الجامعة الإسلامية العالمية شيتاغونغ

International Islamic University Chittagong

Department of Computer and Communication Engineering (CCE)
Course Title : Python Programming Sessional
Course Code : CCE-2406

Project Name:

CLASSMATE

Student Class Schedule Manager

TEAM MEMBERS

- Mikdad Mohammad

  Id: E241006
- Mohammad Tasnimul Hasan Mahir

  Id: E241001
- Golam Mostafa Naim

  Id: E241034

INTRODUCTION

ClassMate is a console-based student class schedule management system developed using Python.

The application is designed to help students organize and manage their daily academic routines in a simple and efficient way.

It provides essential features such as adding new classes, viewing today's and tomorrow's schedules, and deleting existing class entries
The project follows a modular programming approach and uses file handling techniques with JSON to ensure persistent data storage.
ClassMate was developed as part of the Python Programming final examination, with the objective of applying core Python concepts to solve a real-world student problem

PROBLEM STATEMENT

- Students often forget daily class schedules
- Manual routine management is inefficient
- No simple offline tool for managing class routines
- Need for a lightweight and easy-to-use solution

OBJECTIVE

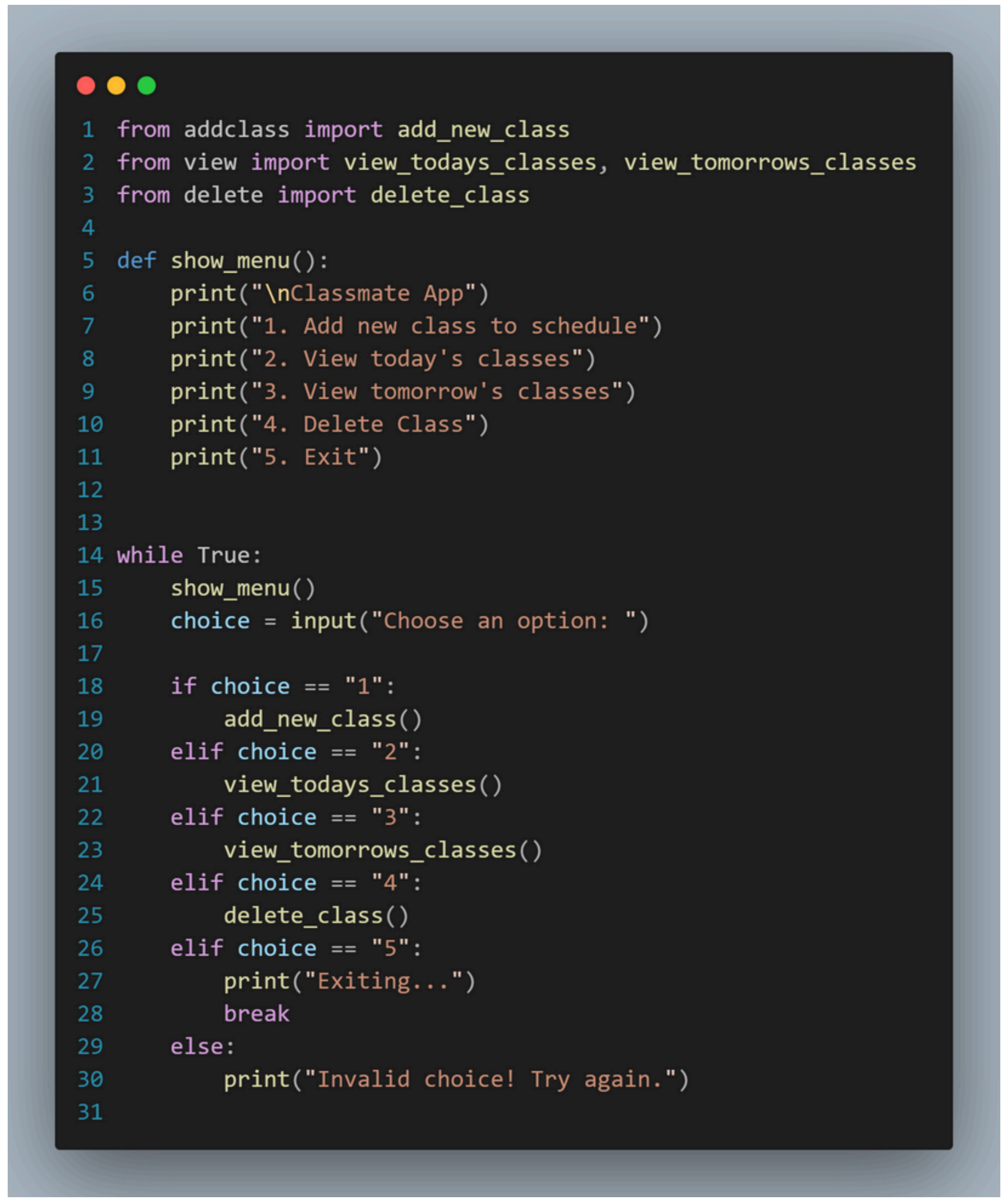
- Develop a console-based class schedule manager
- Apply core Python programming concepts
- Practice file handling and modular programming
- Provide an easy routine management system for students

KEY FEATURES

- Add new class to schedule
- View today’s classes
- View tomorrow’s classes
- Delete a class from schedule
- Automatic data saving
- Simple text-based menu system

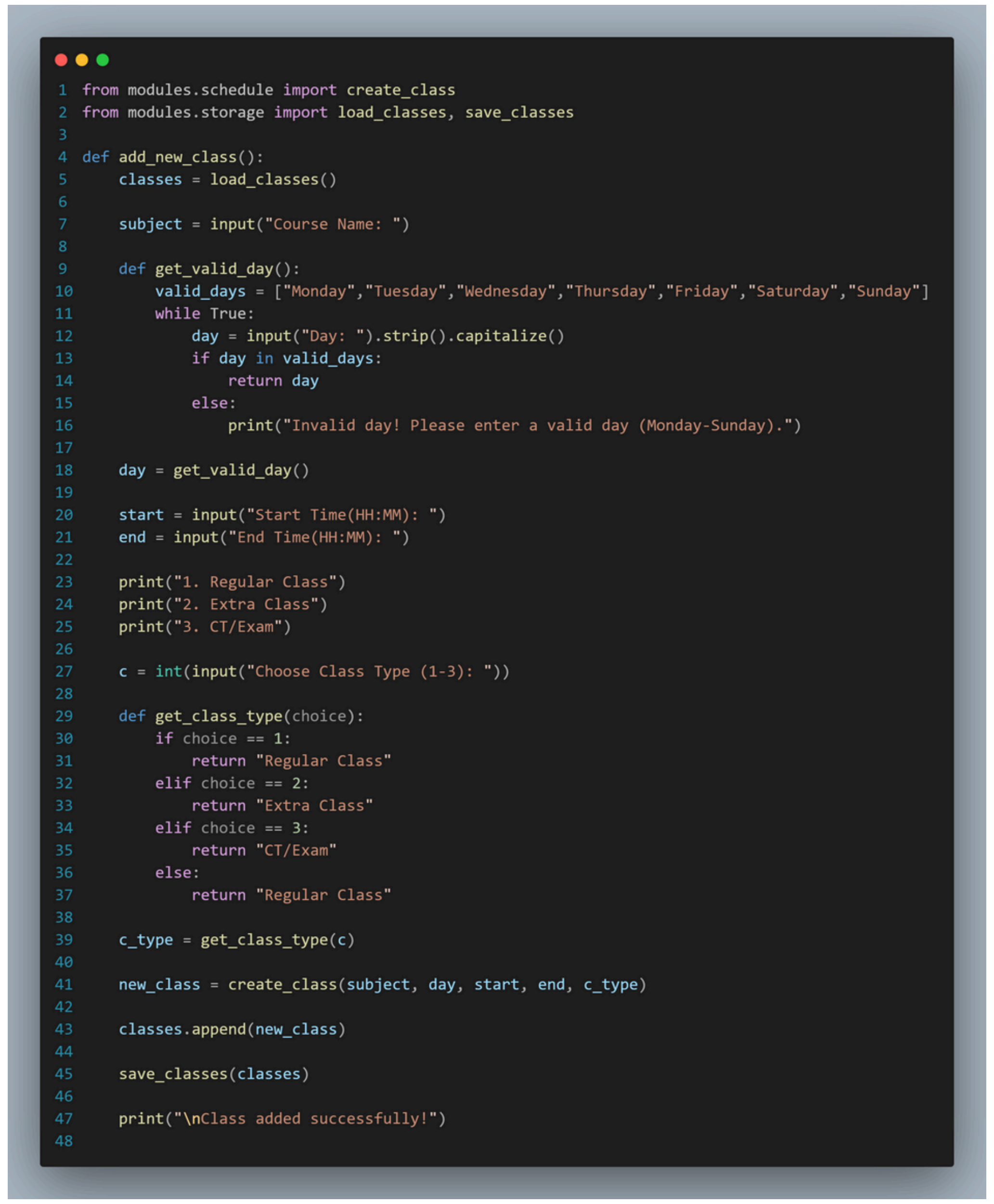
PROJECT STRUCTURE

main.py



```python
from addclass import add_new_class
from view import view_todays_classes, view_tomorrows_classes
from delete import delete_class

def show_menu():
    print("\nClassmate App")
    print("1. Add new class to schedule")
    print("2. View today's classes")
    print("3. View tomorrow's classes")
    print("4. Delete Class")
    print("5. Exit")


while True:
    show_menu()
    choice = input("Choose an option: ")

    if choice == "1":
        add_new_class()
    elif choice == "2":
        view_todays_classes()
    elif choice == "3":
        view_tomorrows_classes()
    elif choice == "4":
        delete_class()
    elif choice == "5":
        print("Exiting...")
        break
    else:
        print("Invalid choice! Try again.")

```

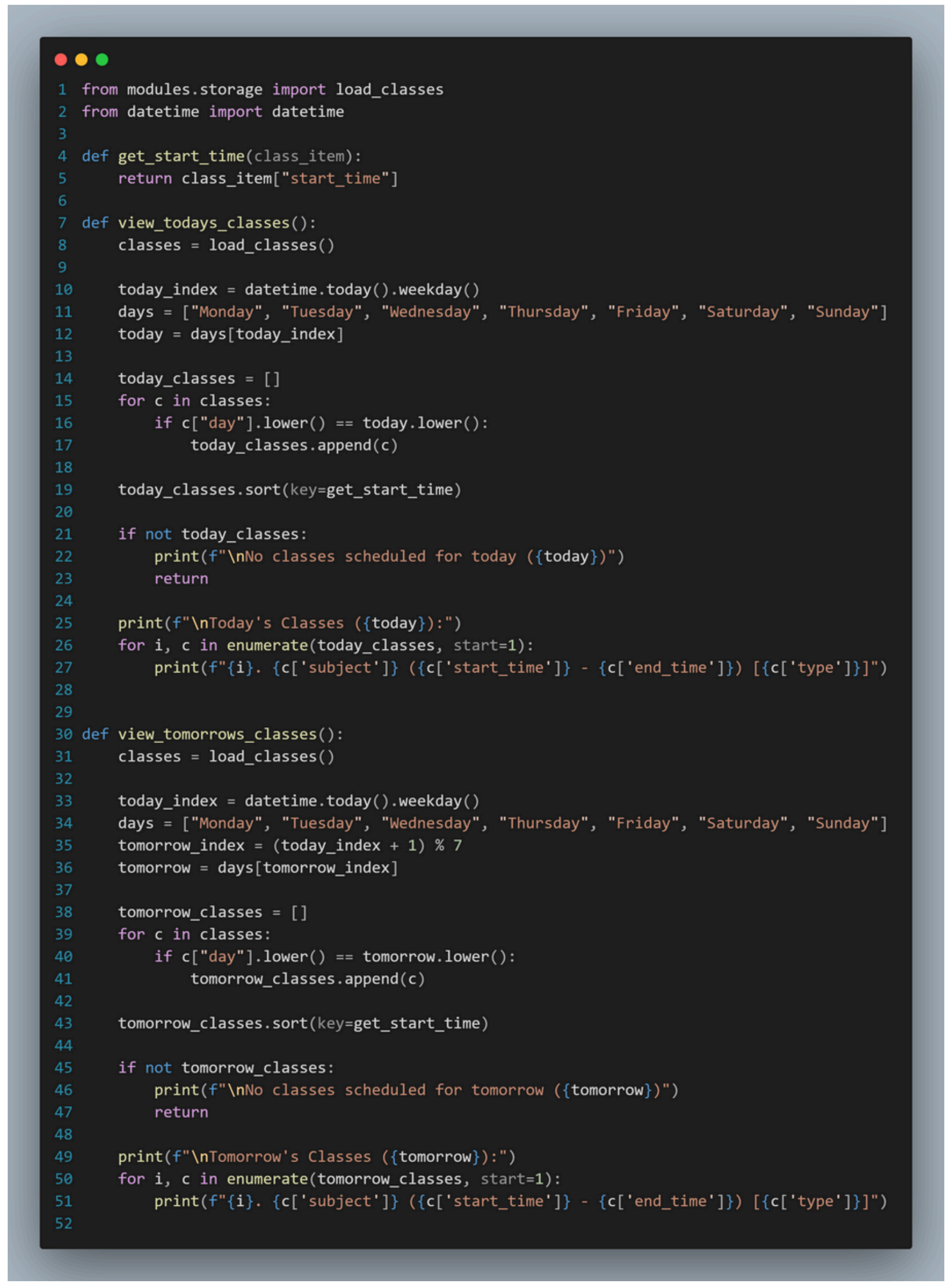
addclass.py



```python
from modules.schedule import create_class
from modules.storage import load_classes, save_classes

def add_new_class():
    classes = load_classes()

    subject = input("Course Name: ")

    def get_valid_day():
        valid_days = ["Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday"]
        while True:
            day = input("Day: ").strip().capitalize()
            if day in valid_days:
                return day
            else:
                print("Invalid day! Please enter a valid day (Monday-Sunday).")

    day = get_valid_day()

    start = input("Start Time(HH:MM): ")
    end = input("End Time(HH:MM): ")

    print("1. Regular Class")
    print("2. Extra Class")
    print("3. CT/Exam")

    c = int(input("Choose Class Type (1-3): "))

    def get_class_type(choice):
        if choice == 1:
            return "Regular Class"
        elif choice == 2:
            return "Extra Class"
        elif choice == 3:
            return "CT/Exam"
        else:
            return "Regular Class"

    c_type = get_class_type(c)

    new_class = create_class(subject, day, start, end, c_type)

    classes.append(new_class)

    save_classes(classes)

    print("\nClass added successfully!")

```

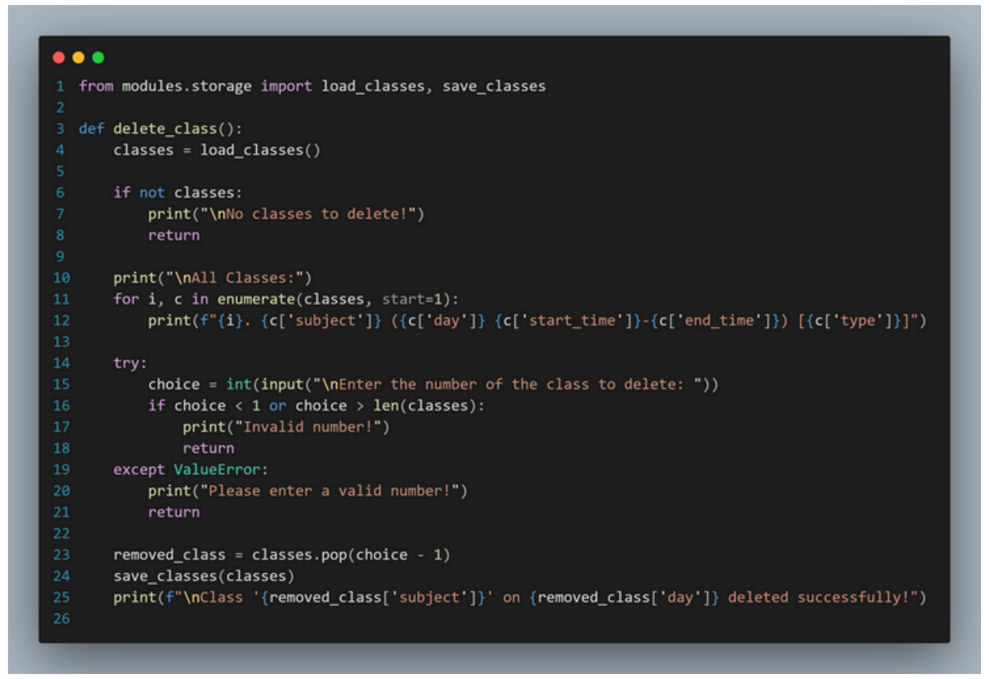
PROJECT STRUCTURE

view.py



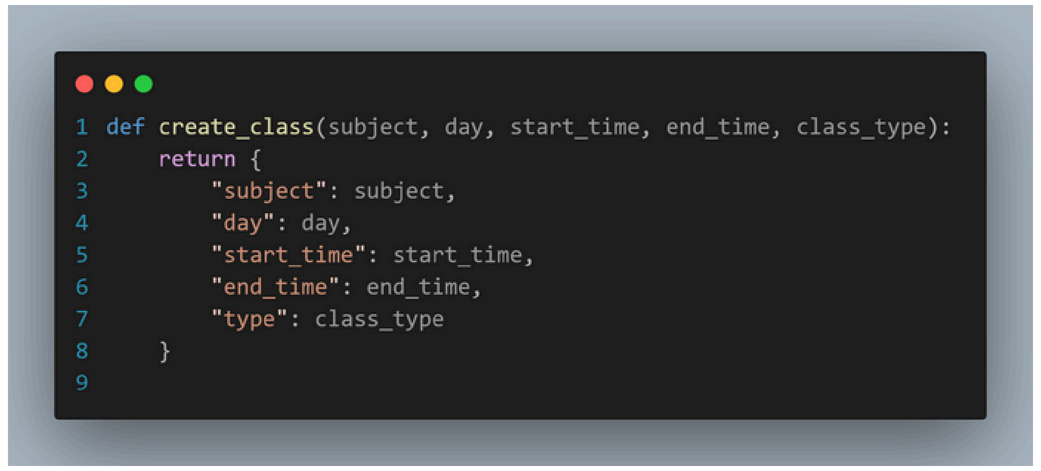
```python
from modules.storage import load_classes
from datetime import datetime

def get_start_time(class_item):
    return class_item["start_time"]

def view_todays_classes():
    classes = load_classes()

    today_index = datetime.today().weekday()
    days = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]
    today = days[today_index]

    today_classes = []
    for c in classes:
        if c["day"].lower() == today.lower():
            today_classes.append(c)

    today_classes.sort(key=get_start_time)

    if not today_classes:
        print(f"\nNo classes scheduled for today ({today})")
        return

    print(f"\nToday's Classes ({today}):")
    for i, c in enumerate(today_classes, start=1):
        print(f"{i}. {c['subject']} ({c['start_time']} - {c['end_time']}) [{c['type']}]")


def view_tomorrows_classes():
    classes = load_classes()

    today_index = datetime.today().weekday()
    days = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]
    tomorrow_index = (today_index + 1) % 7
    tomorrow = days[tomorrow_index]

    tomorrow_classes = []
    for c in classes:
        if c["day"].lower() == tomorrow.lower():
            tomorrow_classes.append(c)

    tomorrow_classes.sort(key=get_start_time)

    if not tomorrow_classes:
        print(f"\nNo classes scheduled for tomorrow ({tomorrow})")
        return

    print(f"\nTomorrow's Classes ({tomorrow}):")
    for i, c in enumerate(tomorrow_classes, start=1):
        print(f"{i}. {c['subject']} ({c['start_time']} - {c['end_time']}) [{c['type']}]")

```

delete.py



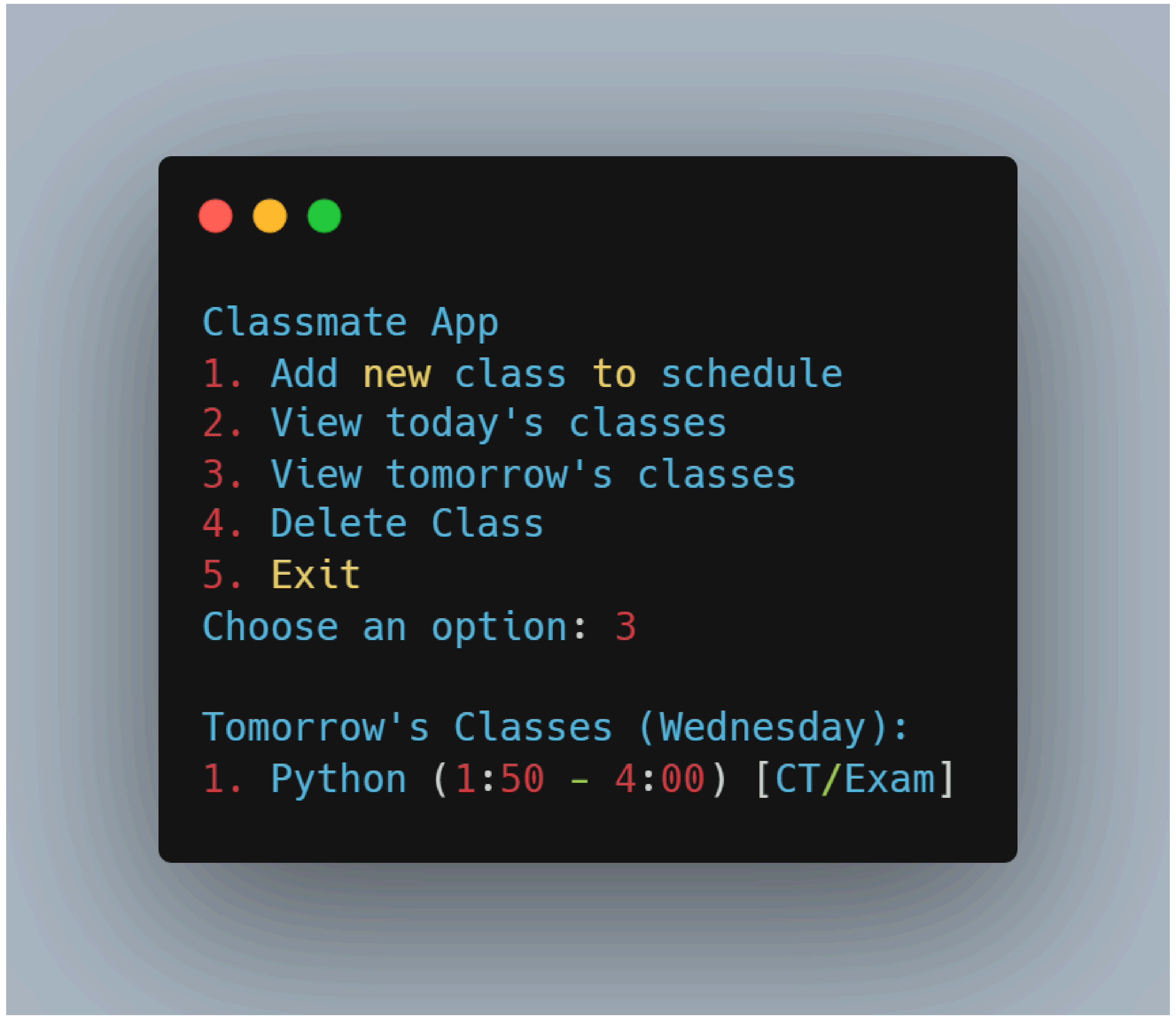
```python
from modules.storage import load_classes, save_classes

def delete_class():
    classes = load_classes()

    if not classes:
        print("\nNo classes to delete!")
        return

    print("\nAll Classes:")
    for i, c in enumerate(classes, start=1):
        print(f"{i}. {c['subject']} ({c['day']} {c['start_time']}-{c['end_time']}) [{c['type']}]")

    try:
        choice = int(input("\nEnter the number of the class to delete: "))
        if choice < 1 or choice > len(classes):
            print("Invalid number!")
            return
    except ValueError:
        print("Please enter a valid number!")
        return

    removed_class = classes.pop(choice - 1)
    save_classes(classes)
    print(f"\nClass '{removed_class['subject']}' on {removed_class['day']} deleted successfully!")

```

schedule.py



```python
def create_class(subject, day, start_time, end_time, class_type):
    return {
        "subject": subject,
        "day": day,
        "start_time": start_time,
        "end_time": end_time,
        "type": class_type
    }

```

RESULT



```
Classmate App
1. Add new class to schedule
2. View today's classes
3. View tomorrow's classes
4. Delete Class
5. Exit
Choose an option: 3

Tomorrow's Classes (Wednesday):
1. Python (1:50 - 4:00) [CT/Exam]
```

GITHUB REPOSITORY & CODESPACES

- Project source code is hosted on GitHub
- GitHub Codespaces is used as a cloud development environment
- Allows running and testing the project directly from the browser
- No local Python setup required
- Helps in easy testing and collaboration
- Public repository with full documentation
- Includes README and modular code structure

Scan to view project source code

https://github.com/Mikdad-12/Classmate

FUTURE IMPROVEMENTS

- Convert to GUI application
- Web-based version using Flask
- Android APK version
- Add notification/reminder system
- Multi-user support

CONCLUSION

ClassMate is a simple and effective Python-based console application designed to help students manage their daily class schedules efficiently.

Through this project, we successfully implemented core Python concepts such as modular programming, file handling and menu-driven logic using real-world requirements.

The application provides essential features like adding, viewing and deleting class schedules while ensuring persistent data storage using JSON.

This project fulfilled the objectives of the Python Programming course and enhanced our understanding of practical software development.

In the future, the project can be further improved by converting it into a GUI application, developing a web-based version using Flask, creating an Android APK, adding a notification or reminder system, and supporting multiple users to make it more scalable and user-friendly.

THANK YOU

For your attention