

Computer Science Project File

PyDo - Command-line Task Manager

2023-2024

ASIAN INTERNATIONAL PRIVATE SCHOOL
Ruwais, Abu Dhabi, UAE



NAME: ADABALA SRIDHAR

GRADE: XII

REGISTER NUMBER:

GUIDED BY: Ms. RESHMA PREMARAJAN

ASIAN INTERNATIONAL PRIVATE SCHOOL-RUWAIS



CERTIFICATE

This is to certify that Miss/Master ADABALA SRIDHAR of Grade **12** has carried out and completed the project work in (Python and SQL connectivity) prescribed by the CBSE, New Delhi during the academic year 2023-24.

Teacher –in-charge :

Date:

Principal:

School Seal:

Table of Contents

SL NO:	TOPICS
1.	Acknowledgement
2.	Objective
3.	Abstract
4.	Packages Used
5.	Files Generated
6.	Methods Used
7.	Source Code
8.	Output Screens
9.	Limitations
10.	Requirements
11.	Bibliography

Acknowledgement

In the accomplishment of this project, many people have bestowed upon their blessings and their heart pledged support. Primarily I thank God Almighty for being able to complete this project with success. Then I would like to thank the management, my Principal Mr. Anzar Abdul Salam and my Computer Science teacher Ms. Reshma Premarajan whose valuable guidance and support has helped me bring out this project. Their suggestions and instructions have served me towards the completion of this project. I would also like to thank my parents and friends for encouraging me during the various phases of this project. Finally, I would like to thank CBSE for giving me this opportunity to undertake this project.

Objective

PyDo is a feature-rich command-line task manager application built using Python. It enables users to efficiently manage tasks, track performance, and redeem rewards for completed tasks. The application also incorporates a child and parent account system, allowing parents to oversee and manage tasks and rewards for their children.

Abstract:

Task Management System with Parental Controls

The "Task Management System with Parental Controls" is designed to provide a comprehensive and interactive solution for children to manage their tasks effectively while allowing parents to actively participate in the process through a dedicated Parental Control system.

Key Components:

1. User Accounts:

- **Child Accounts:** Children create accounts to manage their tasks, set rewards, and earn points for completed activities.
- **Parent Accounts:** Parents have additional functionalities, including overseeing their child's tasks, adding rewards, and adjusting points.

2. Task Management

Users can add, edit, and delete tasks, each with details such as due date, points, and status (Complete, Incomplete, Overdue).

Tasks are automatically categorized as "Overdue" if the due date has passed.

3. Points and Rewards:

Tasks are associated with points, providing a reward system for completed activities.

Children can redeem points for rewards they set, motivating them to accomplish tasks.

4. Parental Oversight:

Parents can view their child's task performance, completed tasks, and overdue tasks.

The Parent Account enables active involvement in the reward system, including adding or deducting points based on behavior and task completion.

5. Rewards System:

Children can set rewards for accomplishing tasks, enhancing the gamification aspect.

Parents can review and manage rewards, fostering a collaborative approach between parents and children.

6. Penalty System:

Overdue tasks result in a penalty, deducting points (e.g., -10) to encourage timely completion.

Workflow:

Child's Perspective:

Create tasks with due dates, points, and status.

Earn points for completed tasks.

Set rewards and redeem points for incentives.

Parent's Perspective:

View the child's tasks, completed tasks, and overdue tasks.

Add rewards to motivate the child.

Adjust points based on behavior and task completion.

Benefits:

Encourages responsibility and time management in children.

Fosters collaboration between parents and children.

Provides an engaging and gamified approach to task management.

This system, through its child-centric design and parental controls, aims to create a positive and collaborative environment for effective task management and personal development.

Packages Used

1. *'mysql.connector' (SQLite Database Interaction):*

Purpose: The `mysql.connector` package is employed for managing user-related data and task information in a relational database. MySQL is a lightweight, serverless database engine, making it suitable for local storage within the application.

2. *'csv' (CSV File Handling):*

Purpose: The `csv` package is used for reading from and writing to CSV files. In this project, it handles the storage and retrieval of user data. The CSV format is commonly used for its simplicity and compatibility with various applications.

3. *'matplotlib.pyplot' (Data Visualization):*

Purpose: The `matplotlib.pyplot` module, part of the Matplotlib library, is utilized for creating visual representations of task performance. It generates bar charts to provide a graphical overview of completed, overdue, and incomplete tasks.

4. *'tabulate' (Tabular Data Formatting):*

Purpose: The `tabulate` package is used for formatting and presenting tabular data in a readable manner. In this project, it aids in displaying task information in a structured table format, enhancing user readability.

5. *'time' (Time Management):*

Purpose: The `time` module introduces time-related delays in the code execution. This feature is useful for controlling the pace at

which messages and prompts are displayed to the user, creating a more user-friendly interaction.

6. *'random'* (Random Number Generation):

Purpose: The random module is employed for generating random numbers. Specifically, it is used to create a new task ID when a task goes overdue. This helps in ensuring uniqueness and unpredictability in task identification.

7. *'datetime'* (Date and Time Operations):

Purpose: The datetime module provides functions for working with dates and times. It is essential for managing task due dates and status. The module facilitates comparisons between current dates and due dates, enabling the system to identify overdue tasks.

These packages collectively contribute to the robust functionality of the Task Management System, encompassing data storage, visualization, user interaction, randomization, and efficient date/time management. The combination of these modules enhances the overall user experience and system reliability.

Files Generated

1. *main.py*:

This Python script serves as the main program file containing the code for the Task Management System.

2. *dat.csv*:

This CSV file stores user data, including usernames and corresponding hashed passwords, providing a secure mechanism for user authentication.

3. *rewards folder*:

This directory contains individual text files for each child, storing the rewards assigned by parents. Each text file is named after the respective child and contains reward information specific to that child. The folder structure ensures organized storage and retrieval of reward data.

SQL Tables in the "Todo_list" Database:

1. User-Specific Tables:

Each user (child) has a dedicated table storing their tasks. The table is named after the respective username, containing task-related information, including task ID, task description, due date, points, and status (Complete, Incomplete, Overdue).

2. Points Table:

There is a centralized table named "points" that tallies the points of each child account. It stores the total points for each user, facilitating easy tracking and management of points earned or deducted.

Functions used

homepage()

Purpose: Initiates the program.

login()

Purpose: Facilitates user login or account creation.

Steps:

Prompts the user for an existing username and password and allows the creation of a new account.

Check if the provided username and password match an existing account in the 'dat.csv' file.

signup()

Purpose: Creates a new user account.

Steps:

Takes user-provided details (username, password, and account type) and adds them to the 'dat.csv' file.

add_task()

Purpose: Adds a task for a child account.

Steps:

Prompts the user (child) to enter task details, including description, due date, and points.

Validates the input and inserts the task data into the child's database table.

set_completed_tasks()

Purpose: Allows the user to mark a task as complete and updates the status. If the completed task has points, the user's total points are also updated.

Steps:

Displays a list of incomplete or overdue tasks with their details.

Updates the task status to 'Complete.'

If the completed task has points:

Adds the task points to the user's total points.

Updates the total points in the database.

`edit_task()`

Purpose: Allows the user to edit an existing task.

Steps:

Displays current tasks with their details.

Asks for the new task name and optionally the new due date.

Updates the task information in the database.

`view_tasks()`

Purpose: Displays tasks for a child account.

Steps:

Retrieves task data from the child's database table.

Prints the tasks, including task ID, description, due date, points, and status.

`view_completed_tasks()`

Purpose: Displays completed tasks for a child account.

`delete_task()`

Purpose: Allows the user to delete an existing task.

`delete_all_tasks()`

Purpose: Deletes all tasks for a child account.

`add_reward()`

Purpose: Adds a reward for a child account.

`view_rewards()`

Purpose: Displays available rewards for a child account.

Steps:

Reads the rewards data from the child's reward text file.
Prints the available rewards along with their associated points.

`claim_rewards()`

Purpose: Allows a child to claim rewards using their earned points.

Steps:

Retrieves available rewards and total points.
Displays the rewards along with their points and prompts the user to choose a reward.
Deducts points if the user has sufficient points and updates the rewards file accordingly.

`edit_reward()`

Purpose: Allows the parent to edit existing rewards.

Steps:

Displays current rewards, prompts the parent to select a reward to edit, and then allows them to modify the reward name and points.

`delete_reward()`

Purpose: Allows the parent to delete a specific reward.

`delete_all_rewards()`

Purpose: Deletes all rewards for a child account.

`add_points()`

Purpose: Allows the parent to add points to a task.

`calculate_total_points()`

Purpose: Calculates and returns the total points of a child account.

deduct_points(points)

Purpose: Deducts points from the total points of a child account.

view_performances()

Purpose: Visualizes task performances (statuses) in a bar chart.

Steps:

Retrieves task statuses and their counts for a child account from the database.

Plots a bar chart showing the distribution of task statuses.

view_overdue()

Purpose: Displays overdue tasks for a child account.

Steps:

Retrieves and prints details of overdue tasks, including task description, due date, points, and status.

update_overdue_tasks()

Purpose: Updates tasks to 'Overdue' status and deducts points when tasks are past their due date.

Steps:

Checks for tasks with due dates earlier than the current date and updates their status. If not completed, deducts points and potentially reassigns a new task ID.

delete_account()

Purpose: Deletes both child and parent accounts.

Steps:

Reads user data from the 'dat.csv' file.

Prompts the user for confirmation to delete their account.

If confirmed, removes the user's data from the CSV file, deletes their database table, and updates the points table accordingly.

Source Code

```
import mysql.connector as mycon
import random
import csv
import time
from datetime import date, timedelta
from tabulate import tabulate
import matplotlib.pyplot as plt
#To connect to the database
con = mycon.connect(
    host = 'localhost',
    user='root',
    password = '12345678'
)
#Creates the cursor
cur = con.cursor()
#To check if the connection is successful
...
if con.is_connected():
    print("ok")
...
#To create the database Todo_list
...
cur.execute("CREATE DATABASE Todo_list;")
...
#To open the database
cur.execute("use todo_list;")
#To create the points table
...
cur.execute("CREATE TABLE points (username varchar(255) NOT NULL, points integer);")
cur.commit()
...
#To encrypt
def encrypt(message):
    key = random.randint(1, 10)
    encrypted_message = ""
    for char in message:
        encrypted_char = chr(ord(char) + key)
        encrypted_message += encrypted_char
    return encrypted_message, key
#To decrypt
def decrypt(encrypted_message, key):
    decrypted_message = ""
    for char in encrypted_message:
        decrypted_char = chr(ord(char) - key)
```

```

        decrypted_message += decrypted_char
    return decrypted_message
#The sign-ip page
def signup():
    global user_name #This is the name of the user(the database) we will work with.
    global access_type #This is the type of the user that has access to the database.
    global current_date
    while True:
        print(" ")
        print("    This is the signup ")
        print("    page. ")
        print(" ")
        print("    Please create a child account ")
        print("    first before creating ")
        print("    a parent account. ")
        print(" ")
        print("    Is this a ")
        print("    1. Child account ")
        print("    2. Parent account ")
        print("    Enter 1 or 2: ")
        print(" ")
        NeedParent_choice = input("INPUT: ")
        if NeedParent_choice == "2":
            parent = True
            access_type = "Parent"
            break
        elif NeedParent_choice == "1":
            parent = False
            child = False
            access_type = "Child"
            break
        else:
            print(" ")
            print("    Invalid input. Please enter ")
            print("    1 or 2. ")
            print(" ")
    while True:
        print(" ")
        print("    Enter a valid username: ")
        print(" ")
        user_name = input("INPUT: ")
        with open('dat.csv', mode='r') as file: #Read the CSV file and checks if the username
already exists.
            reader = csv.reader(file)
            rows = list(reader)
            items = [row[0] for row in rows]
            if user_name in items:

```



```

print(" ")
print(" | Username already exists. | ")
print(" ")
continue
if len(user_name) < 8 or len(user_name) > 20: #Checks if the username length is in
between (8,20).
    print(" ")
    print(" | Username should be between | ")
    print(" | 8 and 20 characters. | ")
    print(" ")
    continue
break
while True:
    print(" ")
    print(" | Enter a valid password: | ")
    print(" ")
    password = input("INPUT: ")
    if len(password) < 8 or password.isnumeric() or password.isalpha(): #Checks if password
length is in between (8,20) and has has at least 1 alphabet.
        print(" ")
        print(" | Please keep the following in mind: | ")
        print(" | - The password must have at least 1 | ")
        print(" | letter and 1 number. | ")
        print(" | - The password must be longer than 8 | ")
        print(" | characters. | ")
        print(" ")
        continue
    print(" ")
    print(" | Confirm your password: | ")
    print(" ")
    confirm_password = input("INPUT: ")
    if password != confirm_password: #To check if password is the same as confirm_password
        print(" ")
        print(" | Passwords do not match. | ")
        print(" ")
        print(" ")
        print(" | Please keep the following in mind: | ")
        print(" | - The password must have at least 1 | ")
        print(" | letter and 1 number. | ")
        print(" | - The password must be longer than 8 | ")
        print(" | characters. | ")
        print(" ")
        continue
    confirm_password = encrypt(password) #Encrypt the password
    encrypted_password = confirm_password[0]
    key = confirm_password[1] #stores the key
    break
if NeedParent_choice == "2": #Connects parent account with the child account
    while True:

```

```

print(" ")
print("    Does your child have an account? ")
print("    (y - yes / n - no): ")
print(" ")
child_account_choice = input("INPUT: ")
if child_account_choice.lower() == "y":
    with open('dat.csv', mode='r') as file:
        reader = csv.reader(file)
        rows = list(reader)
        username_found = False
        while True:
            print(" ")
            print("    Enter your child's username: ")
            print(" ")
            child = input("INPUT: ")
            for row in rows:
                if child == row[0]:
                    while True:
                        print(" ")
                        print("    Enter your child's password: ")
                        print(" ")
                        child_password = input("INPUT: ")
                        if child_password == decrypt(row[1], int(row[2])):
                            print(" ")
                            print("    Child authenticated ")
                            print(" ")
                            username_found = True
                            break
                        else:
                            print(" ")
                            print("    Invalid password. Please try again. ")
                            print(" ")
            if not username_found:
                print(" ")
                print("    User is not valid ")
                print(" ")
                continue
            break
        break
elif child_account_choice.lower() == "n":
    print(" ")
    print("    Then please create a child account ")
    print(" ")
    signup()
    return
else:
    print(" ")
    print("    Invalid input. Please enter y ")
    print(" ")

```

```

        continue
while True:
    print(" ")
    print("Are you sure with the following details")
    print(f"        Username: {user_name} ")
    print(f"        Password: {password} ")
    print(" (y - yes / n - no): ")
    print(" ")
    user_input = input("INPUT: ")
    if user_input.lower() == "n":
        print(" ")
        print("Registration cancelled. ")
        print(" ")
        signup()
        return
    elif user_input.lower() == "y":
        data = [user_name, encrypted_password, key, parent, child]
        with open('dat.csv', mode='a', newline='') as file:
            writer = csv.writer(file)
            writer.writerow(data)
        current_date = date.today()
        if NeedParent_choice == "1":
            cur.execute(f"CREATE TABLE {user_name} (Task_ID INTEGER primary key, Tasks
VARCHAR(225), Due_date DATE, Status VARCHAR(20) DEFAULT 'Incomplete', Points INTEGER)")
            cur.execute(f"INSERT INTO points VALUES ('{user_name}', 0)")
            con.commit()
            main_menu()
            return
        if NeedParent_choice == "2":
            user_name = child
            main_menu()
            return
    else:
        print(" ")
        print("Invalid input. Please enter y and n ")
        print(" ")
def login(): #Login
    while True:
        global user_name
        global access_type
        global current_date
        print(" ")
        print("This is the login page. ")
        print(" ")
        print(" ")
        print("Enter your username: ")
        print(" ")
        user_name = input("INPUT: ")

```

```

with open('dat.csv', mode='r') as file: #Reads the CSV file to find the row in which
the account details are stored
    reader = csv.reader(file)
    rows = list(reader)
found_user = False
for row in rows:
    if user_name == row[0]:
        found_user = True
        while True:
            print(" ")
            print(" |         Enter your password:         | ")
            print(" |_____| ")
            password = input("INPUT: ")
            key = int(row[2])
            decrypted_password = decrypt(row[1], key) #Decrypt the password
            if password == decrypted_password: #checks the password is correct
                current_date = date.today()
                print(" |_____| ")
                print(" |         Login successful         | ")
                print(" |_____| ")
                if row[3] == "True": #Sets the access_type and username
                    access_type = "Parent"
                    user_name = row[4]
                else:
                    access_type = "Child"
                update_overdue_tasks()
                main_menu()
                return
            else:
                print(" |_____| ")
                print(" | Invalid password. Please try again. | ")
                print(" |_____| ")
                continue
if not found_user:
    print(" ")
    print(" |         User is not valid         | ")
    print(" |_____| ")
    while True:
        print(" |_____| ")
        print(" | Would you like to create a new account | ")
        print(" |         1. Yes         | ")
        print(" |         2. No         | ")
        print(" |_____| ")
        exit_choice = input("INPUT: ")
        if exit_choice == "1":
            signup()
            return
        elif exit_choice == "2":
            login()

```

[illegible]

```

print("\n Invalid input. Please choose a number ")
print("\n from 1 to 3. ")
print("\n")

def main_menu():
    while True:
        print("\n")
        print("\n Welcome to the PyD0! ")
        print(f"\n Today is {current_date} ")
        time.sleep(0.2)
        print("\n Please select an option: ")
        print("\n 1. Task manager ")
        print("\n 2. Reward manager ")
        print("\n 3. Status ")
        if access_type == "Parent":
            print("\n 4. Parent menu ")
            print("\n 5. Sign out ")
        else:
            print("\n 4. Sign out ")
        print("\n Enter your choice : ")
        print("\n")
        choice = input("INPUT: ")
        if choice == "1":
            task_manager()
            return
        elif choice == "2":
            reward_manager()
            return
        elif choice == "3":
            status()
            return
        elif choice == "4" and access_type == "Child":
            homepage()
            return
        elif choice == "4" and access_type == "Parent":
            parent_menu()
            return
        elif choice == "5" and access_type == "Parent":
            homepage()
            return
        else:
            if access_type == "Child":
                print("\n")
                print("\n Invalid input. Please choose a number ")
                print("\n from 1 to 4. ")
                print("\n")
            elif access_type == "Parent":
                print("\n")
                print("\n Invalid input. Please choose a number ")
                print("\n from 1 to 5. ")

```

```

def task_manager():
    while True:
        print("\n=====")
        print("\n      This is the task manager menu      \n")
        print(f"\n      Today is {current_date}              \n")
        time.sleep(0.2)
        print("\n      Please select an option:              \n")
        print("\n      1. View Tasks                          \n")
        print("\n      2. View Overdue Tasks                  \n")
        print("\n      3. Complete Tasks                      \n")
        print("\n      4. Add Task                            \n")
        print("\n      5. Edit Task                           \n")
        print("\n      6. Delete Task                         \n")
        print("\n      7. Delete All Tasks                    \n")
        print("\n      8. Back To Main Menu                  \n")
        print("\n      Enter your choice :                    \n")
        print("\n=====")
        choice = input("INPUT: ")
        if choice == "1":
            view_tasks()
            return
        elif choice == "2":
            view_overdue()
            return
        elif choice == "3":
            set_completed_tasks()
            return
        elif choice == "4":
            add_task()
            return
        elif choice == "5":
            edit_task()
            return
        elif choice == "6":
            delete_task()
            return
        elif choice == "7":
            delete_all_tasks()
            return
        elif choice == "8":
            main_menu()
            return
        else:
            print("\n=====")
            print("\n      Invalid input. Please choose a number \n")
            print("\n      from 1 to 7.                          \n")
            print("\n=====")
def reward_manager():

```

```

while True:
    print("┌───────────────────────────────────┐")
    print("│           This is the reward manager menu           │")
    print(f"│           Today is {current_date}           │")
    time.sleep(0.2)
    print("│           Please select an option:           │")
    print("│           1. View Rewards           │")
    print("│           2. Complete Reward           │")
    print("│           3. Back To Main Menu           │")
    print("│           Enter your choice :           │")
    print("└───────────────────────────────────┘")
    choice = input("INPUT: ")
    if choice == "1":
        view_rewards()
        return
    elif choice == "2":
        claim_rewards()
        return
    elif choice == "3":
        main_menu()
        return
    else:
        print("┌───────────────────────────────────┐")
        print("│ Invalid input. Please choose a number │")
        print("│           from 1 to 3.           │")
        print("└───────────────────────────────────┘")

def parent_menu():
    print("┌───────────────────────────────────┐")
    print("│           This is the parent menu           │")
    print(f"│           Today is {current_date}           │")
    time.sleep(0.2)
    print("│           Please select an option:           │")
    print("│           1. Add/Change Points           │")
    print("│           2. Add rewards           │")
    print("│           3. Edit Reward           │")
    print("│           4. Delete Reward           │")
    print("│           5. Delete All Rewards           │")
    print("│           6. Delete Accounts           │")
    print("│           7. Back To Main Menu           │")
    print("│           Enter your choice :           │")
    print("└───────────────────────────────────┘")
    choice = input("INPUT: ")
    if choice == "1":
        add_points()
        return
    elif choice == "2":
        add_reward()
        return
    elif choice == "3":

```



```

    edit_reward()
    return
elif choice == "4":
    delete_reward()
    return
elif choice == "5":
    delete_all_rewards()
    return
elif choice == "6":
    delete_account()
    return
elif choice == "7":
    main_menu()
    return
else:
    print(" ")
    print("|| Invalid input. Please choose a number ||")
    print("||           from 1 to 3.           ||")
    print("|| ")

def status():
    while True:
        print(" ")
        print("||           This is the status menu           ||")
        print(f"||           Today is {current_date}           ||")
        time.sleep(0.2)
        print("||           Please select an option:           ||")
        print("||           1. Look at performances graph       ||")
        print("||           2. Look at all the Complete tasks  ||")
        print("||           3. View Total Points               ||")
        print("||           4. Back to Main menu               ||")
        print("||           Enter your choice :                 ||")
        print("|| ")
        choice = input("INPUT: ")
        if choice == "1":
            view_performances()
        elif choice == "2":
            view_completed_tasks()
            return
        elif choice == "3":
            print(calculate_total_points())
        elif choice == "4":
            main_menu()
            return
        else:
            print(" ")
            print("|| Invalid input. Please choose a number ||")
            print("||           from 1 to 4.           ||")
            print("|| ")

def view_tasks():

```

```

update_overdue_tasks()
time.sleep(0.2)
cur.execute(f"SELECT Tasks, Due_date, Points, Status FROM {user_name} WHERE Status =
'Incomplete' ORDER BY Due_date")
rows = cur.fetchall()
columns = [desc[0] for desc in cur.description]
print(tabulate(rows, headers=columns, tablefmt='double_grid'))
while True:
    print("┌───────────────────────────────────┐")
    print("│           Would you like go to back menu?           │")
    print("│                               (1 - yes)                               │")
    print("└───────────────────────────────────┘")
    choice = input("INPUT: ")
    if choice == "1":
        task_manager()
        return
    else:
        print("┌───────────────────────────────────┐")
        print("│ Invalid input. Please enter only 1 │")
        print("└───────────────────────────────────┘")

def set_completed_tasks():
    time.sleep(0.2)
    cur.execute(f"SELECT * FROM {user_name} WHERE Status in ('Incomplete','Overdue') ORDER BY
Due_date")
    rows = cur.fetchall()
    if not rows: # No incomplete or overdue tasks
        print("┌───────────────────────────────────┐")
        print("│           You have no incomplete           │")
        print("│           or overdue tasks.           │")
        print("│           Returning to the main menu.           │")
        print("└───────────────────────────────────┘")
        task_manager()
        return
    columns = [desc[0] for desc in cur.description]
    print(tabulate(rows, headers=columns, tablefmt='double_grid'))
    print("┌───────────────────────────────────┐")
    print("│ Enter the number of the row (task) │")
    print("│           you have completed:           │")
    print("└───────────────────────────────────┘")
    choice = input("INPUT: ")
    if choice.isdigit():
        choice = int(choice) - 1
    else:
        print("┌───────────────────────────────────┐")
        print("│ Invalid task number. Please enter a │")
        print("│           number.           │")
        print("└───────────────────────────────────┘")
        set_completed_tasks()
    if choice < 0 or choice >= len(rows):

```

```

print(" ")
print("Invalid task number.")
print(" ")
set_completed_tasks()
if rows[choice][4] == None:
    task_id = rows[choice][0]
    cur.execute(f"UPDATE {user_name} SET Status = 'Complete' WHERE Task_id = {task_id}")
    print(" ")
    print("Task marked as complete successfully!")
    print(" ")
else:
    task_id = rows[choice][0]
    task_points = int(rows[choice][4])
    cur.execute(f"SELECT points FROM points WHERE username = '{user_name}'")
    current_points = int(cur.fetchone()[0])
    cur.execute(f"UPDATE {user_name} SET Status = 'Complete' WHERE Task_id = {task_id}")
    cur.execute(f"UPDATE points SET points = {current_points + task_points} WHERE username = '{user_name}'")
    con.commit()
    print(" ")
    print("Task marked as complete successfully!")
    print(" ")
while True:
    print(" ")
    print("Would you like to mark another task as ")
    print("complete or go back to the main menu? ")
    print("1. Mark another task ")
    print("2. Back to menu ")
    print(" ")
    exit_choice = input("INPUT: ")
    if exit_choice == "1":
        set_completed_tasks()
        return
    elif exit_choice == "2":
        task_manager()
        return
    else:
        print(" ")
        print("Invalid task number. Please enter ")
        print("1 or 2 ")
        print(" ")
        continue
def add_task():
    time.sleep(0.2)
    print(" ")
    print("Enter the task: ")
    print(" ")
    task = input("INPUT: ")
    if access_type == "Child":

```

```

while True:
    try:
        cur.execute(f"SELECT Task_id, Tasks, Due_date, Points FROM {user_name} ORDER BY
Due_date")
        rows = cur.fetchall()
        columns = [desc[0] for desc in rows]
        while True:
            task_id = random.randint(0,99999999)
            if task_id % 10 == 0:
                continue
            if task_id not in columns:
                print("Enter a date (YYYY-MM-DD):")
                date_input = input("INPUT: ")
                cur.execute(f"INSERT INTO {user_name} (Task_id ,Tasks, Due_date) VALUES
({task_id},{task}', '{date_input}')")
                con.commit()
                print("Task inserted successfully!")
                break
        except mycon.errors.DataError:
            current_date = date.today()
            if date_input.lower() == "today":
                date_input = current_date
            elif date_input.lower() == "tomorrow":
                date_input = current_date + timedelta(days=1)
            elif date_input.lower().startswith("in"):
                day = date_input[2:].strip()
                if day.isdigit():
                    date_input = current_date + timedelta(days=int(day))
                else:
                    print("Invalid number. Please enter a integer")
                    continue
            else:
                print("Invalid date format. Please enter a date in the format YYYY-MM-DD.")
                continue
            cur.execute(f"INSERT INTO {user_name} (Task_id ,Tasks, Due_date) VALUES
({task_id},{task}', '{date_input}')")
            con.commit()
            print("Task inserted successfully!")

```

```

        print("└──────────────────────────────────┘")
        break
    break
elif access_type == "Parent":
    print("┌──────────────────────────────────┐")
    print("│      Enter the points for this task:      │")
    print("└──────────────────────────────────┘")
    points = input("INPUT: ")
    while True:
        try:
            cur.execute(f"SELECT Task_id, Tasks, Due_date, Points FROM {user_name} ORDER BY
Due_date")

            rows = cur.fetchall()
            columns = [desc[0] for desc in rows]
            task_id = random.randint(0,99999999)
            while True:
                if task_id not in columns:
                    print("┌──────────────────────────────────┐")
                    print("│      Enter a date (YYYY-MM-DD):      │")
                    print("└──────────────────────────────────┘")
                    date_input = input("INPUT: ")
                    cur.execute(f"INSERT INTO {user_name} (Task_id ,Tasks, Due_date, Points)
VALUES ({task_id},'{'task}','{'date_input}','{'points}')")
                    con.commit()
                    print("┌──────────────────────────────────┐")
                    print("│      Task inserted successfully!      │")
                    print("└──────────────────────────────────┘")
                    break
            except mycon.errors.DataError:
                current_date = date.today()
                if date_input.lower() == "today":
                    date_input = current_date
                elif date_input.lower() == "tomorrow":
                    date_input = current_date + timedelta(days=1)
                elif date_input.lower().startswith("in"):
                    day = date_input[2:].strip()
                    if day.isdigit():
                        date_input = current_date + timedelta(days=int(day))
                    else:
                        print("┌──────────────────────────────────┐")
                        print("│      Invalid number. Please enter      │")
                        print("│      a integer                        │")
                        print("└──────────────────────────────────┘")
                        continue
                else:
                    print("┌──────────────────────────────────┐")
                    print("│      Invalid date format. Please enter a      │")
                    print("│      date in the format YYYY-MM-DD.      │")
                    print("└──────────────────────────────────┘")

```

```

        continue
    cur.execute(f"INSERT INTO {user_name} (Task_id ,Tasks, Due_date, Points) VALUES
({task_id},{task}', '{date_input}', {points})")
    con.commit()
    print("Task inserted successfully!")
    break
break
while True:
    print("Would you like add a new task again or to back menu?")
    print("1. New task")
    print("2. Back to menu")
    exit_choice = input("INPUT: ")
    if exit_choice == "1":
        add_task()
        return
    elif exit_choice == "2":
        task_manager()
        return
    else:
        print("Invalid task number. Please enter 1 or 2")
def edit_task():
    time.sleep(0.2)
    cur.execute(f"SELECT * FROM {user_name} ORDER BY Due_date")
    rows = cur.fetchall()
    if not rows: # No incomplete or overdue tasks
        print("You have no tasks to edit.")
        print("Returning to the main menu.")
        task_manager()
        return
    columns = [desc[0] for desc in cur.description]
    print(tabulate(rows, headers=columns, tablefmt='double_grid'))
    print("Enter the number of the row (task) you want to edit:")
    choice = input("INPUT: ")
    if choice.isdigit():
        choice = int(choice) - 1

```

```

else:
    print(" ")
    print("Invalid input. Please enter a number.")
    print(" ")
    if choice < 0 or choice >= len(rows):
        print(" ")
        print("Invalid task number.")
        print(" ")
        edit_task()
    print(" ")
    print("Enter the new name for task ")
    print(" ")
    new_task = input("INPUT: ")
    task_id = rows[choice][0]
    ToCheckIfOverdue = rows[choice][3]
    while True:
        try:
            print(" ")
            print("Enter the new date (n - if you don't ")
            print("want to change the date): ")
            print(" ")
            new_date = input("INPUT: ")
            if new_date == "n":
                cur.execute(f"UPDATE {user_name} SET Tasks = '{new_task}' WHERE Task_id =
{task_id}")
                con.commit()
                print(" ")
                print("Task updated successfully!")
                print(" ")
                break
            if ToCheckIfOverdue != "Overdue":
                cur.execute(f"UPDATE {user_name} SET Tasks = '{new_task}', Due_date =
'{new_date}' WHERE Task_ID = '{task_id}'")
                con.commit()
                print(" ")
                print("Task updated successfully!")
                print(" ")
                break
            cur.execute(f"UPDATE {user_name} SET Tasks = '{new_task}', Due_date = '{new_date}',
Status = 'Incomplete' WHERE Task_ID = '{task_id}'")
            con.commit()
        except mycon.errors.DataError:
            current_date = date.today()
            if new_date.lower() == "today":
                new_date = current_date
            elif new_date.lower() == "tomorrow":
                new_date = current_date + timedelta(days=1)
            elif new_date.lower().startswith("in"):
                day = new_date[2:].strip()

```

```

        if day.isdigit():
            new_date = current_date + timedelta(days=int(day))
        else:
            print("Invalid number. Please enter")
            print("a integer")
            continue

    else:
        print("Invalid date format. Please enter a")
        print("date in the format YYYY-MM-DD.")
        continue

    cur.execute(f"UPDATE {user_name} SET Tasks = '{new_task}', Due_date = '{new_date}'")
    WHERE Task_ID = '{task_id}'")
    con.commit()
    print("Task updated successfully!")
    break

while True:
    print("Would you like edit a row (task) again")
    print("or to back menu?")
    print("1. Edit again")
    print("2. Back to menu")
    exit_choice = input("INPUT: ")
    if exit_choice == "1":
        edit_task()
        return
    elif exit_choice == "2":
        task_manager()
        return
    else:
        print("Invalid task number. Please enter")
        print("1 or 2")

def delete_task():
    time.sleep(0.2)
    cur.execute(f"SELECT * FROM {user_name} ORDER BY Due_date")
    rows = cur.fetchall()
    if not rows: # No incomplete or overdue tasks
        print("You have no")
        print("tasks to delete.")
        print("Returning to the main menu.")

```



```

print("\n")
task_manager()
return

columns = [desc[0] for desc in cur.description]
print(tabulate(rows, headers=columns, tablefmt='double_grid'))
print("\n")
print("\n Enter the number of the row (task) you")
print("\n want to delete: ")
print("\n")
choice = input("INPUT: ")
if choice.isdigit():
    choice = int(choice) - 1
else:
    print("\n")
    print("\n Invalid task number, please enter a ")
    print("\n valid number ")
    print("\n")
    delete_task()
if choice < 0 or choice >= len(rows):
    print("\n")
    print("\n Invalid task number ")
    print("\n")
    delete_task()
task_id = rows[choice][0]
while True:
    print("\n")
    print("\n Are you sure you want to delete this ")
    print("\n task? (y - yes, n - no): ")
    print("\n")
    conformation = input("INPUT: ")
    if conformation.lower() == "y":
        cur.execute(f"DELETE FROM {user_name} WHERE Task_id = {task_id}")
        con.commit()
        print("\n")
        print("\n Task deleted successfully! ")
        print("\n")
        break
    elif conformation.lower() == "n":
        print("\n")
        print("\n Task not deleted. ")
        print("\n")
        task_manager()
        return
    else:
        print("\n")
        print("\n Invalid input. Please enter only y or n ")
        print("\n")
        continue
while True:

```

```

print("
print("|| Would you like delete a row (task) ||")
print("|| again or go back to the menu? ||")
print("|| (1. Delete again, 2. Back to menu): ||")
print("
exit_choice = input("INPUT: ")
if exit_choice == "1":
    delete_task()
    break
elif exit_choice == "2":
    task_manager()
    break
else:
    print("
    print("|| Invalid task number. Please enter ||")
    print("|| 1 or 2 ||")
    print("

print("
print("|| Invalid task number, please enter a ||")
print("|| valid number ||")
print("

delete_task()
def add_points():
    time.sleep(0.2)
    cur.execute(f"SELECT * FROM {user_name} WHERE Status = 'Incomplete'")
    rows = cur.fetchall()
    columns = [desc[0] for desc in cur.description]
    print(tabulate(rows, headers=columns, tablefmt='double_grid'))
    print("
    print("|| Enter the number of the row (task) to ||")
    print("|| add points: ||")
    print("

    choice = input("INPUT: ")
    if choice.isdigit():
        choice = int(choice) - 1
    else:
        print("
        print("|| Invalid task number. Please enter a ||")
        print("|| valid number. ||")
        print("

        add_points()
    if choice < 0 or choice >= len(rows):
        print("
        print("|| Invalid task number ||")
        print("

        add_points()
    task_id = rows[choice][0]
    print("
    print("|| Enter the points to add: ||")

```

```

print("_____")
points = input("INPUT: ")
cur.execute(f"UPDATE {user_name} SET Points = {points} WHERE Task_id = {task_id}")
con.commit()
print("_____")
print("Points added successfully!")
print("_____")
while True:
    print("_____")
    print("Would you like to add points to another")
    print("task or go back to the menu?")
    print("(1. Add points, 2. Back to menu):")
    print("_____")
    exit_choice = input("INPUT: ")
    if exit_choice == "1":
        add_points()
        return
    elif exit_choice == "2":
        parent_menu()
        return
    else:
        print("_____")
        print("Invalid task number. Please enter")
        print("1 or 2")
        print("_____")
def delete_account():
    time.sleep(0.2)
    print("_____")
    print("This will delete your")
    print("child and parent accounts both.")
    print("_____")
    with open('dat.csv', 'r') as file:
        reader = csv.reader(file)
        rows = list(reader)
    for row in rows:
        if user_name in row:
            print("_____")
            print("Are you sure you want to delete your")
            print("account? This action cannot be undone.")
            print("(1. Delete, 2. Cancel):")
            print("_____")
            confirm = input("INPUT: ")
            if confirm == "1":
                for row in rows:
                    if user_name == row[4]:
                        rows.remove(row)
                for row in rows:
                    if user_name == row[0]:
                        rows.remove(row)

```

```

        with open('dat.csv', 'w', newline='') as file:
            writer = csv.writer(file)
            writer.writerows(rows)
        cur.execute(f"DROP TABLE {user_name}")
        cur.execute(f"DELETE FROM points WHERE username = '{user_name}'")
        print(" ")
        print("    Account deleted successfully.    ")
        print(" ")
        homepage()
        return
    elif confirm == "2":
        print(" ")
        print("    Account deletion canceled.    ")
        print(" ")
        parent_menu()
        return

print(" ")
print(" Username not found. Please try again. ")
print(" ")
delete_account()
def add_reward():
    time.sleep(0.2)
    print(" ")
    print("    Enter the reward name:    ")
    print(" ")
    reward_name = input("INPUT:")
    while True:
        print(" ")
        print("    Enter the points required for this    ")
        print("    reward:    ")
        print(" ")
        reward_points = input("INPUT:")
        if reward_points.isdigit():
            reward_data = f"{reward_name},{reward_points}\n"
            if int(reward_points) > 0:
                with open(f"rewards/{user_name}.txt", mode='a') as file:
                    file.write(reward_data)
                print(" ")
                print("    Reward added successfully!    ")
                print(" ")
                break
            else:
                print(" ")
                print("    Invalid points.    ")
                print(" ")
                continue
        else:
            print(" ")
            print("    Invalid input. Please enter a valid    ")

```

```

        print("||                number.                ||")
        print("||_____|")
        continue
while True:
    print("||_____|")
    print("|| Would you like to add one more reward ||")
    print("||                or go back to the menu? ||")
    print("||                (1. Add reward, 2. Back to menu) ||")
    print("||_____|")
    exit_choice = input("INPUT:")
    if exit_choice == "1":
        add_reward()
        return
    elif exit_choice == "2":
        parent_menu()
        return
    else:
        print("||_____|")
        print("|| Invalid task number. Please enter ||")
        print("||                1 or 2                ||")
        print("||_____|")
def view_rewards():
    time.sleep(0.2)
    reward_file = f"rewards/{user_name}.txt"
    try:
        file = open(reward_file, 'r')
        file.close()
        rewards = []
        with open(reward_file, mode='r') as file:
            for line in file:
                reward_name, reward_points = line.strip().split(',')
                rewards.append((reward_name, int(reward_points)))
        print("Available Rewards:")
        for i, reward in enumerate(rewards, start=1):
            reward_name, reward_points = reward
            print(f"{i}. {reward_name} - {reward_points} points")
    except FileNotFoundError:
        print("No reward available yet.")
while True:
    print("||_____|")
    print("|| Would you like to go to the back menu? ||")
    print("||                (1. Back to menu)                ||")
    print("||_____|")
    choice = input("INPUT: ")
    if choice == "1":
        reward_manager()
        return
    else:
        print("||_____|")

```

```
print("Invalid task number. Please enter ")
print("1")
print("")

def claim_rewards():
    time.sleep(0.2)
    reward_file = f"rewards/{user_name}.txt"
    try:
        with open(reward_file, mode='r') as file:
            rewards = []
            for line in file:
                reward_name, reward_points = line.strip().split(',')
                rewards.append((reward_name, int(reward_points)))
    if len(rewards) == 0:
        print("")
        print("No reward available yet.")
        print("")
        reward_manager()

    print("")
    print("Available Rewards:")
    for i, reward in enumerate(rewards, start=1):
        reward_name, reward_points = reward
        print(f"{i}. {reward_name} - {reward_points} points")
    print("")
    total_points = calculate_total_points()
    print("")
    print(f"Total Points: {total_points}")
    print("")
    if total_points > 0:
        print("")
        print("Enter the number of the reward you ")
        print("want to redeem (or 'q' to quit): ")
        print("")
        choice = input("INPUT: ")
        if choice.isdigit():
            choice = int(choice) - 1
            if 0 <= choice < len(rewards):
                asked_reward_name, reward_points = rewards[choice]
                if total_points >= reward_points:
                    print("")
                    print("Congratulations! You redeemed ")
                    print(f"the reward: {asked_reward_name} ")
                    print("")
                    deduct_points(reward_points)
                    del rewards[choice]
                    with open(reward_file, mode='w') as file:
                        for reward in rewards:
                            reward_name, reward_points = reward
                            file.write(f"{reward_name},{reward_points}\n")
                    reward_manager()
```

```

        else:
            print(" ")
            print("|| You don't have enough points to redeem ||")
            print("||           this reward.           ||")
            print(" ")
            reward_manager()

        else:
            print(" ")
            print("||           Invalid reward number.           ||")
            print(" ")
            claim_rewards()

    elif choice.lower() == 'q':
        reward_manager()

    else:
        print(" ")
        print("|| Invalid input. Please enter a valid ||")
        print("||           reward number or 'q' to quit.           ||")
        print(" ")
        claim_rewards()

    else:
        print(" ")
        print("|| You don't have any points to redeem ||")
        print("||           rewards.           ||")
        print(" ")
        reward_manager()

except FileNotFoundError:
    print(" ")
    print("||           No rewards available yet.           ||")
    print(" ")
    reward_manager()

def calculate_total_points():
    cur.execute(f"SELECT points FROM points WHERE username = '{user_name}'")
    total_points = int(cur.fetchone()[0])
    if total_points == None:
        return 0
    else:
        return total_points

def deduct_points(points):
    cur.execute(f"UPDATE points SET points = points - {points} where username = '{user_name}'")
    con.commit()

def view_performances():
    time.sleep(0.2)
    cur.execute(f"SELECT Status, COUNT(*) as Count FROM {user_name} GROUP BY Status")
    rows = cur.fetchall()
    statuses = [row[0] for row in rows]
    counts = [row[1] for row in rows]
    plt.bar(statuses, counts)
    plt.xlabel('Task Status')
    plt.ylabel('Count')

```

```
plt.title('Task Performances')
plt.show()

def view_completed_tasks():
    time.sleep(0.2)
    cur.execute(f"SELECT Tasks, Due_date, Points, Status FROM {user_name} WHERE Status = 'Complete' ORDER BY Due_date")
    rows = cur.fetchall()
    columns = [desc[0] for desc in cur.description]
    print(tabulate(rows, headers=columns, tablefmt='double_grid'))
    while True:
        print("┌───────────────────────────────────┐")
        print("│ Would you like to go back to the menu? │")
        print("│                               (1 - yes): │")
        print("└───────────────────────────────────┘")
        choice = input("INPUT: ")
        if choice == "1":
            status()
            return
        else:
            print("┌───────────────────────────────────┐")
            print("│ Invalid task number. Please enter │")
            print("│                               1 │")
            print("└───────────────────────────────────┘")

def delete_all_tasks():
    time.sleep(0.2)
    print("┌───────────────────────────────────┐")
    print("│ Are you sure you want to delete │")
    print("│ all tasks? │")
    print("│       (1. Yes, 2. No) │")
    print("└───────────────────────────────────┘")
    choice = input("INPUT:")
    if choice == "1":
        cur.execute(f"DELETE FROM {user_name}")
        con.commit()
        print("┌───────────────────────────────────┐")
        print("│ All tasks deleted successfully. │")
        print("└───────────────────────────────────┘")
        task_manager()
    elif choice == "2":
        task_manager()
    else:
        print("┌───────────────────────────────────┐")
        print("│ Invalid task number. Please enter │")
        print("│               1 or 2 │")
        print("└───────────────────────────────────┘")
        delete_all_tasks()

def edit_reward():
    time.sleep(0.2)
    reward_file = f"rewards/{user_name}.txt"
```



```
try:
    with open(reward_file, 'r') as file:
        rewards = []
        for line in file:
            reward_name, reward_points = line.strip().split(',')
            rewards.append((reward_name, int(reward_points)))

    print("\n")
    print("\t\t\t\t\tCurrent Rewards:\t\t\t\t\t")
    for i, reward in enumerate(rewards, start=1):
        reward_name, reward_points = reward
        print(f"    {i}. {reward_name} - {reward_points} points")
    print("\n")
    print("\n")
    print("\t\t\t\t\tEnter the number of the reward you\t\t\t\t\t")
    print("\t\t\t\t\twant to edit (or 'q' to quit):\t\t\t\t\t")
    print("\n")
    choice = input("INPUT: ")
    if choice.isdigit():
        choice = int(choice) - 1
        if 0 <= choice < len(rewards):
            print("\n")
            print("\t\t\t\t\tEnter the new reward name:\t\t\t\t\t")
            print("\n")
            new_reward_name = input("INPUT: ")
            print("\n")
            print("\t\t\t\t\tEnter the new reward points:\t\t\t\t\t")
            print("\n")
            new_reward_points = input("INPUT: ")
            rewards[choice] = (new_reward_name, int(new_reward_points))
            with open(reward_file, 'w') as file:
                for reward in rewards:
                    reward_name, reward_points = reward
                    file.write(f"{reward_name},{reward_points}\n")
            print("\n")
            print("\t\t\t\t\tReward edited successfully.\t\t\t\t\t")
            print("\n")
            view_rewards()
        else:
            print("\n")
            print("\t\t\t\t\tInvalid reward number.\t\t\t\t\t")
            print("\n")
            view_rewards()
    elif choice.lower() == 'q':
        parent_menu()
    else:
        print("\n")
        print("\t\t\t\t\tInvalid input. Please enter a valid\t\t\t\t\t")
        print("\t\t\t\t\treward number or 'q' to quit.\t\t\t\t\t")
        print("\n")
```

```

        view_rewards()
except FileNotFoundError:
    print(" ")
    print("    File does not exist. ")
    print(" ")
    parent_menu()
def delete_reward():
    time.sleep(0.2)
    reward_file = f"rewards/{user_name}.txt"
    try:
        with open(reward_file, 'r') as file:
            rewards = []
            for line in file:
                rewards.append(line.strip())

        print(" ")
        print("    Current Rewards: ")
        print(" ")
        for i, reward in enumerate(rewards, start=1):
            print(f"{i}. {reward}")
        print(" ")
        print("    Enter the number of the reward you ")
        print("    want to edit (or 'q' to quit): ")
        print(" ")
        choice = input("INPUT: ")
        if choice.isdigit():
            choice = int(choice) - 1
            if 0 <= choice < len(rewards):
                deleted_reward = rewards.pop(choice)
                with open(reward_file, 'w') as file:
                    for reward in rewards:
                        file.write(f"{reward}\n")

                print(" ")
                print(f"    Reward '{deleted_reward}' ")
                print("    deleted successfully. ")
                print(" ")
                parent_menu()
            else:
                print(" ")
                print("    Invalid reward number. ")
                print(" ")
                view_rewards()
        elif choice.lower() == 'q':
            parent_menu()
        else:
            print(" ")
            print("    Invalid input. Please enter a valid ")
            print("    reward number or 'q' to quit. ")
            print(" ")
            view_rewards()

```

```

except FileNotFoundError:
    print("")
    print("File does not exist.")
    print("")
    parent_menu()
def delete_all_rewards():
    time.sleep(0.2)
    reward_file = f"rewards/{user_name}.txt"
    try:
        with open(reward_file, 'w') as file:
            file.write("")
        print("")
        print("All rewards deleted successfully.")
        print("")
        parent_menu()
    except FileNotFoundError:
        print("")
        print("File does not exist.")
        print("")
        parent_menu()
def view_overdue():
    time.sleep(0.2)
    cur.execute(f"SELECT Tasks, Due_date, Points, Status FROM {user_name} WHERE Status = 'Overdue' ORDER BY Due_date")
    rows = cur.fetchall()
    columns = [desc[0] for desc in cur.description]
    print(tabulate(rows, headers=columns, tablefmt='double_grid'))
    while True:
        print("")
        print("Would you like to go back to the menu?")
        print("(1 - yes):")
        print("")
        choice = input("INPUT: ")
        if choice == "1":
            task_manager()
            return
        else:
            print("")
            print("Invalid task number. Please enter")
            print("1")
            print("")
def update_overdue_tasks():
    cur.execute(f"SELECT COUNT(*) FROM {user_name}")
    count = cur.fetchone()[0]
    if count == 0:
        return
    cur.execute(f"SELECT Task_ID FROM {user_name} WHERE Due_date < '{current_date}' AND Status != 'Complete'")
    overdue_task_ids = cur.fetchall()

```

```

if overdue_task_ids:
    for i in overdue_task_ids:
        a = i[0]
        if a % 10 != 0:
            cur.execute(f"UPDATE points SET points = points - 10 WHERE username
='{user_name}''")
        else:
            while True:
                retask_id = random.randint(0,99999999)
                if retask_id % 10 == 0:
                    cur.execute(f"UPDATE {user_name} SET Task_ID = '{retask_id}' WHERE
Task_ID = '{a}' ")
                    break
            cur.execute(f"UPDATE {user_name} SET Status = 'Overdue' WHERE Due_date < '{current_date}'
AND Status != 'Complete'")
            con.commit()
homepage()

```

Output Screen

Child account creation

Py-00

Welcome to the Main Menu!

Please select an option:

1. Sign up
2. Login
3. Quit

Enter your choice (1-3):

INPUT: 1

This is the signup page.

Please create a child account first before creating a parent account.

Is this a

1. Child account
2. Parent account

Enter 1 or 2:

INPUT: 1

Enter a valid username:

INPUT: childtest

Enter a valid password:

INPUT: 12345678a

Confirm your password:

INPUT: 12345678a

Are you sure with the following details
Username: childtest
Password: 12345678a
(y - yes / n - no):

INPUT: t

Invalid input. Please enter y and n


Are you sure with the following details
Username: childtest
Password: 12345678a
(y - yes / n - no):

INPUT: y

Welcome to the PyD0!
Today is 2023-12-20
Please select an option:
1. Task manager
2. Reward manager
3. Status
4. Sign out
Enter your choice :

INPUT: 1

```
mysql> show tables;
+-----+
| Tables_in_todo_list |
+-----+
| childtest           |
| points              |
+-----+
2 rows in set (0.01 sec)
```

 dat.csv

```
1 childtest,56789;<e,4,False,False
```

Add Tasks and view tasks

This is the task manager menu
Today is 2023-12-20
Please select an option:
1. View Tasks
2. View Overdue Tasks
3. Complete Tasks
4. Add Task
5. Edit Task
6. Delete Task
7. Delete All Tasks
8. Back To Main Menu
Enter your choice :

INPUT: 4

Enter the task:

INPUT: complete homework

Enter a date (YYYY-MM-DD):

INPUT: tomorrow

Task inserted successfully!

Would you like add a new task again or
to back menu?

1. New task
2. Back to menu

INPUT: 1

Enter the task:

INPUT: computer exam

Enter a date (YYYY-MM-DD):

INPUT: 2024-01-05

This is the task manager menu
Today is 2023-12-20
Please select an option:
1. View Tasks
2. View Overdue Tasks
3. Complete Tasks
4. Add Task
5. Edit Task
6. Delete Task
7. Delete All Tasks
8. Back To Main Menu
Enter your choice :

INPUT: 1

Tasks	Due_date	Points	Status
complete homework	2023-12-21		Incomplete
computer exam	2024-01-05		Incomplete

Would you like go to back menu?
(1 - yes)

INPUT: 1

```
mysql> select * from childtest;
```

Task_ID	Tasks	Due_date	Status	Points
56637388	complete homework	2023-12-21	Incomplete	NULL
58837525	computer exam	2024-01-05	Incomplete	NULL

Edit tasks

```
This is the task manager menu
Today is 2023-12-20
Please select an option:
  1. View Tasks
  2. View Overdue Tasks
  3. Complete Tasks
  4. Add Task
  5. Edit Task
  6. Delete Task
  7. Delete All Tasks
  8. Back To Main Menu
Enter your choice :
```

INPUT: 5

Task_ID	Tasks	Due_date	Status	Points
56637388	complete homework	2023-12-21	Incomplete	
58837525	computer exam	2024-01-05	Incomplete	

```
Enter the number of the row (task) you
want to edit:
```

INPUT: 2

```
Enter the new name for task
```

INPUT: maths exam

```
Enter the new date (n - if you don't
want to change the date):
```

INPUT: n

```
Task updated successfully!
```

```
Would you like edit a row (task) again
or to back menu?
  1. Edit again
  2. Back to menu
```

INPUT: 2

```
mysql> select * from childtest;
+-----+-----+-----+-----+-----+
| Task_ID | Tasks          | Due_date | Status   | Points |
+-----+-----+-----+-----+-----+
| 56637388 | complete homework | 2023-12-21 | Incomplete | NULL |
| 58837525 | maths exam      | 2024-01-05 | Incomplete | NULL |
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

Delete Task

INPUT: 2

```
This is the task manager menu
Today is 2023-12-20
Please select an option:
1. View Tasks
2. View Overdue Tasks
3. Complete Tasks
4. Add Task
5. Edit Task
6. Delete Task
7. Delete All Tasks
8. Back To Main Menu
Enter your choice :
```

INPUT: 6

Task_ID	Tasks	Due_date	Status	Points
56637388	complete homework	2023-12-21	Incomplete	
58837525	maths exam	2024-01-05	Incomplete	

```
Enter the number of the row (task) you
want to delete:
```

INPUT: 1

```
Are you sure you want to delete this
task? (y - yes, n - no):
```

INPUT: y

```
Task deleted successfully!
```

```
Would you like delete a row (task)
again or go back to the menu?
(1. Delete again, 2. Back to menu):
```

INPUT: 2

```
mysql> select * from childtest;
```

```
+-----+-----+-----+-----+-----+
| Task_ID | Tasks      | Due_date | Status    | Points |
+-----+-----+-----+-----+-----+
| 58837525 | maths exam | 2024-01-05 | Incomplete | NULL   |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```


Completing tasks

This is the task manager menu

Today is 2023-12-20

Please select an option:

1. View Tasks
2. View Overdue Tasks
3. Complete Tasks
4. Add Task
5. Edit Task
6. Delete Task
7. Delete All Tasks
8. Back To Main Menu

Enter your choice :

INPUT: 3

Task_ID	Tasks	Due_date	Status	Points
58837525	maths exam	2024-01-05	Incomplete	

Enter the number of the row (task)
you have completed:

INPUT: 1

Task marked as complete successfully!

Would you like to mark another task as
complete or go back to the main menu?

1. Mark another task
2. Back to menu

```
mysql> select * from childtest;
```

```
+-----+-----+-----+-----+-----+
| Task_ID | Tasks      | Due_date | Status  | Points |
+-----+-----+-----+-----+-----+
| 58837525 | maths exam | 2024-01-05 | Complete | NULL  |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Overdue tasks

Enter the task:

INPUT: clean room

Enter a date (YYYY-MM-DD):

INPUT: 2023-12-19

Task inserted successfully!

Would you like add a new task again or
to back menu?

1. New task
2. Back to menu

INPUT: 2

This is the task manager menu

Today is 2023-12-20

Please select an option:

1. View Tasks
2. View Overdue Tasks
3. Complete Tasks
4. Add Task
5. Edit Task
6. Delete Task
7. Delete All Tasks
8. Back To Main Menu

Enter your choice :

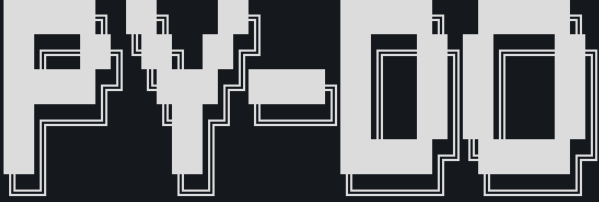
INPUT: 2

Tasks	Due_date	Points	Status
clean room	2023-12-19		Overdue

Would you like to go back to the menu?
(1 - yes):

```
mysql> select * from points;
+-----+-----+
| username | points |
+-----+-----+
| childtest |    -10 |
+-----+-----+
1 row in set (0.00 sec)
```

Creating a parent account and connecting it with a child account



Welcome to the Main Menu!

Please select an option:

1. Sign up
2. Login
3. Quit

Enter your choice (1-3):

INPUT: 1

This is the signup page.

Please create a child account first before creating a parent account.

Is this a

1. Child account
2. Parent account

Enter 1 or 2:

INPUT: 2

Enter a valid username:

INPUT: parenttest

Enter a valid password:

INPUT: abcdefghi1

Confirm your password:

INPUT: abcdefghi1

Does your child have an account?
(y - yes / n - no):

INPUT: y

Enter your child's username:

INPUT: childtest1

User is not valid

Enter your child's username:

INPUT: childtest

Enter your child's password:

INPUT: 12345678a

Child authenticated

Are you sure with the following details
Username: parenttest
Password: abcdefghi1
(y - yes / n - no):

INPUT: y

Welcome to the PyD0!
Today is 2023-12-20
Please select an option:

1. Task manager
2. Reward manager
3. Status
4. Parent menu
5. Sign out

Enter your choice :

Adding points to tasks made by the child

```
Welcome to the PyD0!  
Today is 2023-12-20  
Please select an option:  
1. Task manager  
2. Reward manager  
3. Status  
4. Parent menu  
5. Sign out  
Enter your choice :
```

INPUT: 4

```
This is the parent menu  
Today is 2023-12-20  
Please select an option:  
1. Add/Change Points  
2. Add rewards  
3. Edit Reward  
4. Delete Reward  
5. Delete All Rewards  
6. Delete Accounts  
7. Back To Main Menu  
Enter your choice :
```

INPUT: 1

Task_ID	Tasks	Due_date	Status	Points
2905519	phy exam	2024-01-07	Incomplete	

```
Enter the number of the row (task) to  
add points:
```

INPUT: 1

```
Enter the points to add:
```

INPUT: 20

```
Points added successfully!
```

```
Would you like to add points to another  
task or go back to the menu?  
(1. Add points, 2. Back to menu):
```

```
mysql> select * from childtest;
```

```
+-----+-----+-----+-----+-----+  
| Task_ID | Tasks      | Due_date | Status   | Points |  
+-----+-----+-----+-----+-----+  
| 2905519 | phy exam   | 2024-01-07 | Incomplete | 20 |  
| 58837525 | maths exam | 2024-01-05 | Complete   | NULL |  
| 79053554 | clean room | 2023-12-19 | Overdue    | NULL |  
+-----+-----+-----+-----+-----+  
3 rows in set (0.00 sec)
```

Adding and editing rewards

This is the parent menu
Today is 2023-12-20
Please select an option:
1. Add/Change Points
2. Add rewards
3. Edit Reward
4. Delete Reward
5. Delete All Rewards
6. Delete Accounts
7. Back To Main Menu
Enter your choice :

INPUT: 2

Enter the reward name:

INPUT:pizza

Enter the points required for this
reward:

INPUT:10

Reward added successfully!

≡ *childtest.txt* ✕

rewards > ≡ *childtest.txt*

1 pizza,10
2

This is the parent menu
Today is 2023-12-20
Please select an option:
1. Add/Change Points
2. Add rewards
3. Edit Reward
4. Delete Reward
5. Delete All Rewards
6. Delete Accounts
7. Back To Main Menu
Enter your choice :

INPUT: 3

Current Rewards:
1. pizza - 10 points

Enter the number of the reward you
want to edit (or 'q' to quit):

INPUT: 1

Enter the new reward name:

INPUT: burger

Enter the new reward points:

INPUT: 10

Reward edited successfully.

≡ *childtest.txt* ✕

rewards > ≡ *childtest.txt*

1 burger,10
2

Completing tasks with points and claiming rewards.

Task_ID	Tasks	Due_date	Status	Points
79053554	clean room	2023-12-19	Overdue	
38174398	phy exam	2024-01-07	Incomplete	20

Enter the number of the row (task) you have completed:

INPUT: 2

Task marked as complete successfully!

```
mysql> select * from points;
+-----+-----+
| username | points |
+-----+-----+
| childtest |    10 |
+-----+-----+
1 row in set (0.00 sec)
```

Welcome to the PyD0!
Today is 2023-12-20
Please select an option:
1. Task manager
2. Reward manager
3. Status
4. Parent menu
5. Sign out
Enter your choice :

INPUT: 2

This is the reward manager menu
Today is 2023-12-20
Please select an option:
1. View Rewards
2. Complete Reward
3. Back To Main Menu
Enter your choice :

INPUT: 2

Available Rewards:

1. burger – 10 points

Total Points: 10

Enter the number of the reward you want to redeem (or 'q' to quit):

INPUT: 1

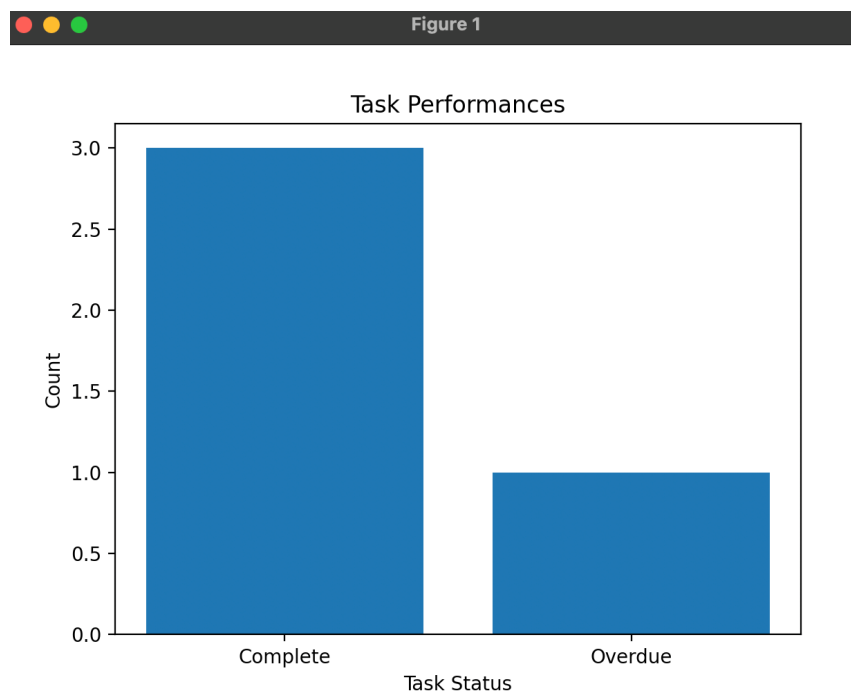
Congratulations! You redeemed the reward: burger

```
≡ childtest.txt ×
rewards > ≡ childtest.txt
1
```

```
mysql> select * from points;
+-----+-----+
| username | points |
+-----+-----+
| childtest |     0 |
+-----+-----+
1 row in set (0.00 sec)
```

Performances Graph

```
Welcome to the PyD0!  
Today is 2023-12-20  
Please select an option:  
1. Task manager  
2. Reward manager  
3. Status  
4. Parent menu  
5. Sign out  
Enter your choice :  
  
INPUT: 3  
  
This is the status menu  
Today is 2023-12-20  
Please select an option:  
1. Look at performances graph  
2. Look at all the Complete tasks  
3. View Total Points  
4. Back to Main menu  
Enter your choice :  
  
INPUT: 1
```



Sign out

```
Welcome to the PyD0!  
Today is 2023-12-20  
Please select an option:  
1. Task manager  
2. Reward manager  
3. Status  
4. Parent menu  
5. Sign out  
Enter your choice :
```

INPUT: 5

Py-D00

```
Welcome to the Main Menu!
```

```
Please select an option:
```

```
1. Sign up  
2. Login  
3. Quit  
Enter your choice (1-3):
```

INPUT: 3

```
Thank you for using Py-D0.  
Stay organized and stay productive!
```

GOODBYE!

Limitations

Dependency on Parent-Child Structure:

The project is designed for a specific parent-child account relationship, limiting its use to scenarios with at least one parent and one child. It doesn't cater to individual users without this structure.

Single Parent and Child:

The system currently supports only a single parent-child pair. Extending it to accommodate multiple parents or children would require modifications for broader usability.

Limited Task Attributes:

The tasks are currently represented with basic attributes such as task name, due date, points, and status. Enhancing the system to support additional details or task categorization could provide users with more comprehensive task management.

Requirements

HARDWARE REQUIRE

- Printer, to print the required documents for the project.
- Processor: Apple M2
- RAM: 8 GB
- Hard Disk: 500 Gb

SOFTWARE REQUIRED

- Operating system: macOS Sonoma
- Programming Language: Written with Python and Thonny for execution of the program.
- Application Software: MS Word, for presentation of output.

BIBLIOGRAPHY



<http://www.google.com>

<http://en.wikipedia.org>

Computer Science with Python: PREETI ARORA