Final Report: C.H.I.E.T

COMP1080SEF

Group Members:

Albin [13872558]
Maheen [13766356]
Yunchho [13880269]
Rohan [13511636]
Thomas [13877806]

Introducing:
C.H.I.E.T

Start

- **C**lassroom Attendance Tracker
- **H**omework Assignment Tracker
- **I**nteractive Math Quiz
- **E**ducational Learning
- **T**imetable Scheduler
- Budget Planner

Introduction:

Background:

In today's academic environment, student face numerous challenges regarding organization and time management. With heavy workloads and multiple responsibilities, students often struggle to keep track of attendance, assignments, quizzes, schedules, and personal finances. A comprehensive tool like **C.H.I.E.T** is essential to help students help manage these aspects effectively. **C.H.I.E.T** is a command-line application designed to assist students in managing their academic and financial responsibilities.

Problem Statement:

C.H.I.E.T addresses several critical problems in students life:

Classroom Attendance Tracker (C): As mentioned in our proposal existing solutions are mostly for teachers and it may involve complex features that are not necessary for simple attendance tracking.
In order to address this problem, **(C)** Classroom attendance tracker subpage is developed by python and works on CLI. It allows students to input and the information about the class that they've attended.

Homework Assignment Tracker (H): This is a program created to help students from secondary to university grade disciplines enhances their productivity. Through the usage of the app, we hope to decrease the tardiness of students who wish to do better in school but are disadvantaged by having too many things to do. This program was created with help from Youtube and Stackoverflow.

Interactive Math Quiz (I): This math's quiz is a short quiz competition specially targeted to high school students. This program may really come in handy to practice and recall student's math's skills to increase their IQ level and preparing students for their examination. It creates an opportunity to get more involved in this subject .

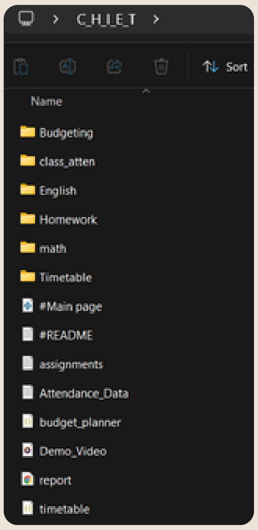
Educational Learning (E): Educational English learning program are for those students who needs help to improve their English vocabulary and grammar. By integrating features such as quizzes and practices.
Also, It offers the convenience of studying anytime and anywhere. Students who want to enhance their knowledge in English language in a fun and efficient way can use this program.

Timetable Scheduler (T): In the fast-paced environment of student life, managing time effectively is crucial. The Timetable Management system is designed to assist students in organizing their weekly schedules by allowing them to add, edit, display, and delete classes or events. This program not only simplifies the process of timetable management but also ensures that students can keep track of their academic commitments efficiently.

Budget Planner: Managing finances effectively is crucial, especially for students who are teenagers they often juggle multiple responsibilities. This application addresses the real-life problem of financial management by providing a user-friendly interface to track their income and expenses. This report outlines the functionality of the application, its implementation, and the methodologies employed to create a robust budgeting tool.

preview.jpg

- After downloading the file make sure to extract it.



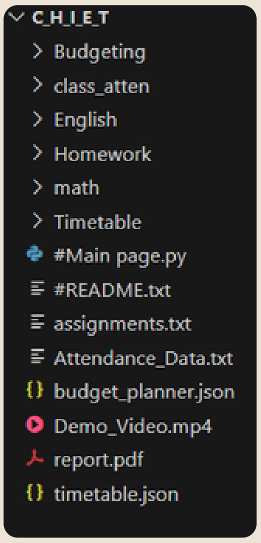
C_H_I_E_T >
Sort
Name
Budgeting
class_atten
English
Homework
math
Timetable
#Main page
#README
assignments
Attendance_Data
budget_planner
Demo_Video
report
timetable

- There should be 14 files in side the folder.
- Our group has made a README (md) file, describing how our code should be tested and we have provided several test cases. It also includes the suitable platform to run the code without any problems. eg:

Visual Studio Code python

we suggested users to use VS code to run our program and we also mentioned that python version 3.12.7 64-bit is really efficient while running our program.

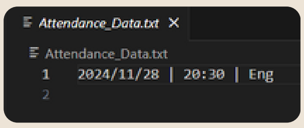
- We also have a Demo_Video showing how the program can be executed



C_H_I_E_T
Budgeting
class_atten
English
Homework
math
Timetable
#Main page.py
#README.txt
assignments.txt
Attendance_Data.txt
budget_planner.json
Demo_Video.mp4
report.pdf
timetable.json

- In order to test our program we want users to run the program from the Main page.py.
- There are 4 text files outside which are related to assignment tracker, attendance, timetable and budget planner.
- There are 2 text files math.txt and english.txt in their respective folder.
- These text files shows the users input on the particular program

eg:

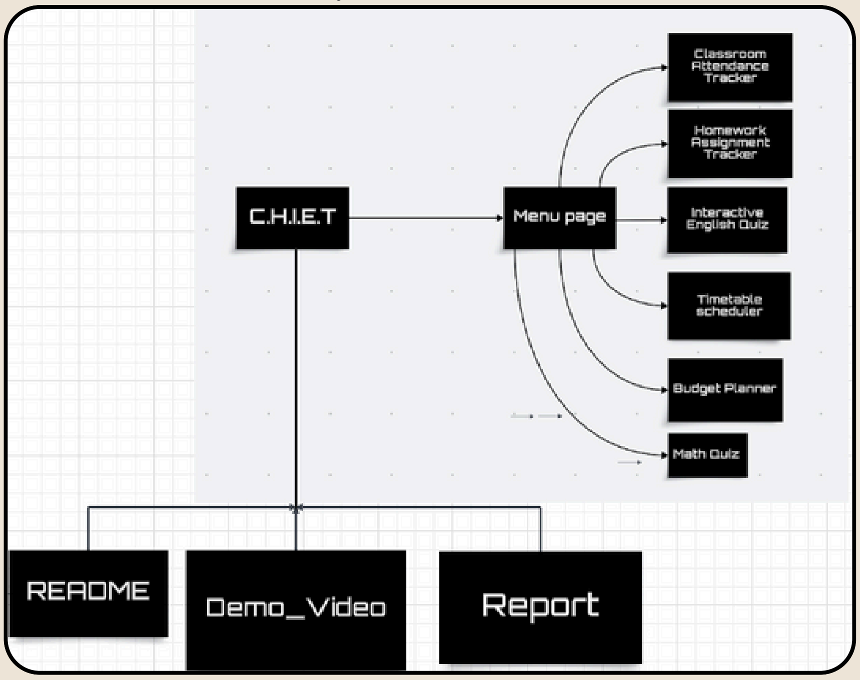


Attendance_Data.txt

```
Attendance_Data.txt
2024/11/28 | 20:30 | Eng
```

Methodology:

System Block



Classroom Attendance Tracker:

The attendance tracker contains seven (including main menu function) and imports two libraries (OS and DATETIME). There are two functions used to load a data file from an external data file. The OS library used in load_record() function for inputting the data in that file into a dictionary for the afterward editing. The DATETIME library is used for the input validation in take_attendance().

To run the program:

- When you start the program, it will automatically create a variable that contain a string that is equal to the file name that store the previous data.(if that data file doesn't exists. The code will automatically create one.
- For the take_attendance(), the user should enter the date and time in the correct format. The code will use try function to check is that the entered date invalid or not. After the user entered all of the data, the function save_data() will automatically run and save the data in to the data file.
- For the display_attendance(), I use an if else statement to create a search function that allows user to search for the subject they want to check or they can just enter "no" to view all of the record. The record will be show by a for loop.
- For the edit_attendance (), the user can use this function to delete the record that is wrong. The deleted record will not be show in the temporary dictionary and the data file.
- For the clear_attendance(), the user can clear all record that record in the data file or the temporary dictation.

Limitation:

1.In the "take_attendance()" function the user can't escape from the function back to the main menu. If the user accidentally enter into the take_attendance() function.

2.Also in the "take_attendance()" function, I didn't make a time validation that make sure the time are not larger than the current time. The user

Homework Assignment Tracker:

The assignment tracking app consists of the main part of the code and a text file. It has various functions including, [IPO] for inputting assignments into the text file for storing and output for listing out the assignments stored. If and while loops to check if there is information on a specific line. Try and valueError to make sure if the user inputs an invalid option they don't get exited out the program and delete their stored information. Additionally, we've used a text file to store, edit and delete the data on specific lines.

HOW IT WORKS

- Press the run button, you will be greeted with a Menu of 5 options.
- Type either 1/2/3/4/5 into the CLI then press enter to access add assignments, view assignments, remove assignments, mark assignments as complete, exit
- If you press options 1,2,3 or 4 type your answer into the CLI respectively then hit Enter. After you finish typing out all the information in whatever option you pick from 1-4. The program will go back to the main menu.

Reflection:

The limitation of this program is although we have tried our best to account for possible failures or errors in the code. There is still potential for the user to make errors. Common examples include typing out wrong information. They may accidentally write the description when the program asks to type the due date. Adding a check format would have been able to make sure that we can avoid such human errors.

Future work:

Addition of a function that will tell the user's when assignments are about to reach their due date a couple days in advance. This will tell the user what to prioritize so they have a much lower chance of missing deadlines.
Implement functions that allow the user to add grade/marks of assignments, tests, group projects, etc. and get the Grade Point Average (GPA). This can help evaluate which subjects they are falling behind on and what they need to improve.

Interactive Math Quiz:

The methodology used in this code is straightforward. It combines elements of game design (quiz), user interaction (input/output), randomization (import random), and file I/O to provide a user-friendly math quiz application.

To Run the program:

- To run the program, you will see a welcome message and the first question printed on the screen.
- To answer a question, type your answer and press Enter.
- If you want to exit the quiz at any time, type exit and hit Enter.
- After going through the questions, the program will read your previous score from math_score.txt and compare your performance.
- It will then write your current score back to math_score.txt for the next time you run the quiz.

Educational Learning (English Quiz)

This Educational English learning program consist of 3 sub-topics (Grammer, Vocabulary, Riddles). This are build using (input/output for interaction, randomization (import random), While-loop and Boolean for looping the program if input is invalid, from subproce import call function for connecting python file, define-function fo labeling parts, import sys and break function to break of the program, json file to store data, txt file to store score, library to store question and answer.

To Run the Program:

- Run the program (VS code) Press the run, you will see a welcome message and read the text and follow the instructio written in the text.

TimeTableScheduler:

This program utilizes a JSON file to store timetable data, ensuring that the information persists between sessions.

Key Components
- Data Storage: The program uses JSON for structured storage of timetable information. This format allows for easy reading and writing of data.
- Functions: The program is modular, with distinct functions for each operation, including:
  - load_timetable(): Loads the timetable from a JSON file.
  - save_timetable(timetable): Saves the current timetable to a JSON file.
  - add_class(timetable): Adds a new class/event to the timetable.
  - display_classes(timetable): Displays all scheduled classes/events.
  - edit_class(timetable): Edits an existing class/event.
  - delete_class(timetable): Deletes a specified class/event.
- User Interaction: The program interacts with users through the console, prompting for input to perform various operations.

Programming Techniques
- File I/O: The program reads from and writes to a JSON file to manage data persistence.
- Conditional Structures: Used to ensure correct user inputs and to validate data before processing.
- Repetition Structures: A loop is utilized to continuously display the menu until the user decides to exit.
- Data Structures: The timetable is stored as a dictionary, where each key represents a day of the week, and each value is a list of events for that day.

Steps to Run the Program;
1. Environment Setup:
- Ensure you have Python installed on your computer. You can download it from python.org.
- Make sure to have a text editor or an Integrated Development Environment (IDE) like PyCharm, Visual Studio Code, or IDLE to edit and run the Python script.

2. run the program from the main.py file choose 6

3. Once the program is running, you will see a menu with options to add, display, edit, or delete classes/events.
- Follow the prompts to input the required information. For example, to add a class, you will need to enter the day of the week, the subject/event name, and the time.
- To exit the program, choose the exit option from the menu.

Limitation:
Data Persistence: The program relies on a single JSON file (timetable.json) for storing timetable data. If this file is accidentally deleted or corrupted, all user data could be lost. Additionally, concurrent access is not supported; if multiple instances of the program are run simultaneously, it may lead to data inconsistencies.

User Input Validation: While the program includes basic input validation, it does not comprehensively handle all potential user errors. For example, if a user enters a day that does not conform to expected formats (e.g., "Monday" vs. "mon"), the program may fail to recognize it, leading to confusion.

Future Work:

- Enhanced User Interface: Developing a GUI using libraries such as Tkinter or PyQt could significantly improve user experience, making it more intuitive and accessible.

Budget Planner:

This Budget Planner is designed to help users manage their finances by allowing them to add, edit, view, and delete income and expenses. The program uses a JSON file for data storage, enabling easy data retrieval and persistence between sessions.

Key Components
- Data Storage: Budget information is stored in a JSON file (budget_planner.json). This format allows easy manipulation and storage of complex data types like dictionaries and lists.
- Functions: The program is modular, consisting of distinct functions for each operation:
  - load_budget(): Loads existing budget data from a JSON file.
  - save_budget(budget): Saves the current budget data to a JSON file.
  - add_income(budget): Adds a new income entry to the budget.
  - edit_income(budget): Edits the existing income amount.
  - add_expense(budget): Adds a new expense entry to the budget.
  - view_balance(budget): Calculates and displays the current balance.
  - view_expenses(budget): Displays all recorded expenses.
  - edit_expense(budget): Edits an existing expense entry.
  - delete_expense(budget): Deletes a specified expense entry.
- User Interaction: The program interacts with users via a console menu, prompting them for inputs to perform various operations.

Programming Techniques
- File I/O: The program implements reading from and writing to a JSON file to manage data persistence.
- Conditional Structures: Used for input validation and to ensure that operations are performed only on valid data.
- Repetition Structures: A loop is utilized to continuously display the menu until the user decides to exit.
- Data Structures: The budget is managed as a dictionary, with keys for income and a list of expenses.

the steps to run is similar to TimeTableScheduler.py.

Limitation:
- Limited Functionality: The current features are focused on basic income and expense management. Advanced functionalities, such as budget forecasting, categorization of expenses, or financial reporting, are not included, which could limit the program's usefulness for users looking for more detailed financial insights.

Future Work:
- Search and Filter Capabilities: Adding functionality to search for specific expenses or filter them by category

Declaration:

- Youtube tutorials were mostly used:

Brocode, Programming with mosh, Corey Schafer

and some other tutorials as well :
https://youtu.be/cwDP9m3XeLs?si=W9tIPg1Z6I0YFrMN
https://youtu.be/Uh2ebFW8OYM?si=Y5VcUFHsamT2Vgyg
https://youtu.be/Zp5MuPOtsSY?si=GY7Bamhh2CGCob
https://youtu.be/94UHCEmprCY?si=0Nog5m3LH04wL-vW
https://youtu.be/tdjDQ-uVjR0?si=fskX-KPXaaGnNop6

Work Allocation:

- Albin - 20%
- Maheen - 20%
- Yunchho - 20%
- Rohan - 20%
- Thomas - 20%