" _| | _ _ _ _ _ | | _ ".center(114))
    print(r"(_ _ _ _ _)".center(114))

def cmloadmenu():
    """
    <DOCSTRING: COURSE MANAGEMENT MENU FOR CREATION>
    This Function displays user options if savedata.json is empty
    """
    displayheading("Create New Save Menu")
    print(r"_____".center(114))
    print(r"(_ _ _ _ _)".center(114))
    print(r" | | | | ".center(114))
    print(r" | | (1) - LOAD SYLLABUS TEMPLATE | | ".center(114))
    print(r" | | (2) - CREATE FROM SCRATCH | | ".center(114))
    print(r" | | | | ".center(114))
    print(r" | | (PRESS ANY KEY TO GO BACK) | | ".center(114))
    print(r" _| | _ _ _ _ _ | | _ ".center(114))
    print(r"(_ _ _ _ _)".center(114))

def addcourse():
    """
    <DOCSTRING: COURSE MANAGEMENT ADD FROM SCRATCH>
    This Function prints a dialog for the add from scratch section
    """
    print(r"_____".center(114))
    print(r" | | Add From Scratch | | ".center(114))
    print(r"_____".center(114))
    print(r" | | | | ".center(114))
    print(r" | | What term are you in? | | ".center(114))
    print(r" | | | | ".center(114))
    print(r" | | Enter (exit) in prompt to go back. | | ".center(114))
    print(r"_____".center(114))

def addfromsyb():
    """
    <DOCSTRING: COURSE MANAGEMENT ADD FROM SYLLABUS>
    This Function prints a dialog for the add from syllabus section
    """
    print(r"_____".center(114))
    print(r" | | Add From Syllabus Template | | ".center(114))
    print(r"_____".center(114))
    print(r" | | | | ".center(114))

```

```

print(r"      What term are you in?      || ".center(114))
print(r"                                || ".center(114))
print(r"      Enter (exit) in prompt to go back. || ".center(114))
print(r"_____|| ".center(114))

```

```
def sybcourse(course):
```

```

    """
    <DOCSTRING: COURSE MANAGEMENT ADD FROM SYLLABUS EDIT>
    This Function prints a dialog to edit through the courses in the syllabus in the selected term
    """

    print(r"_____|| ".center(114))
    print(f"      {course}      || ".center(114))
    print(r"                                || ".center(114))
    print(r"      (1) - Change Course Code      || ".center(114))
    print(r"      (2) - Input GPA                || ".center(114))
    print(r"                                || ".center(114))
    print(r"      (ENTER (exit) TO ABORT)      || ".center(114))
    print(r"_____|| ".center(114))

```

```
def coursemanagementmenu(term, termword, course, tgpa, unitlist):
```

```

    """
    <DOCSTRING: COURSE MANAGEMENT MENU>
    This Function displays the courses and gpa in the savedata.json file
    """

    logo()
    print("\n\n\n")
    n = 1

```

```

print(r"_____|| ".center(114))
print(r"                                || ".center(114))

```

```

print(r"_____|| ".center(114))
print(r"                                || ".center(114))
print(r"                                || ".center(114))
if len(str(term)) == 2:
    print(f"{termword:^53}".center(114))
elif len(str(term)) == 1:
    print(f"{termword:^53}".center(114))
print(r"                                || ".center(114))

```

```

print(r"_____|| ".center(114))
print(r"                                || ".center(114))
print(r"                                || ".center(114))

```

```

for keys, values in course.items():
    if type(values) == float:
        displaygrades = f"    ({str(n)})    {keys}    {unitlist[n-1]}    {values:.1f}    "
    else:
        displaygrades = f"    ({str(n)})    {keys}    {unitlist[n-1]}    {values}    "
    print(f" || {displaygrades:^53} || ".center(114))

```

```

    n += 1

    print(r"|| " .center(114))
    print(f"|| TERM GPA: {tgpa} " .center(114))

    gpacheck = 0
    for gpachecker in course.values():
        if gpachecker == " P":
            continue
        if gpachecker == " F":
            gpacheck += 1
            break
        if float(gpachecker) < 2:
            gpacheck += 1
            continue
        if float(gpachecker) >= 2:
            continue

    if float(tgpa) >= 3.0 and float(tgpa) < 3.5 and gpacheck == 0:
        print(r"|| DEANS LISTER: 2ND " .center(114))
    elif float(tgpa) >= 3.5 and gpacheck == 0:
        print(r"|| DEANS LISTER: 1ST " .center(114))
    else:
        print(r"|| DEANS LISTER: (X) " .center(114))

    print(r"|| " .center(114))

    print(r"|| " .center(114))
    return n

def cm_displayoptions():
    """
    <DOCSTRING: COURSE MANAGEMENT MENU DISPLAY OPTIONS>
    This Function displays the options the user can do in the course management menu
    """
    print("")
    print(f"{'(a) ADD COURSE TO SAVE FILE " .center(114)}")
    print(f"{'(r) REMOVE COURSE FROM SAVE FILE " .center(114)}")
    print(f"{'(c) CLEAR DATA FROM SAVE FILE " .center(114)}")
    print(f"{'(e) EXIT COURSE MANAGEMENT MENU " .center(114)}")
    print("")
    print(f"{'EDIT COURSE BY ENTERING NUMBER NEXT TO COURSE':<58} " .center(114))

def cm_editcourse(course, mode=1):
    """
    <DOCSTRING: COURSE MANAGEMENT EDIT COURSE MENU>
    This Function displays the edit options the user can do after selecting a course
    """
    if mode == 1:

    print(r"|| " .center(114))

```



```

        print(f"          (EDIT {course})           || ".center(114))
        print(r"          || ".center(114))
        print(f"{'(1) EDIT COURSE CODE' :^53}|| ".center(114))
        print(f"{'(2) EDIT COURSE GPA':^53}|| ".center(114))
        print(r"          || ".center(114))
        print(f"{'PRESS ENTER TO GO BACK':^53}|| ".center(114))

print(r"=====|| ".center(114))
4))
    if mode == 2:

print(r"=====|| ".center(114))
4))
        print(f"          (EDIT {course})           || ".center(114))
        print(r"          || ".center(114))
        print(f"          ENTER {course} NEW COURSE CODE      || ".center(114))
        print(r"          || ".center(114))
        print(f"{'PRESS ENTER TO GO BACK':^53}|| ".center(114))

print(r"=====|| ".center(114))
4))
    if mode == 3:

print(r"=====|| ".center(114))
4))
        print(f"          (EDIT {course})           || ".center(114))
        print(r"          || ".center(114))
        print(f"          ENTER {course} NEW GPA              || ".center(114))
        print(r"          || ".center(114))
        print(f"{'PRESS ENTER TO GO BACK':^53}|| ".center(114))

print(r"=====|| ".center(114))
4))

def cm_addcourse(stage=1):
    """
    <DOCSTRING: COURSE MANAGEMENT ADD COURSE MENU>
    This Function displays the add course (course code and gpa)
    """
    if stage == 1:
        print(r"=====|| ".center(114))
        print(f"          (ADD COURSE)           || ".center(114))
        print(r"          || ".center(114))
        print(f"          ENTER THE COURSE CODE      || ".center(114))
        print(r"          || ".center(114))
        print(f"          PRESS ENTER TO GO BACK     || ".center(114))
        print(r"=====|| ".center(114))
    if stage == 2:
        print(r"=====|| ".center(114))
        print(f"          (ADD COURSE)           || ".center(114))
        print(r"          || ".center(114))
        print(f"          ENTER THE COURSE GPA      || ".center(114))
        print(r"          || ".center(114))

```

```

print(f"|| PRESS ENTER TO GO BACK ||".center(114))
print(r"=====||".center(114))

def cm_deletecourse():
    """
    <DOCSTRING: COURSE MANAGEMENT DELETE COURSE MENU>
    This Function displays the instructions on how to remove a course entry
    """

    print(r"=====||".center(114))
    print(f"|| (REMOVE COURSE) ||".center(114))
    print(r"||".center(114))
    print(f"|| ENTER THE NUMBER NEXT TO COURSE TO REMOVE IT FROM SAVE FILE ||".center(114))
    print(f"||".center(114))
    print(f"|| PRESS ENTER TO GO BACK ||".center(114))

    print(r"=====||".center(114))

def displayheading(heading):
    """
    <DOCSTRING: DISPLAY HEADING>
    This Function prints a heading, adaptive to the length of the argument
    """

    spacesheading = "-"*len(heading)
    print("\n", spacesheading.center(114), "\n", heading.center(114), "\n", spacesheading.center(114))

def title_menu():
    """
    <DOCSTRING: TITLE_MENU>
    This Function is where the main menu is displayed.
    """

    logo()
    print("\n\n", "DE LA SALLE UNIVERSITY - MANILA".center(114), "\n", "COMPUTER ENGINEERING (CpE) STUDENT PROGRESS TRACKER".center(114))
    menubox()

def help_menu():
    """
    <DOCSTRING: HELP_MENU>
    This Function is where the the help doc and the credits of the project are stored
    """

    logo()

    displayheading("Help Doc")
    print(r"=====")
    Features:
    1. Menu and Save Management:
    • The program automatically detects a save file and loads it into the course management system
    • If the program detects an empty save file, it displays the create new save menu

    2. Create New Save Feature:

```

- If a syllabus template is selected, minor subject placeholders can be updated
- Users can manually edit subject code names and GPA even if they chose the syllabus template
- If the start from scratch is selected, users can add their current subjects and GPA

3. Grade Input and Computation:

- The program automatically calculates their term GPA and total units completed in that term
- The program displays if you are eligible for dean's list and Latin honors

4. Output and Data Management:

- The program is entirely console based with ascii art and tables for organization
- The system uses a .json file handling system to store data

```

"""
displayheading("Credits")
print(r"""
This project was made by John Carlo E. Cheng Roa, as a core requirement for the course LBYCPA1-EQ3

DLSU Computer Engineering Student Progress Tracker - Version 0.6
""")

def create_menu():
    """
    <DOCSTRING: CREATE_MENU>
    This function displays a menu for creating a save file
    """
    logo()
    displayheading("Create Save Menu")

```

coursemanagement.py

```

# Import System Files
import os
import json

# Import Project Files
from menu import *
from main import clr

# Function that reads the JSON file with Error Handling
def read_jsonfile(file_name):
    """
    <DOCSTRING>
    This function reads the JSON File passed in this function. It creates the file if not found.
    """
    if os.path.exists(file_name) and os.path.getsize(file_name) > 0:
        try:
            with open(file_name, 'r') as file:
                data = json.load(file)
            if file_name == "savedata.json":
                if data["grades"] == {}:
                    with open(file_name, 'w') as rwfile:
                        rwfile.truncate(0)
            return data
        except json.JSONDecodeError:

```

```

        return "Invalid file format"
    else:
        try:
            with open(file_name, 'x') as file:
                return {}
        except FileExistsError:
            try:
                with open(file_name, 'r') as file:
                    return json.load(file)
            except json.JSONDecodeError:
                return {}

def creatorsave():
    """
    <DOCSTRING>
    This function detects if the savedata.json file contains data or not.
    Which decides if the coursemanagement menu will launch or not.
    """
    read_jsonfile("savedata.json")
    if os.path.getsize("savedata.json") == 0:
        createmenu()
    elif os.path.getsize("savedata.json") > 0:
        lmfromfile()

# Functions for Term GPA Computation
def tgpa_calc(impcourse):
    """
    <DOCSTRING>
    This function calculates the term gpa and returns a string with 2 decimal places
    """
    with open("syllabus.json", 'r') as file:
        coursewithunits = json.load(file)

    totalunits = 0
    cwuh = {}
    for course in impcourse.keys():
        for refcourse, units in coursewithunits["course_units"].items():
            if course == refcourse:
                cwuh[refcourse] = units
                totalunits += units

    honorpoints = 0
    for course, gpa in impcourse.items():
        for refcourse, refunits in cwuh.items():
            if course == refcourse:
                if type(gpa) != str:
                    honorpoints += (gpa*refunits)

    result = honorpoints/totalunits
    return str(f"{result:.2f}")

# Functions for Create
def createmenu():

```

```

"""
<DOCSTRING>
This function Launches the Create Menu if no data is detected in the savedata.json
"""

while True:
    clr()
    logo()
    print("\n\n")
    cmloadmenu()

    try:
        userinput = input(">> ")
    except KeyboardInterrupt:
        userinput = " "
    if userinput == "1": # Load Syllabus Template
        cmtemplate()
        break
    elif userinput == "2": # Start from Scratch
        cmbuilder()
        break
    elif userinput == " ":
        continue
    else:
        break

def sybcreate(n):
    """
    <DOCSTRING>
    This function reads the syllabus, iterates through each course,
    and prompts the user to edit the course code and gpa.
    """

    errorhandline4 = 0
    syb = read_jsonfile("syllabus.json")
    coursetmp = {}
    grades = {}
    continueloop = True
    forcequit = False

    sybref = "cpeterm" + str(n)

    for course in syb.get(sybref, []):
        while continueloop == True:
            clr()

            logo()
            print("\n\n")
            sybcourse(course)
            print("\n\n")

            if errorhandline4 == 1:
                print("ERROR: Invalid Course Code\n")
            if errorhandline4 == 2:
                print("ERROR: Invalid GPA\n")

```

```

if errorhandline4 == 3:
    print("Invalid Input")

tmpinput = input(">> ")

if tmpinput == "1":
    inputnc = input("What is the new Course Code?\n>> ")
    if len(inputnc.lower()) != 7:
        errorhandline4 = 1
        continue
    gradeinput = input("\n\nWhat is your GPA?\n>> ")
    try:
        if gradeinput == "P" or gradeinput == "F" or gradeinput == "p" or gradeinput == "f":
            gradeinput = gradeinput.upper()
        elif gradeinput > 0 and gradeinput <= 4:
            gradeinput = float(gradeinput)
        elif gradeinput < 0 and gradeinput > 4:
            errorhandline4 = 2
            continue
    except Exception:
        errorhandline4 = 2
        continue
    grades[inputnc] = gradeinput
    errorhandline4 = 0
    break
elif tmpinput == "2":
    inputnc = course
    gradeinput = input("What is your GPA?\n>> ")
    try:
        if gradeinput == "P" or gradeinput == "F" or gradeinput == "p" or gradeinput == "f":
            gradeinput = " " + gradeinput.upper()
        elif gradeinput:
            gradeinput = float(gradeinput)
        elif gradeinput < 0 and gradeinput > 4:
            errorhandline4 = 2
            continue
    except Exception:
        errorhandline4 = 2
        continue
    grades[inputnc] = gradeinput
    errorhandline4 = 0
    break
elif tmpinput == "exit":
    continueloop = False
    forcequit = True
    break
else:
    errorhandline4 = 3
    continue
if forcequit == True:
    break

coursetmp["grades"] = grades

```

```

coursetmp["term"] = n
with open("savedata.json", "w") as file:
    json.dump(coursetmp, file, indent=2)

```

```
def cmtemplate():
```

```
    """
```

```
    <DOCSTRING>
```

```
    This function compiles all functions for the
    Add from Syllabus Template Feature
```

```
    """
```

```
    while True:
```

```
        clr()
```

```
        logo()
```

```
        print("\n\n")
```

```
        addfromsyb()
```

```
        print("\n\n")
```

```
        terminput = input("\n\n>> ")
```

```
        try:
```

```
            terminput = int(terminput)
```

```
        except Exception:
```

```
            pass
```

```
        if type(terminput) == int:
```

```
            sybcreate(terminput)
```

```
            break
```

```
        elif terminput == "exit":
```

```
            break
```

```
        else:
```

```
            continue
```

```
def cmbuilder():
```

```
    """
```

```
    <DOCSTRING>
```

```
    This function compiles all functions for the
    Add from Scratch Feature
```

```
    """
```

```
    course = {}
```

```
    grades = {}
```

```
    donotcontinue = False
```

```
    while True:
```

```
        clr()
```

```
        logo()
```

```
        print("\n\n")
```

```
        addcourse()
```

```
        print("\n\n")
```

```
        terminput = input("\n\n>> ")
```

```
        try:
```

```
            terminput = int(terminput)
```

```
        except Exception:
```

```
            pass
```

```
        if type(terminput) == int:
```

```
            break
```

```

    if terminput == "exit":
        donotcontinue = True
        break
    else:
        continue

errorhandline3 = 0
while donotcontinue == False:
    clr()
    logo()
    print("\n\n")
    addcourse()
    print("\n\n")
    if errorhandline3 == 1:
        print("\nERROR: Invalid Course Code")
    elif errorhandline3 == 2:
        print("\nERROR: Invalid GPA")
    elif errorhandline3 == 3:
        print("\nCourse Added Successfully\n")

    courseinput = input("Enter course code: ")
    if courseinput.lower() == "exit":
        break
    if len(courseinput.lower()) != 7:
        errorhandline3 = 1
        continue
    gradeinput = input("Input GPA in course: ")
    try:
        if gradeinput == "P" or gradeinput == "F" or gradeinput == "p" or gradeinput == "f":
            gradeinput = gradeinput.upper()
        elif float(gradeinput) > 0 and float(gradeinput) <= 4:
            gradeinput = float(gradeinput)
        elif float(gradeinput) < 0 and float(gradeinput) > 4:
            errorhandline3 = 2
            continue
    except Exception:
        errorhandline3 = 2
        continue

    grades[courseinput.upper()] = gradeinput
    errorhandline3 = 3

if donotcontinue == False:
    course["grades"] = grades
    course["term"] = terminput
    with open("savedata.json", "w") as file:
        json.dump(course, file, indent=2)

# Functions for Load
def lmfromfile():
    """
    <DOCSTRING>
    This function loads the data from savedata.json and launches coursemanagement menu with the info

```



```

This is required to load the coursemanagement menu properly
"""

data = read_jsonfile("savedata.json")
term = data["term"]
course = data["grades"]

coursemanagement(term, course)

# The Course Management Function
def coursemanagement(term, course):
    """
    <DOCSTRING>
    This function is the main highlight of this project.
    """
    while True:
        clr()
        termword = "Term " + str(term)
        tgpa = tgpa_calc(course)

        unitlist = []
        cwu = read_jsonfile("syllabus.json")
        for rkeys in course.keys():
            for ukeys, uvalues in cwu["course_units"].items():
                if rkeys == ukeys:
                    unitlist.append(uvalues)

        qry = coursemanagementmenu(term, termword, course, tgpa, unitlist)
        cm_displayoptions()
        courselist = []
        for courses in course.keys():
            courselist.append(courses)

        usrinput = input(">> ")
        try:
            usrinput = int(usrinput)
        except Exception:
            pass

        # Exit Menu
        if usrinput == "e":
            break

        # Clean Save File
        if usrinput == "c":
            confirmation_deletion = input("Are You Sure? Performing this action can't be undone!\nEnter 'Y' or 'yes' to continue>> ")
            if confirmation_deletion == "Y" or confirmation_deletion == "yes":
                with open("savedata.json", "w") as file:
                    file.truncate(0)
                print("\nSave File Wiped!\n")
                input("<< < Press Enter to Exit this Page >>\n")
                break
            else:

```

```

        pass
import json

# Add Course
if usrinput == "a":
    clr()
    coursemanagementmenu(term, termword, course, tgpa, unitlist)
    cm_addcourse()

    crsaddmenuaction = input(">> ")

    if len(crsaddmenuaction) == 7:
        clr()
        coursemanagementmenu(term, termword, course, tgpa, unitlist)
        cm_addcourse(stage=2)

        crsaddgpaction = input(">> ")
        try:
            keys = crsaddmenuaction.upper()
            if crsaddgpaction == "P" or crsaddgpaction == "F" or crsaddgpaction == "p" or crsaddgpaction == "f":
                values = " " + crsaddgpaction.upper()
            elif float(crsaddgpaction) > 0 and float(crsaddgpaction) <= 4:
                values = float(crsaddgpaction)
            modifydata = read_jsonfile("savedata.json")
            modifydata["grades"][keys] = values

            with open("savedata.json", "w") as file:
                json.dump(modifydata, file, indent=2)

            course = modifydata["grades"]
        except Exception:
            pass

# Remove Course
if usrinput == "r":
    clr()
    coursemanagementmenu(term, termword, course, tgpa, unitlist)
    cm_deletecourse()

    crsaction = input(">> ")

    try:
        if int(crsaction) > 0 and int(crsaction) < int(qry):
            selectedcourse = courselist[int(crsaction)-1]

            modifydata = read_jsonfile("savedata.json")
            newgrades = {}
            for keys, values in modifydata["grades"].items():
                if keys == str(selectedcourse):
                    pass
                else:
                    newgrades[keys] = values

```

```

        modifydata["grades"] = newgrades
        course = modifydata["grades"]

        with open("savedata.json", "w") as file:
            json.dump(modifydata, file, indent=2)
except Exception:
    pass

# Modify Courses in Menu
try:
    if int(usrinput) > 0 and int(usrinput) < int(qry):
        selectedcourse = courselist[int(usrinput)-1]

        clr()
        coursemanagementmenu(term, termword, course, tgpa, unitlist)
        cm_editcourse(selectedcourse)
        crsaction = input(">> ")
        if crsaction == "1":
            clr()
            coursemanagementmenu(term, termword, course, tgpa, unitlist)
            cm_editcourse(selectedcourse, mode=2)
            crsaddaction = input(">> ")
            if type(crsaddaction) == str:
                if len(crsaddaction) == 7:
                    modifydata = read_jsonfile("savedata.json")

                    newgrades = {}
                    for keys, values in modifydata["grades"].items():
                        if keys == str(selectedcourse):
                            newgrades[str(crsaddaction)] = values
                        else:
                            newgrades[keys] = values

                    modifydata["grades"] = newgrades
                    course = modifydata["grades"]

                    with open("savedata.json", "w") as file:
                        json.dump(modifydata, file, indent=2)

        if crsaction == "2":
            clr()
            coursemanagementmenu(term, termword, course, tgpa, unitlist)
            cm_editcourse(selectedcourse, mode=3)
            gpaaddaction = input(">> ")

            try:
                if gpaaddaction == "P" or gpaaddaction == "F" or gpaaddaction == "p" or gpaaddaction == "f":
                    gpaaddaction = " " + gpaaddaction.upper()
                elif float(gpaaddaction) > 0 and float(gpaaddaction) <= 4:
                    gpaaddaction = float(gpaaddaction)
            except Exception:
                pass

```

```

modifydata = read_jsonfile("savedata.json")
try:
    newgrades = {}
    for keys, values in modifydata["grades"].items():
        if keys == str(selectedcourse):
            newgrades[keys] = gpaaddaction
        else:
            newgrades[keys] = values

    modifydata["grades"] = newgrades
    course = modifydata["grades"]

    with open("savedata.json", "w") as file:
        json.dump(modifydata, file, indent=2)
except Exception:
    pass
else:
    pass
except Exception:
    pass
else:
    pass

```