ACIT4420 Assignment II

1.0 Introduction

`study_reminders` is a lightweight Python package designed to serve as a reminder system for students. This report outlines its design architecture, structure, and any implementation challenges met during the project.

2.0 Package design

The package is composed of five primary modules

Name	Description
`reminder_generator`	Provides a `generate()` function for the creation of reminders for students.
`reminder_logger`	Logs reminder messages with timestamps to a file.
`reminder_sender`	Simulates sending reminders by printing them to the console.
`reminder_scheduler`	Schedules and runs daily reminders at specified times using the `schedule` package.
`student_manager`	Manages student data: load, save, add, remove, list.

The user of this package can interact with these modules either through the `main.py` script, which comes with a simple interactive command-line interface, or through their own implementation.

2.1 Example code snippets

Example of the menu implementation in `main.py`:

```
def main_menu(student_manager: StudentsManager):
    while True:
        print("\nMAIN MENU")
        print("1. Manage students")
        print("2. Run scheduler")
        print("3. Run tests")
        print("4. Exit")

        choice = input("Select an option (1-4 or CTRL-C to Exit):")

        if choice == "1":
            student_menu(student_manager)
        elif choice == "2":
            scheduler_menu(student_manager)
        elif choice == "3":
            run_tests()
        elif choice == "4":
            sys.exit(0)
```

Each sub-menu is its own function with a `while True` loop so the system does not terminate, and the user can keep navigating through the interface.

3.0 Challenges

One of the bigger challenges was managing the python environment through `venv`. This was the first time working with `venv`, and the goal was not to install any packages globally on the system.

4.0 Discussion

I am personally happy with the resulting package. I feel like, even though most of the implementation was already provided, I could still add onto it and make it my own. I was especially happy with the interactive menu for testing the modules.

There are still some improvements that could be made. Adding tests, for example, could help stop regressions and future-proof the code. Python package structure is also quite confusing, and I am certain there are some choices that could have improved the end result.

5.0 Environment and tools

- **Language**: Python
- **External packages**: `schedule`, `pytz`
- **Documentation**: Typst