ICCT Colleges, Inc. Antipolo Satellite Campus

Intermediate Programming (CC03) – Daily Time Record (DTR) System

Group 7

Group Leader: Pantaleon, Lian Mae A.

Group Members:

Pagulong, Aaron James F.

Panizares, Gennilyn E.

Perez, Timothy C.

Prestado, Mark Anthony B.

Razon, Angelo S.

Reyes, Jovanne D.

Sagales, Rhim Angelo A.

Sambile, Johnlex R.

Villamora, Ricardo G.

Villaflores, Rinel L.

<u>Mr. Ronelle Clark Esplanada</u>

Instructor

Date submitted: Dec 5, 2023

Table of Contents

I. TITLE **1**

II. TABLE OF CONTENTS **2**

III. INPUT SECTION **3**

IV. PROCESS SECTION **9**

- Menu Screen 9
- Timekeeping Screen 9
- Register Screen 13
- View Employee 16
- Exit Screen 17

V. OUTPUT SECTION **18**

- Menu Screen 18
- Timekeeping Screen 18
- Register Screen 19
- View Employee 19

VI. CONCLUSION **20**

VII. APPENDICES **21**

- Appendix A 22
- Appendix B 25

Input Section

The program operates by employing various elements such as variables, different types of data, and initial values, each serving a specific role in ensuring the effective functioning of the system. These components collectively contribute to the smooth execution and accurate management of data within the daily time record system. By using different kinds of information and ways to store it, the program is good at handling details, which is important for making sure the system works well and efficiently. This approach to managing variables and data is integral to the overall functionality of the system, ensuring it performs optimally in recording and overseeing daily time-related details.

Variable Name	**Data Type**	**Initial Value**	**Description**
formattedTime	String	Current Time	A variable that stores the current time, formatted in a 12-hour cycle without a leading zero on the hour.
formattedDate	String	Current Date	A variable that contains the current date, formatted in the 'YYYY-MM-DD' style.
askStartDate	String	Empty	A variable that stores the user's input regarding the start date.
intStartDate	DateTime Object	Empty	A variable that contains a datetime object representing the start date.

askEndDate	String	Empty	A variable that stores the user’s input regarding the end date
intEndDate	DateTime Object	Empty	A variable that contains a datetime object representing the end date.
intTimeIn	DateTime Object	Time-In Input	An object representation of the user's Time-In input.
intTimeOut	DateTime Object	Time-Out Input	An object representation of the user's Time-Out input.
timeIn	String	Empty	A variable that stores the user's input regarding time-in.
timeOut	String	Empty	A variable that stores the user's input regarding time-out.
empId	List	Empty	A variable that will act as a container for Employee ID.
empFirstName	List	Empty	A variable that will act as a container for Employee First Name.
empLastName	List	Empty	A variable that will act as a container for Employee Last Name.
empDepartment	List	Empty	A variable that will act as a container for Employee Department.
empPosition	List	Empty	A variable that will act as a container for Employee Position.

empTotalHours	List	Empty	A variable that will act as a container for Employee Total Hours Worked.
empTotalAbsent	List	Empty	A variable that will act as a container for Employee Total Absences.
empDateStart	List	Empty	A variable that will act as a container for Employee Start Dates.
empDateEnd	List	Empty	A variable that will act as a container for Employee End Dates.
menuChoice	Integer	Empty	A variable that stores the user's input regarding menu choice.
askEmpID	Integer	Empty	A variable that stores the user's input about Empoyee ID.
askToRegister	String	Empty	A variable that stores the user's input about wanting to register or not..
empInd	Integer	Empty	A variable that holds the index corresponding to a specific employee, with the value varying based on the Employee ID (askEmpID).
totalAbsent	Integer	0	A variable that will act as temporary container for total absences.
totalHours	Integer	0	A variable that will act as temporary container for total hours worked.

hours	Integer	Empty	A variable that holds the hours extracted from the ‘timeIn’/‘timeOut’ after splitting.
minutes	Integer	Empty	A variable that holds the minutes extracted from the ‘timeIn’/‘timeOut’ after splitting.
timeWorked	TimeDelta Object	Difference between two Time Object	A variable that holds the difference when we subtract 'intStartDate' and 'intEndDate'.
result	Float	Total Hours worked	A variable that acts like a temporary holder for the total hours worked(floating point number) in a specific date.
registerFname	String	Empty	A variable that stores the user's input about his/her first name.
registerLname	String	Empty	A variable that stores the user's input about his/her last name.
splitParts	List	Extracted First/Last name after splitting	A variable that stores the first name/last name obtained after performing a split operation.

registerID	Integer	Empty	A variable that stores the user's input about the chosen employee ID.
registerDep	Integer	Empty	A variable that stores the user's input about the chosen department.
registerPos	Integer	Empty	A variable that stores the user's input about the chosen position.
askToDo	String	Empty	A varaible that stores the user's input about what to do after a successful registration.
viewEmp	Integer	Empty	A variable that holds the user’s input regarding the employee ID they wish to view.
viewEmpID	Integer	Empty	A variable that holds the index corresponding to a specific employee, with the value varying based on the Employee ID (viewEmp).
startDate	String	Empty	A variable that stores the string representation of the start date after extraction.
endDate	String	Empty	A variable that stores the string representation of the end date after extraction.

totalAbsentEx	Integer	Empty	A variable that stores the extracted total hours worked of a specific employee.
totalHoursEx	Float	Empty	A variable that stores the extracted total absences of a specific employee.
askIfView	String	Empty	A variable that holds the user’s input inquiring whether they wish to view another employee.

Process Section

The system's functionality is structured around three main screens: the Menu Screen, Timekeeping Screen, and Register Screen. Each screen involves distinct steps and decision points outlined to guide the program's execution. Within the outlined instructions, checks and validations maintain data integrity, handle user inputs gracefully, and guide the program through various scenarios. The structure provides a clear blueprint for implementing the Daily Time Record System.

Menu Screen

How do we handle inputs in the main menu?

if Choice = 1, then you enter Time Keeping screen.

Else if Choice = 2, then you enter the Register Employee screen.

Else if Choice = 3, then you enter the View Employee screen.

Else if Choice = 4, it will take you to the Exit screen.

Else, it will ask the user repeatedly until it enters a correct input.

Timekeeping Screen

How do we handle inputs for Employee ID?

If the entered Employee ID is registered, then we proceed to the next step of timekeeping.

Else if the entered Employee ID is a negative integer, then display message saying kindly put positive integer only and repeat this process.

Else If the entered Employee ID is not registered, ask the employee if he/she wants to register.

Else, display a message saying that only integer input is allowed and then repeat the process.

Start Date

How do we handle and make sure that the input for Start Date is in "YYYY-MM-DD" format?

If the conversion of the input to a Date-Time Object is successful ("YYYY-MM-DD" format), then store the entered Start Date to the storage(variable) and then proceed to the next process.

Else, display a message saying that it is an invalid Start Date Entry and then repeat this process.

End Date

How do we handle and make sure that the input for End Date is in "YYYY-MM-DD" format? And how do we make sure that End Date is not prior to Start Date entry?

If the input is in the "YYYY-MM-DD" format, then proceed to the next checking.

If the End Date is earlier than the Start Date, then display a message saying that the entered End Date is earlier than Start Date and then repeat this process.

Else, we proceed to the next timekeeping step.

Else, display a message saying that it is an invalid End Date Entry and then repeat this process.

Check-In

How do we handle Correct Format and Incorrect Format input for Check-In? And how do we calculate total absences for a particular period?

First, divide the user's input into hours and minutes.

Second, try converting the Check-In input to a Date-Time Object.

Successful Conversion to Date-Time Object (Correct Format):

If either the hour or minute does not consist of exactly two digits, then display a message saying that the input is not following the correct format and then repeat this process.

Unsuccessful Conversion to Date-Time Object (Incorrect Format):

If input is A **or** a, then add 1 to a temporary storage(variable) for total absences because we need to accumulate the total absences in that period before storing it to the storage(variable) for total absences and then proceed to Check-Out.

Ask the user for Check-Out input.

If Check-Out input is “A” **or** “a”, then proceed to next timekeeping step.

Else, it displays a message saying enter ‘a’ if absent.

Else, it displays a message saying that the user entered an incorrect Time-In input and then repeats this process.

Check-Out

How do we handle Correct Format and Incorrect Format input for Check-Out? And how do we calculate total hours worked for a particular period?

First, divide the user’s input into hours and minutes.

Second, try converting the Check-Out input to a Date-Time Object.

Successful Conversion to Date-Time Object (Correct Format):

If either the hour or minute does not consist of exactly two digits, then display a message saying that the input is not following the correct format and then repeat this process.

Else, proceed to the next check.

If Check-Out Input is earlier than the Check-In, then display a message saying that the Check-Out input is earlier than the Check-In.

Else, subtract Check-Out to Check-In, after that we can divide the difference to 3600(1 Hour), after dividing it, the quotient will be the total hours worked, and then we will add this value to a temporary storage(variable) for total hours worked before adding it to the real storage for total hours worked, and then proceed to next instruction.

Unsuccessful Conversion to Date-Time Object (Incorrect Format):

Display a message saying that user entered an incorrect Time-Out input and then repeat this process.

After the Check-In and Check-Out, what's next?

After completing the Check-In and Check-Out processes, we can finally store the accumulated absences and total hours worked in their respective variables.

What to do next after a successful timekeeping?

Ask the user for input if he/she wants to place another time keeping entry.

If input is Y or y, then repeat the whole process of timekeeping.

Else If input is N or n, then go back to the main menu.

Else, display a message that says only "Y" and "N" are the only allowed input and then repeat this process.

Register Screen

How do we handle and make sure the entered Employee ID is unique and valid?

If the entered Employee ID is not a registered Employee ID and not a negative integer, it will the add Employee ID on the list.

Else, it displays the text that shows invalid input, and it will repeat the process of entering an Employee ID until it becomes valid.

How do we handle input for first name in Register Screen?

If the entered first name has no digit or any special character (except space), the process will continue the next step, and the first name will be added on the list (list variable).

Else, it will display the text "Invalid Entry! Please enter a valid First Name."

How do we handle input for last name in Register Screen?

If the entered last name has no digit or any special character (except space), the process will continue the next step, and the last name will be added on the list (list variable).

Else, it will display the text "Invalid Entry! Please enter a valid last Name."

How do we handle and validate input for department?

If the entered input is less than 3 and greater than 0, proceed to the next evaluation.

If input = 1, then add “Faculty” to the list (list variable).

Else, then add “Non-Faculty” to the list (list variable).

Else, display a message saying that only 1 or 2 are the only allowed input.

How do we handle and validate input for position?

If the entered input is less than 3 and greater than 0, proceed to the next evaluation.

If input = 1, then add “Full-Time” to the list (list variable).

Else, then add “Part-Time” to the list (list variable).

Else, display a message saying that only 1 or 2 are the only allowed input.

After a successful registration, what to do next?

Ask the user if he/she wants to register again.

If choice = Y or y, then repeat the whole process of registration.

Else if choice = N or n, simply go back to the main menu.

Else, repeatedly ask the user for correct input.

View Employee

How do we handle inputs for Employee ID?

If the entered Employee ID is not registered, proceed to the next evaluation.

If the entered Employee ID is less than 0, then display a message saying that only positive integer is allowed and then continue the process.

Else, ask the user to register and evaluate what to do based on the input.

Else, stop asking the user and then proceed to the next step.

How do we make sure that the entered Employee ID is integer?

Using try block, all instructions about getting the employee id is done in try block.

Using except block, if there are exceptions raised in the try block, it will display a message saying only integer values are allowed.

How do we retrieve and display the information of an employee?

First, using the entered Employee ID as basis we search for its index in the list (variable list).

Once we have found the index, we will now display the personal information using the index as a basis.

Next, we are going to check the records of that employee.

If the records regarding the Start Date stored for that employee are not empty, then display all the records stored like total hours worked and total absences, separately for each period.

Else, just display a message saying there are no records found.

After a successful viewing, what to do next?

Ask the user if he/she wants to view another employee.

If choice = Y or y, then repeat the whole process of viewing an employee from the start.

Else if choice = N or n, then simply go back to the main menu.

Else, repeatedly ask the user until a valid input is entered.

Exit Screen

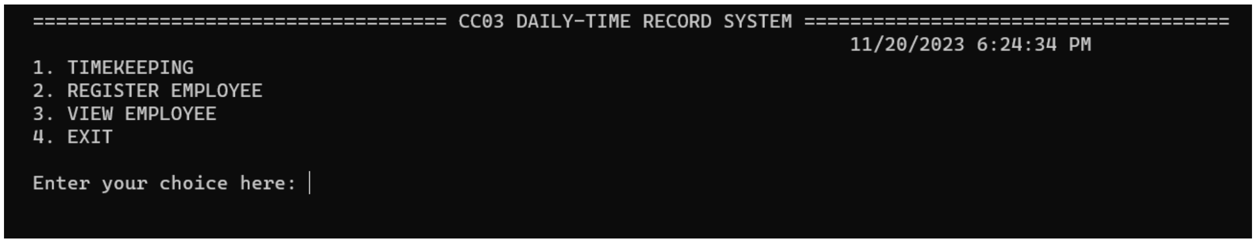
How do we exit/terminate the program?

Display a message saying Goodbye, then immediately terminate the program, effectively stopping all running processes.

Output Section

Menu Screen

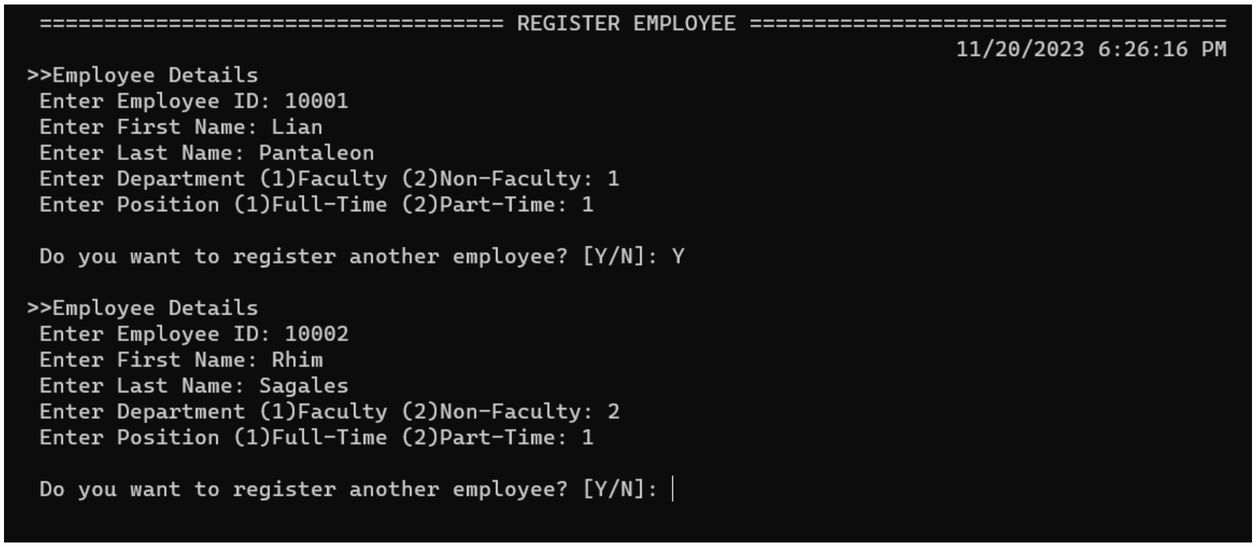
On the Welcome screen, the upper right corner displays the current time and date, while on the left side, there are four options: Timekeeping, Employee Registration, Employee Viewing, and Exit. You have the flexibility to select the desired option for entry.



```
==================================== CC03 DAILY-TIME RECORD SYSTEM ======================================
                                                                        11/20/2023 6:24:34 PM
1. TIMEKEEPING
2. REGISTER EMPLOYEE
3. VIEW EMPLOYEE
4. EXIT

Enter your choice here:
```

Register Employee Screen

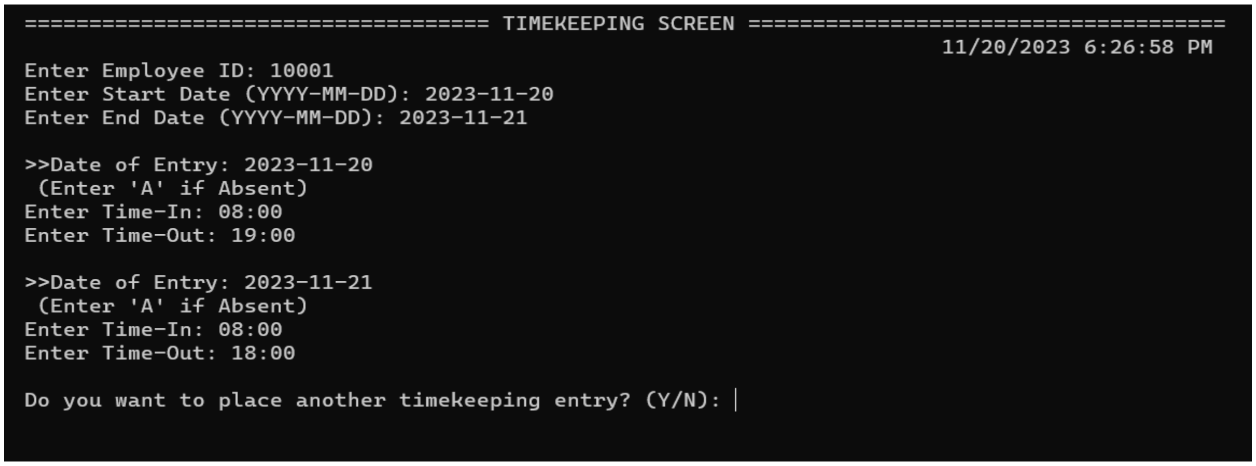
On this screen, the system will request you to input your information, such as Employee ID, Name, Department, and Position. This function manages the input from users during employee registration, ensuring that essential data is provided and adheres to specified formats and constraints.



```
 ==================================== REGISTER EMPLOYEE =====================================
                                                                       11/20/2023 6:26:16 PM
>>Employee Details
 Enter Employee ID: 10001
 Enter First Name: Lian
 Enter Last Name: Pantaleon
 Enter Department (1)Faculty (2)Non-Faculty: 1
 Enter Position (1)Full-Time (2)Part-Time: 1

 Do you want to register another employee? [Y/N]: Y

>>Employee Details
 Enter Employee ID: 10002
 Enter First Name: Rhim
 Enter Last Name: Sagales
 Enter Department (1)Faculty (2)Non-Faculty: 2
 Enter Position (1)Full-Time (2)Part-Time: 1

 Do you want to register another employee? [Y/N]:
```

Timekeeping Screen

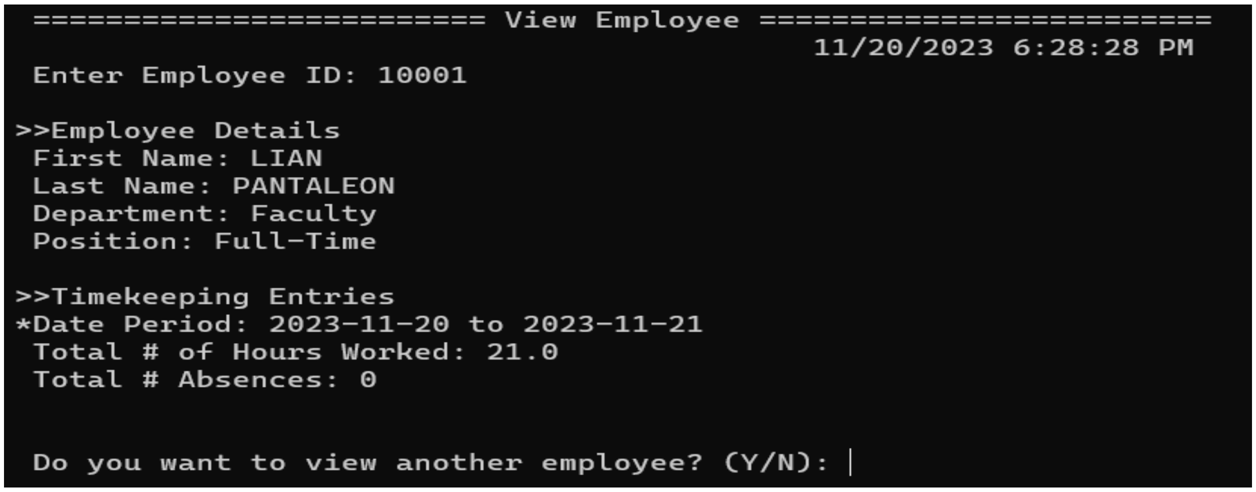
Specify start and end dates for the timekeeping period while entering time-in and time-out for employees. This function oversees the entire timekeeping process, validating input formats, calculating total hours worked, and storing the records appropriately.



```
=================================== TIMEKEEPING SCREEN ====================================
                                                                   11/20/2023 6:26:58 PM
Enter Employee ID: 10001
Enter Start Date (YYYY-MM-DD): 2023-11-20
Enter End Date (YYYY-MM-DD): 2023-11-21

>>Date of Entry: 2023-11-20
 (Enter 'A' if Absent)
Enter Time-In: 08:00
Enter Time-Out: 19:00

>>Date of Entry: 2023-11-21
 (Enter 'A' if Absent)
Enter Time-In: 08:00
Enter Time-Out: 18:00

Do you want to place another timekeeping entry? (Y/N):
```

View Employee Screen

The View Employee records all your timekeeping entries and associated information. By inputting the Employee ID of the individual you wish to check, the system displays their basic information along with detailed timekeeping entries, offering a comprehensive overview of their attendance history.



```
 ========================= View Employee =========================
                                             11/20/2023 6:28:28 PM
 Enter Employee ID: 10001

>>Employee Details
 First Name: LIAN
 Last Name: PANTALEON
 Department: Faculty
 Position: Full-Time

>>Timekeeping Entries
*Date Period: 2023-11-20 to 2023-11-21
 Total # of Hours Worked: 21.0
 Total # Absences: 0


 Do you want to view another employee? (Y/N):
```

Conclusion

The development of our Daily Time Record System showcase a concrete framework designed for effective time management and employee record-keeping. The pseudo code outlines diligent processes, from handling user inputs to computing working hours and validating date entries, ensuring accuracy and reliability.

With features like dynamic date and time entry, employee records, and the ability to register new employees, the system addresses essential aspects of time tracking and workforce management. Hence, the console style, along with prompts and error messages, needs to be enhance to make the user interaction more effective and minimizes potential issues.

In essence, the Daily Time Record System, emerges as a practical and well-structured solution for organizations seeking an efficient and user-centric approach to managing employee attendance and time-related data. Its adaptability and focus on user experience position it as a valuable tool for businesses aiming to streamline their daily time recording processes.

In enhancing the Daily Time Record System, several strategic recommendations can be considered to elevate its functionality and user experience.

- User Feedback and Notifications

Implement clear and informative messages or notifications to guide users through the system. This can include success messages, error notifications, and prompts for the next steps.

- Color Coding for Output

Use color coding in console outputs to distinguish between different types of messages (information, warnings, and errors). This visual cue can quickly draw attention to critical information.

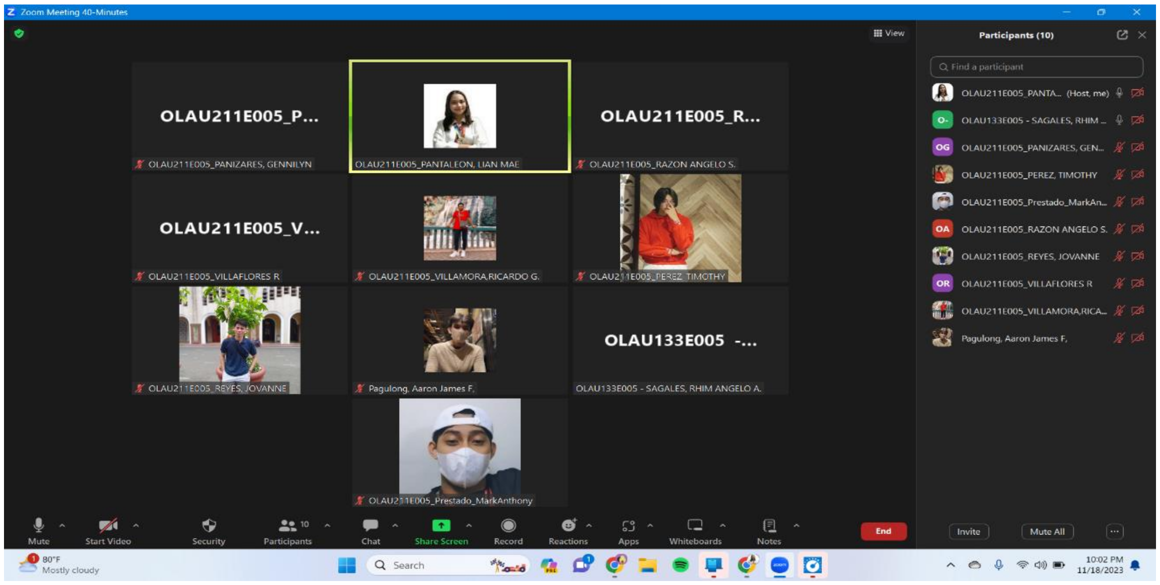
- Responsive Design

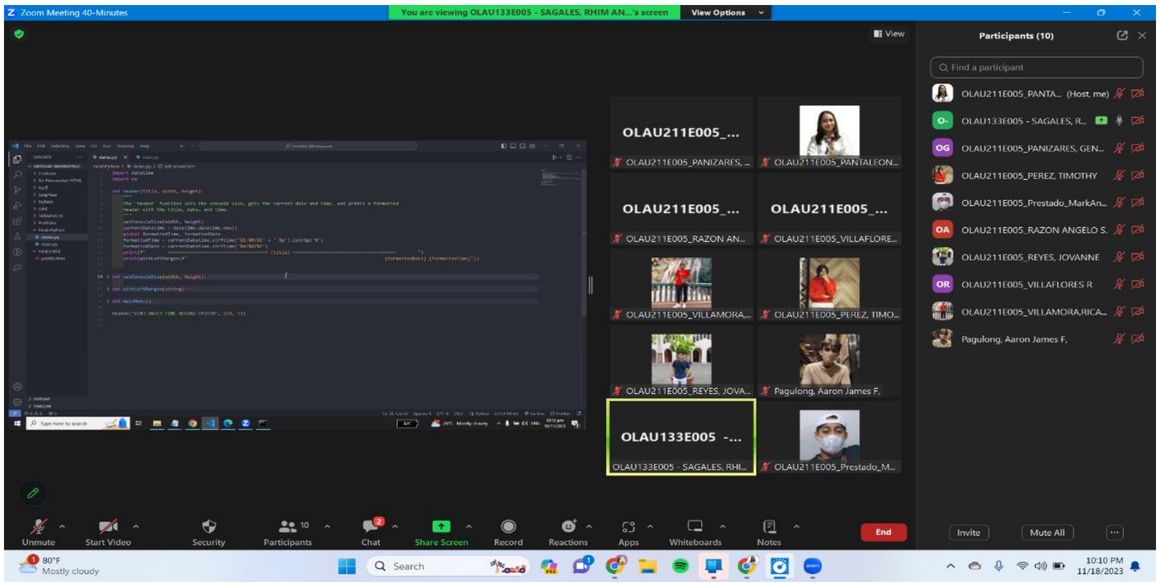
Ensure that the console interface is responsive to different screen sizes. Design the layout to adapt naturally to varying screen dimensions without requiring manual resizing.

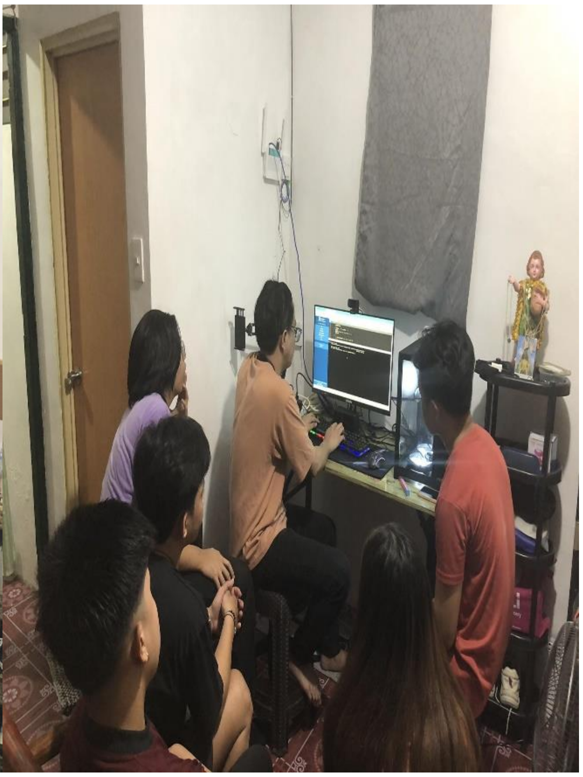
Appendices

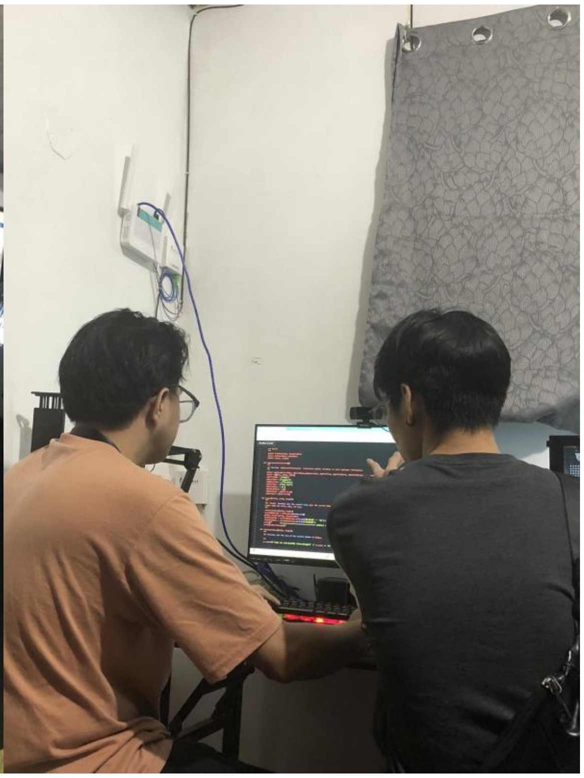
Appendix A

Virtual and personal meeting during the development of our system.


Zoom Meeting 40-Minutes
Participants (10)
OLAU211E005_PANIZARES, GENNILYN
OLAU211E005_PANTALEON, LIAN MAE
OLAU211E005_RAZON ANGELO S.
OLAU211E005_VILLAFLORES R
OLAU211E005_VILLAMORA,RICARDO G.
OLAU211E005_PEREZ, TIMOTHY
OLAU211E005_REYES, JOVANNE
Pagulong, Aaron James F.
OLAU133E005 - SAGALES, RHIM ANGELO A.
OLAU211E005_Prestado_MarkAnthony
Mute
Start Video
Security
Participants
Chat
Share Screen
Record
Reactions
Apps
Whiteboards
Notes
End
Invite
Mute All
10:02 PM
11/18/2023

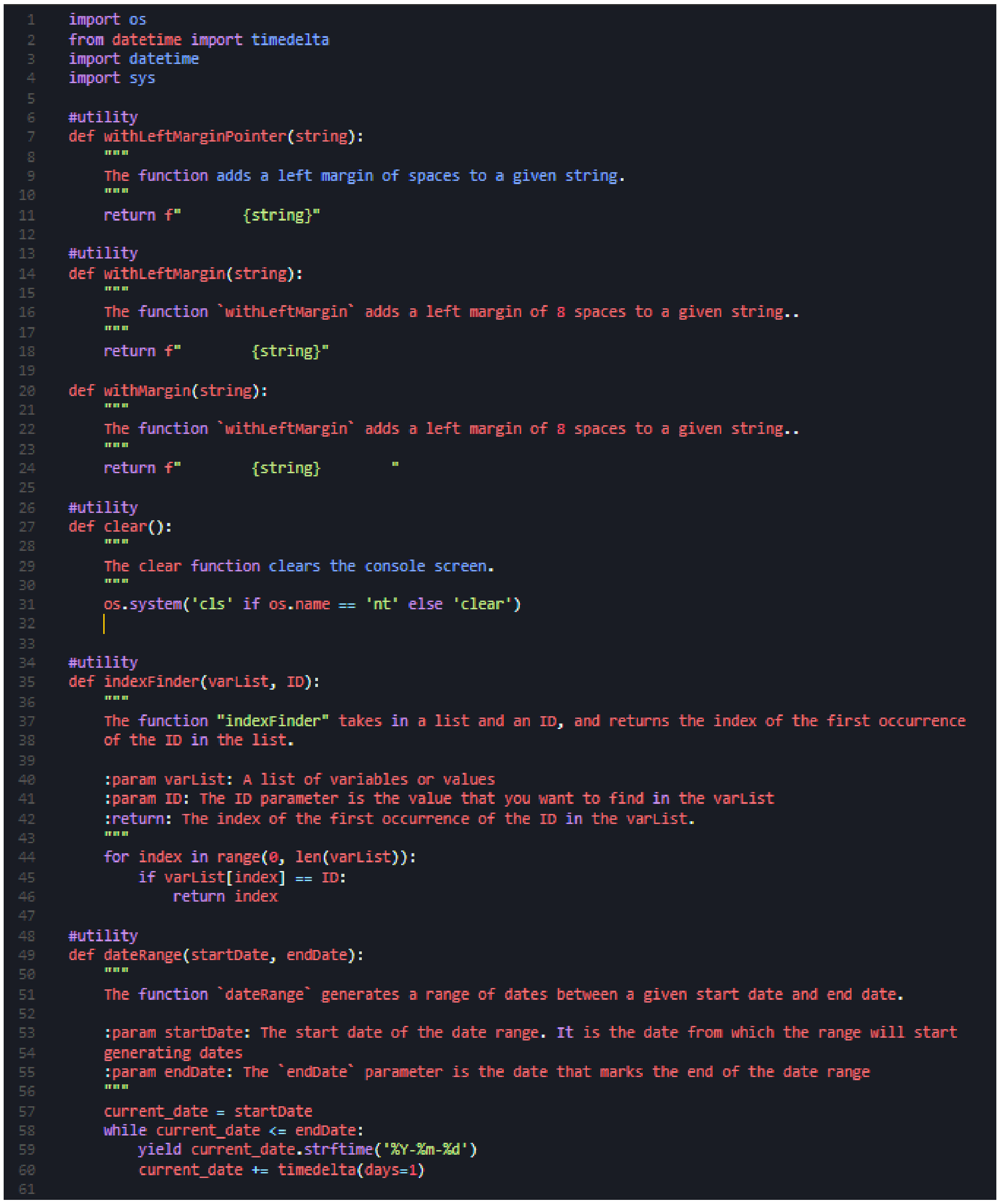




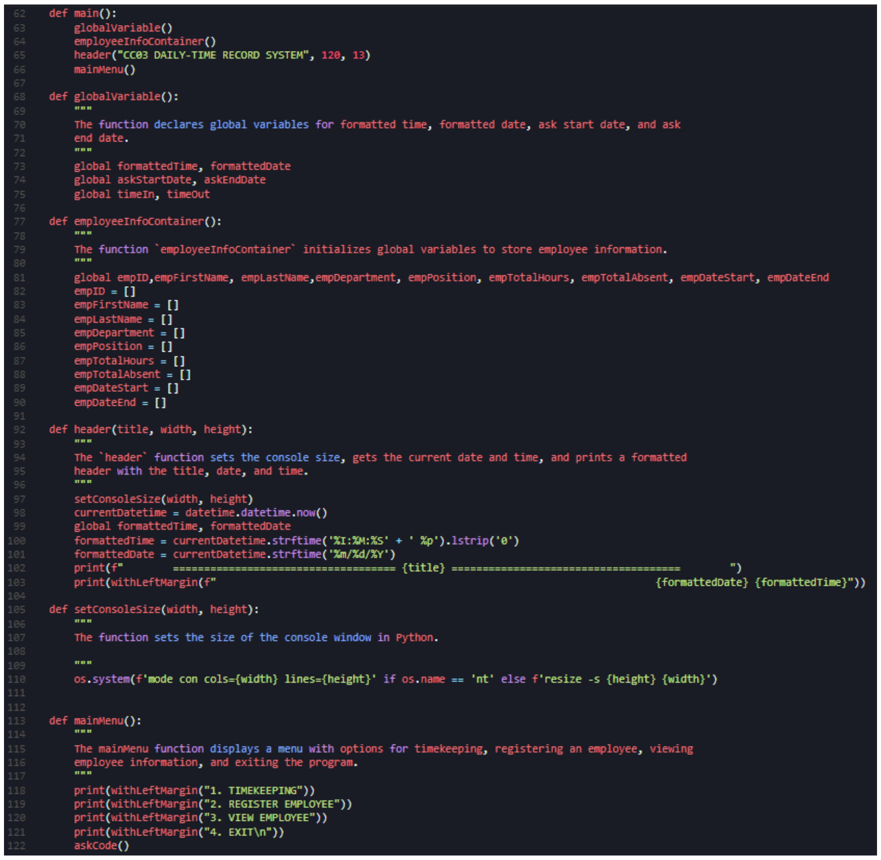


Appendix B

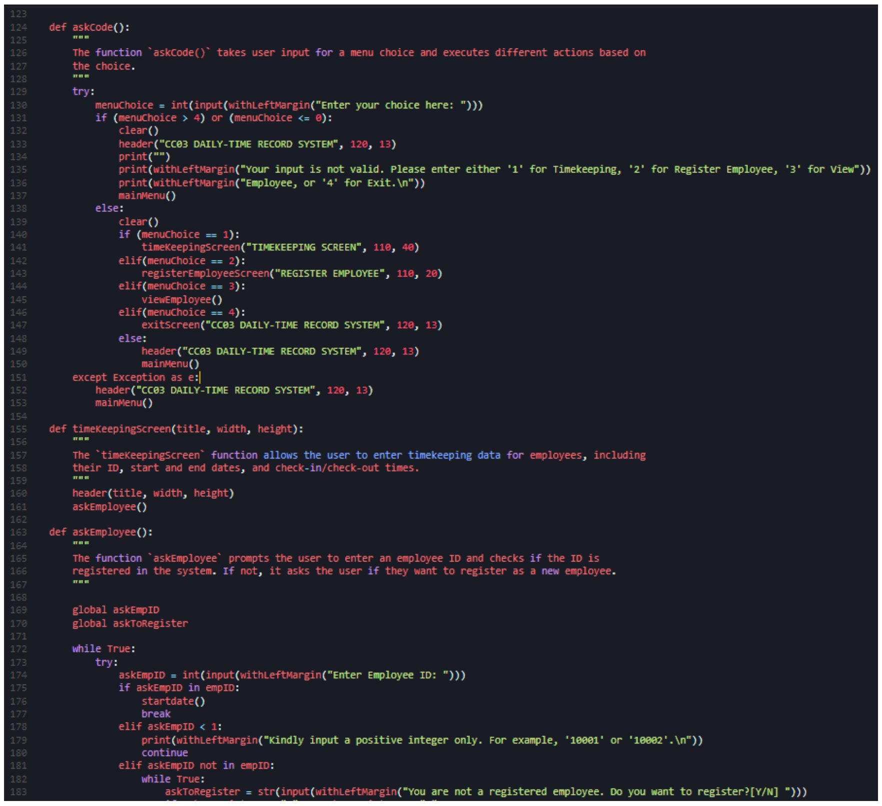
Source code: Daily Time Record System



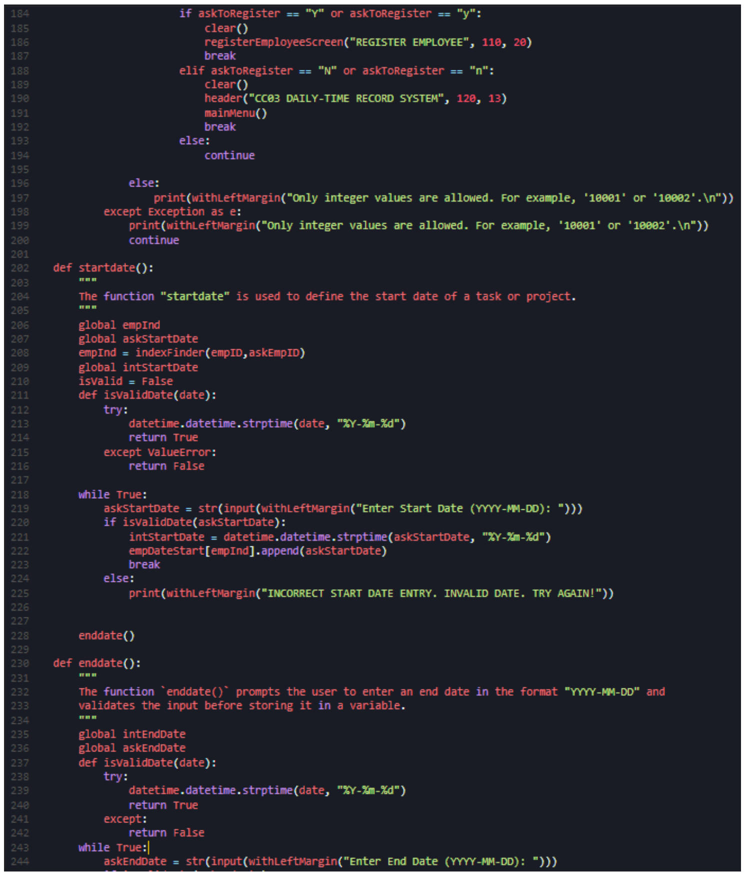
```python
import os
from datetime import timedelta
import datetime
import sys

#utility
def withLeftMarginPointer(string):
    """
    The function adds a left margin of spaces to a given string.
    """
    return f"       {string}"

#utility
def withLeftMargin(string):
    """
    The function `withLeftMargin` adds a left margin of 8 spaces to a given string..
    """
    return f"        {string}"

def withMargin(string):
    """
    The function `withLeftMargin` adds a left margin of 8 spaces to a given string..
    """
    return f"        {string}        "

#utility
def clear():
    """
    The clear function clears the console screen.
    """
    os.system('cls' if os.name == 'nt' else 'clear')
    

#utility
def indexFinder(varList, ID):
    """
    The function "indexFinder" takes in a list and an ID, and returns the index of the first occurrence
    of the ID in the list.

    :param varList: A list of variables or values
    :param ID: The ID parameter is the value that you want to find in the varList
    :return: The index of the first occurrence of the ID in the varList.
    """
    for index in range(0, len(varList)):
        if varList[index] == ID:
            return index

#utility
def dateRange(startDate, endDate):
    """
    The function `dateRange` generates a range of dates between a given start date and end date.

    :param startDate: The start date of the date range. It is the date from which the range will start
    generating dates
    :param endDate: The `endDate` parameter is the date that marks the end of the date range
    """
    current_date = startDate
    while current_date <= endDate:
        yield current_date.strftime('%Y-%m-%d')
        current_date += timedelta(days=1)

```



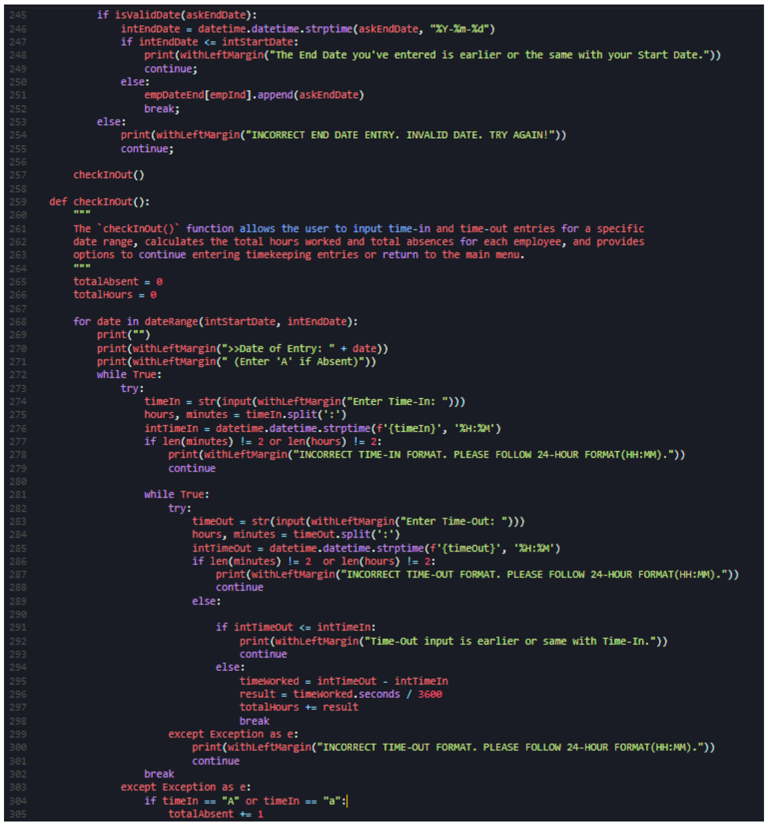
```python
def main():
    globalVariable()
    employeeInfoContainer()
    header("CC03 DAILY-TIME RECORD SYSTEM", 120, 13)
    mainMenu()

def globalVariable():
    """
    The function declares global variables for formatted time, formatted date, ask start date, and ask
    end date.
    """
    global formattedTime, formattedDate
    global askStartDate, askEndDate
    global timeIn, timeOut

def employeeInfoContainer():
    """
    The function `employeeInfoContainer` initializes global variables to store employee information.
    """
    global empID,empFirstName, empLastName,empDepartment, empPosition, empTotalHours, empTotalAbsent, empDateStart, empDateEnd
    empID = []
    empFirstName = []
    empLastName = []
    empDepartment = []
    empPosition = []
    empTotalHours = []
    empTotalAbsent = []
    empDateStart = []
    empDateEnd = []

def header(title, width, height):
    """
    The `header` function sets the console size, gets the current date and time, and prints a formatted
    header with the title, date, and time.
    """
    setConsoleSize(width, height)
    currentDatetime = datetime.datetime.now()
    global formattedTime, formattedDate
    formattedTime = currentDatetime.strftime('%I:%M:%S' + ' %p').lstrip('0')
    formattedDate = currentDatetime.strftime('%m/%d/%Y')
    print(f"        ==================================== {title} =====================================        ")
    print(withLeftMargin(f"                                                                         {formattedDate} {formattedTime}"))

def setConsoleSize(width, height):
    """
    The function sets the size of the console window in Python.

    """
    os.system(f'mode con cols={width} lines={height}' if os.name == 'nt' else f'resize -s {height} {width}')


def mainMenu():
    """
    The mainMenu function displays a menu with options for timekeeping, registering an employee, viewing
    employee information, and exiting the program.
    """
    print(withLeftMargin("1. TIMEKEEPING"))
    print(withLeftMargin("2. REGISTER EMPLOYEE"))
    print(withLeftMargin("3. VIEW EMPLOYEE"))
    print(withLeftMargin("4. EXIT\n"))
    askCode()
```



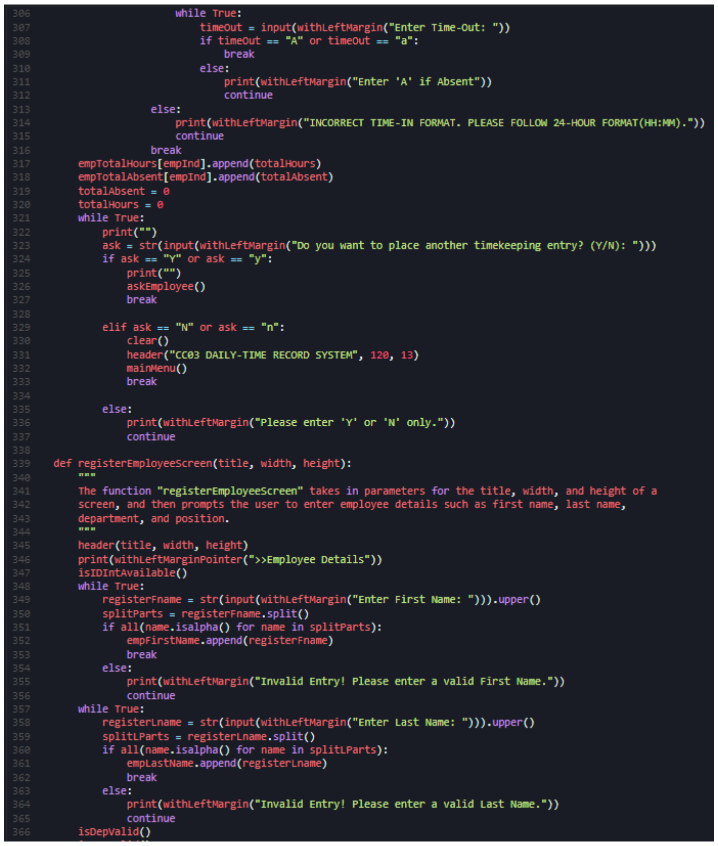
```python

def askCode():
    """
    The function `askCode()` takes user input for a menu choice and executes different actions based on
    the choice.
    """
    try:
        menuChoice = int(input(withLeftMargin("Enter your choice here: ")))
        if (menuChoice > 4) or (menuChoice <= 0):
            clear()
            header("CC03 DAILY-TIME RECORD SYSTEM", 120, 13)
            print("")
            print(withLeftMargin("Your input is not valid. Please enter either '1' for Timekeeping, '2' for Register Employee, '3' for View"))
            print(withLeftMargin("Employee, or '4' for Exit.\n"))
            mainMenu()
        else:
            clear()
            if (menuChoice == 1):
                timeKeepingScreen("TIMEKEEPING SCREEN", 110, 40)
            elif(menuChoice == 2):
                registerEmployeeScreen("REGISTER EMPLOYEE", 110, 20)
            elif(menuChoice == 3):
                viewEmployee()
            elif(menuChoice == 4):
                exitScreen("CC03 DAILY-TIME RECORD SYSTEM", 120, 13)
            else:
                header("CC03 DAILY-TIME RECORD SYSTEM", 120, 13)
                mainMenu()
    except Exception as e:
        header("CC03 DAILY-TIME RECORD SYSTEM", 120, 13)
        mainMenu()

def timeKeepingScreen(title, width, height):
    """
    The `timeKeepingScreen` function allows the user to enter timekeeping data for employees, including
    their ID, start and end dates, and check-in/check-out times.
    """
    header(title, width, height)
    askEmployee()

def askEmployee():
    """
    The function `askEmployee` prompts the user to enter an employee ID and checks if the ID is
    registered in the system. If not, it asks the user if they want to register as a new employee.
    """

    global askEmpID
    global askToRegister

    while True:
        try:
            askEmpID = int(input(withLeftMargin("Enter Employee ID: ")))
            if askEmpID in empID:
                startdate()
                break
            elif askEmpID < 1:
                print(withLeftMargin("Kindly input a positive integer only. For example, '10001' or '10002'.\n"))
                continue
            elif askEmpID not in empID:
                while True:
                    askToRegister = str(input(withLeftMargin("You are not a registered employee. Do you want to register?[Y/N] ")))
```



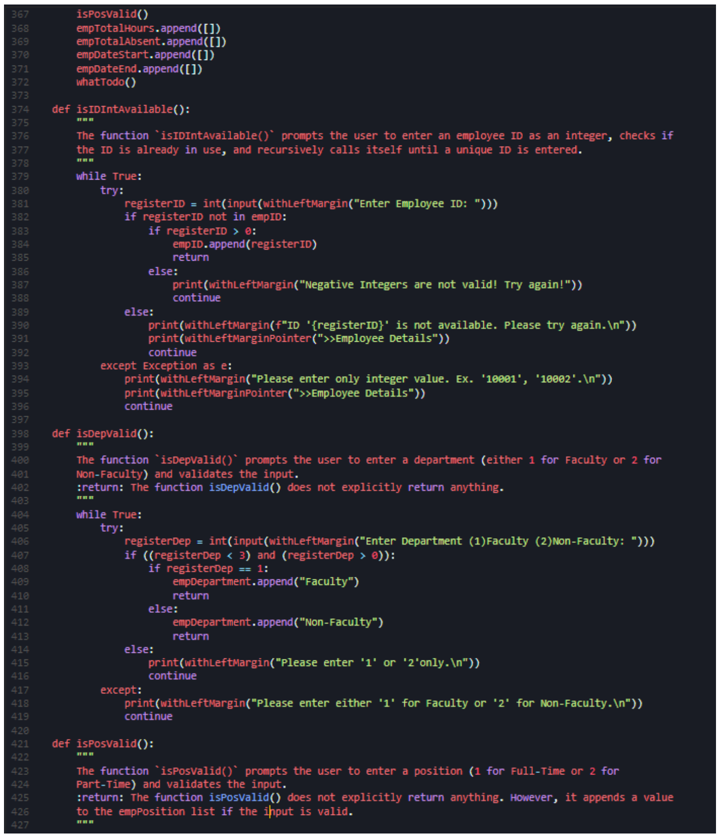
```python
                        if askToRegister == "Y" or askToRegister == "y":
                            clear()
                            registerEmployeeScreen("REGISTER EMPLOYEE", 110, 20)
                            break
                        elif askToRegister == "N" or askToRegister == "n":
                            clear()
                            header("CC03 DAILY-TIME RECORD SYSTEM", 120, 13)
                            mainMenu()
                            break
                        else:
                            continue

                else:
                    print(withLeftMargin("Only integer values are allowed. For example, '10001' or '10002'.\n"))
            except Exception as e:
                print(withLeftMargin("Only integer values are allowed. For example, '10001' or '10002'.\n"))
                continue

    def startdate():
        """
        The function "startdate" is used to define the start date of a task or project.
        """
        global empInd
        global askStartDate
        empInd = indexFinder(empID,askEmpID)
        global intStartDate
        isValid = False
        def isValidDate(date):
            try:
                datetime.datetime.strptime(date, "%Y-%m-%d")
                return True
            except ValueError:
                return False

        while True:
            askStartDate = str(input(withLeftMargin("Enter Start Date (YYYY-MM-DD): ")))
            if isValidDate(askStartDate):
                intStartDate = datetime.datetime.strptime(askStartDate, "%Y-%m-%d")
                empDateStart[empInd].append(askStartDate)
                break
            else:
                print(withLeftMargin("INCORRECT START DATE ENTRY. INVALID DATE. TRY AGAIN!"))


        enddate()

    def enddate():
        """
        The function `enddate()` prompts the user to enter an end date in the format "YYYY-MM-DD" and
        validates the input before storing it in a variable.
        """
        global intEndDate
        global askEndDate
        def isValidDate(date):
            try:
                datetime.datetime.strptime(date, "%Y-%m-%d")
                return True
            except:
                return False
        while True:
            askEndDate = str(input(withLeftMargin("Enter End Date (YYYY-MM-DD): ")))
```



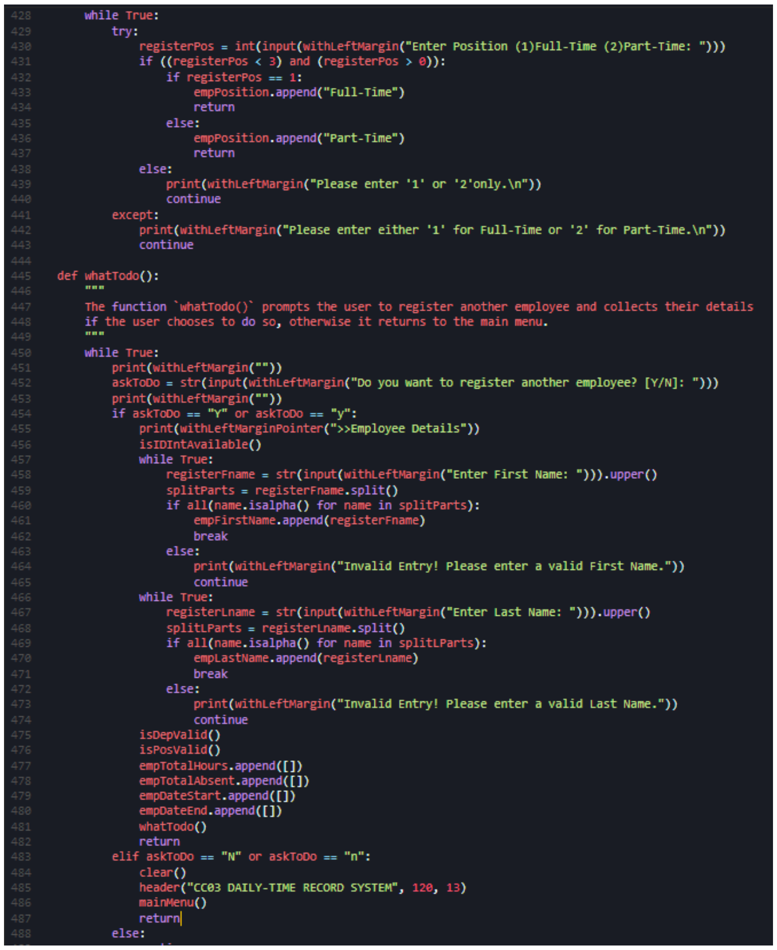
```python
            if isValidDate(askEndDate):
                intEndDate = datetime.datetime.strptime(askEndDate, "%Y-%m-%d")
                if intEndDate <= intStartDate:
                    print(withLeftMargin("The End Date you've entered is earlier or the same with your Start Date."))
                    continue;
                else:
                    empDateEnd[empInd].append(askEndDate)
                    break;
            else:
                print(withLeftMargin("INCORRECT END DATE ENTRY. INVALID DATE. TRY AGAIN!"))
                continue;

        checkInOut()

    def checkInOut():
        """
        The `checkInOut()` function allows the user to input time-in and time-out entries for a specific
        date range, calculates the total hours worked and total absences for each employee, and provides
        options to continue entering timekeeping entries or return to the main menu.
        """
        totalAbsent = 0
        totalHours = 0

        for date in dateRange(intStartDate, intEndDate):
            print("")
            print(withLeftMargin(">>Date of Entry: " + date))
            print(withLeftMargin(" (Enter 'A' if Absent)"))
            while True:
                try:
                    timeIn = str(input(withLeftMargin("Enter Time-In: ")))
                    hours, minutes = timeIn.split(':')
                    intTimeIn = datetime.datetime.strptime(f'{timeIn}', '%H:%M')
                    if len(minutes) != 2 or len(hours) != 2:
                        print(withLeftMargin("INCORRECT TIME-IN FORMAT. PLEASE FOLLOW 24-HOUR FORMAT(HH:MM)."))
                        continue

                    while True:
                        try:
                            timeOut = str(input(withLeftMargin("Enter Time-Out: ")))
                            hours, minutes = timeOut.split(':')
                            intTimeOut = datetime.datetime.strptime(f'{timeOut}', '%H:%M')
                            if len(minutes) != 2  or len(hours) != 2:
                                print(withLeftMargin("INCORRECT TIME-OUT FORMAT. PLEASE FOLLOW 24-HOUR FORMAT(HH:MM)."))
                                continue
                            else:

                                if intTimeOut <= intTimeIn:
                                    print(withLeftMargin("Time-Out input is earlier or same with Time-In."))
                                    continue
                                else:
                                    timeWorked = intTimeOut - intTimeIn
                                    result = timeWorked.seconds / 3600
                                    totalHours += result
                                    break
                        except Exception as e:
                            print(withLeftMargin("INCORRECT TIME-OUT FORMAT. PLEASE FOLLOW 24-HOUR FORMAT(HH:MM)."))
                            continue
                    break
                except Exception as e:
                    if timeIn == "A" or timeIn == "a":
                        totalAbsent += 1
```



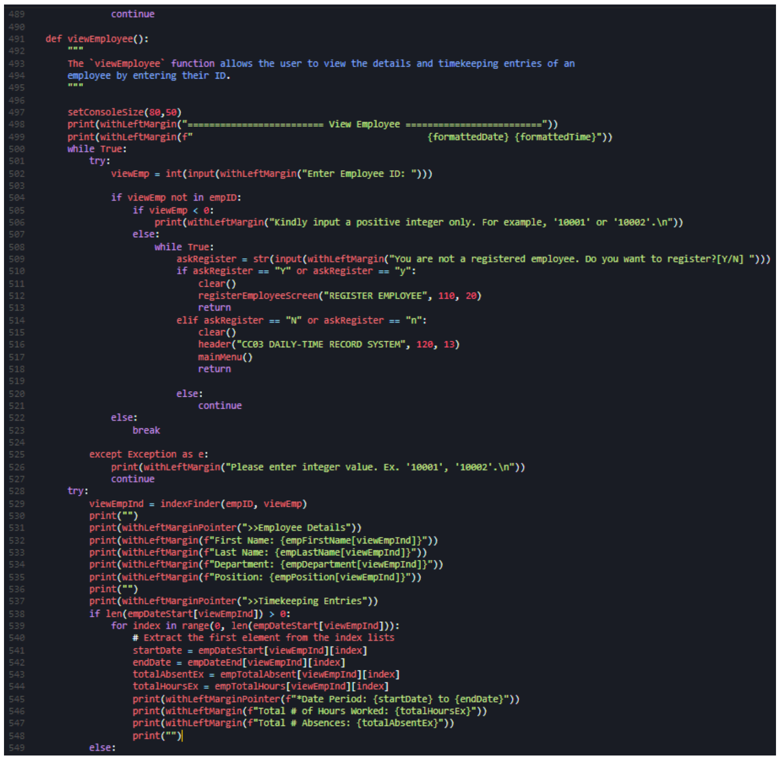
```python
                        while True:
                            timeOut = input(withLeftMargin("Enter Time-Out: "))
                            if timeOut == "A" or timeOut == "a":
                                break
                            else:
                                print(withLeftMargin("Enter 'A' if Absent"))
                                continue
                    else:
                        print(withLeftMargin("INCORRECT TIME-IN FORMAT. PLEASE FOLLOW 24-HOUR FORMAT(HH:MM)."))
                        continue
                    break
    empTotalHours[empInd].append(totalHours)
    empTotalAbsent[empInd].append(totalAbsent)
    totalAbsent = 0
    totalHours = 0
    while True:
        print("")
        ask = str(input(withLeftMargin("Do you want to place another timekeeping entry? (Y/N): ")))
        if ask == "Y" or ask == "y":
            print("")
            askEmployee()
            break

        elif ask == "N" or ask == "n":
            clear()
            header("CC03 DAILY-TIME RECORD SYSTEM", 120, 13)
            mainMenu()
            break

        else:
            print(withLeftMargin("Please enter 'Y' or 'N' only."))
            continue

def registerEmployeeScreen(title, width, height):
    """
    The function "registerEmployeeScreen" takes in parameters for the title, width, and height of a
    screen, and then prompts the user to enter employee details such as first name, last name,
    department, and position.
    """
    header(title, width, height)
    print(withLeftMarginPointer(">>Employee Details"))
    isIDIntAvailable()
    while True:
        registerFname = str(input(withLeftMargin("Enter First Name: "))).upper()
        splitParts = registerFname.split()
        if all(name.isalpha() for name in splitParts):
            empFirstName.append(registerFname)
            break
        else:
            print(withLeftMargin("Invalid Entry! Please enter a valid First Name."))
            continue
    while True:
        registerLname = str(input(withLeftMargin("Enter Last Name: "))).upper()
        splitLParts = registerLname.split()
        if all(name.isalpha() for name in splitLParts):
            empLastName.append(registerLname)
            break
        else:
            print(withLeftMargin("Invalid Entry! Please enter a valid Last Name."))
            continue
    isDepValid()
```



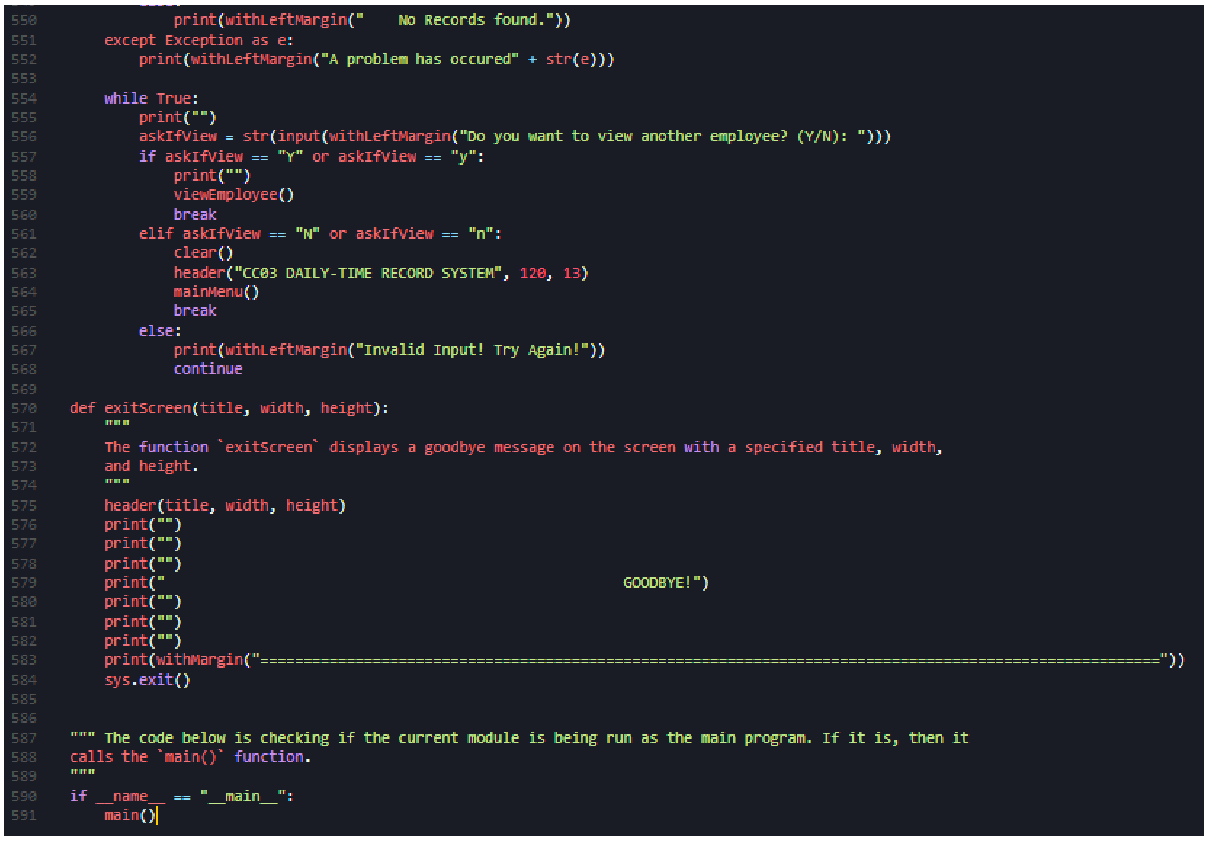
```python
    isPosValid()
    empTotalHours.append([])
    empTotalAbsent.append([])
    empDateStart.append([])
    empDateEnd.append([])
    whatTodo()

def isIDIntAvailable():
    """
    The function `isIDIntAvailable()` prompts the user to enter an employee ID as an integer, checks if
    the ID is already in use, and recursively calls itself until a unique ID is entered.
    """
    while True:
        try:
            registerID = int(input(withLeftMargin("Enter Employee ID: ")))
            if registerID not in empID:
                if registerID > 0:
                    empID.append(registerID)
                    return
                else:
                    print(withLeftMargin("Negative Integers are not valid! Try again!"))
                    continue
            else:
                print(withLeftMargin(f"ID '{registerID}' is not available. Please try again.\n"))
                print(withLeftMarginPointer(">>Employee Details"))
                continue
        except Exception as e:
            print(withLeftMargin("Please enter only integer value. Ex. '10001', '10002'.\n"))
            print(withLeftMarginPointer(">>Employee Details"))
            continue

def isDepValid():
    """
    The function `isDepValid()` prompts the user to enter a department (either 1 for Faculty or 2 for
    Non-Faculty) and validates the input.
    :return: The function isDepValid() does not explicitly return anything.
    """
    while True:
        try:
            registerDep = int(input(withLeftMargin("Enter Department (1)Faculty (2)Non-Faculty: ")))
            if ((registerDep < 3) and (registerDep > 0)):
                if registerDep == 1:
                    empDepartment.append("Faculty")
                    return
                else:
                    empDepartment.append("Non-Faculty")
                    return
            else:
                print(withLeftMargin("Please enter '1' or '2'only.\n"))
                continue
        except:
            print(withLeftMargin("Please enter either '1' for Faculty or '2' for Non-Faculty.\n"))
            continue

def isPosValid():
    """
    The function `isPosValid()` prompts the user to enter a position (1 for Full-Time or 2 for
    Part-Time) and validates the input.
    :return: The function isPosValid() does not explicitly return anything. However, it appends a value
    to the empPosition list if the input is valid.
    """
```



```python
    while True:
        try:
            registerPos = int(input(withLeftMargin("Enter Position (1)Full-Time (2)Part-Time: ")))
            if ((registerPos < 3) and (registerPos > 0)):
                if registerPos == 1:
                    empPosition.append("Full-Time")
                    return
                else:
                    empPosition.append("Part-Time")
                    return
            else:
                print(withLeftMargin("Please enter '1' or '2'only.\n"))
                continue
        except:
            print(withLeftMargin("Please enter either '1' for Full-Time or '2' for Part-Time.\n"))
            continue

def whatTodo():
    """
    The function `whatTodo()` prompts the user to register another employee and collects their details
    if the user chooses to do so, otherwise it returns to the main menu.
    """
    while True:
        print(withLeftMargin(""))
        askToDo = str(input(withLeftMargin("Do you want to register another employee? [Y/N]: ")))
        print(withLeftMargin(""))
        if askToDo == "Y" or askToDo == "y":
            print(withLeftMarginPointer(">>Employee Details"))
            isIDIntAvailable()
            while True:
                registerFname = str(input(withLeftMargin("Enter First Name: "))).upper()
                splitParts = registerFname.split()
                if all(name.isalpha() for name in splitParts):
                    empFirstName.append(registerFname)
                    break
                else:
                    print(withLeftMargin("Invalid Entry! Please enter a valid First Name."))
                    continue
            while True:
                registerLname = str(input(withLeftMargin("Enter Last Name: "))).upper()
                splitLParts = registerLname.split()
                if all(name.isalpha() for name in splitLParts):
                    empLastName.append(registerLname)
                    break
                else:
                    print(withLeftMargin("Invalid Entry! Please enter a valid Last Name."))
                    continue
            isDepValid()
            isPosValid()
            empTotalHours.append([])
            empTotalAbsent.append([])
            empDateStart.append([])
            empDateEnd.append([])
            whatTodo()
            return
        elif askToDo == "N" or askToDo == "n":
            clear()
            header("CC03 DAILY-TIME RECORD SYSTEM", 120, 13)
            mainMenu()
            return
        else:
```



```python
            continue

def viewEmployee():
    """
    The `viewEmployee` function allows the user to view the details and timekeeping entries of an
    employee by entering their ID.
    """

    setConsoleSize(80,50)
    print(withLeftMargin("======================== View Employee ========================"))
    print(withLeftMargin(f"                                                  {formattedDate} {formattedTime}"))
    while True:
        try:
            viewEmp = int(input(withLeftMargin("Enter Employee ID: ")))

            if viewEmp not in empID:
                if viewEmp < 0:
                    print(withLeftMargin("Kindly input a positive integer only. For example, '10001' or '10002'.\n"))
                else:
                    while True:
                        askRegister = str(input(withLeftMargin("You are not a registered employee. Do you want to register?[Y/N] ")))
                        if askRegister == "Y" or askRegister == "y":
                            clear()
                            registerEmployeeScreen("REGISTER EMPLOYEE", 110, 20)
                            return
                        elif askRegister == "N" or askRegister == "n":
                            clear()
                            header("CC03 DAILY-TIME RECORD SYSTEM", 120, 13)
                            mainMenu()
                            return

                        else:
                            continue
            else:
                break

        except Exception as e:
            print(withLeftMargin("Please enter integer value. Ex. '10001', '10002'.\n"))
            continue
    try:
        viewEmpInd = indexFinder(empID, viewEmp)
        print("")
        print(withLeftMarginPointer(">>Employee Details"))
        print(withLeftMargin(f"First Name: {empFirstName[viewEmpInd]}"))
        print(withLeftMargin(f"Last Name: {empLastName[viewEmpInd]}"))
        print(withLeftMargin(f"Department: {empDepartment[viewEmpInd]}"))
        print(withLeftMargin(f"Position: {empPosition[viewEmpInd]}"))
        print("")
        print(withLeftMarginPointer(">>Timekeeping Entries"))
        if len(empDateStart[viewEmpInd]) > 0:
            for index in range(0, len(empDateStart[viewEmpInd])):
                # Extract the first element from the index lists
                startDate = empDateStart[viewEmpInd][index]
                endDate = empDateEnd[viewEmpInd][index]
                totalAbsentEx = empTotalAbsent[viewEmpInd][index]
                totalHoursEx = empTotalHours[viewEmpInd][index]
                print(withLeftMarginPointer(f"*Date Period: {startDate} to {endDate}"))
                print(withLeftMargin(f"Total # of Hours Worked: {totalHoursEx}"))
                print(withLeftMargin(f"Total # Absences: {totalAbsentEx}"))
                print("")
        else:
```



```
            print(withLeftMargin("    No Records found."))
    except Exception as e:
        print(withLeftMargin("A problem has occured" + str(e)))

    while True:
        print("")
        askIfView = str(input(withLeftMargin("Do you want to view another employee? (Y/N): ")))
        if askIfView == "Y" or askIfView == "y":
            print("")
            viewEmployee()
            break
        elif askIfView == "N" or askIfView == "n":
            clear()
            header("CC03 DAILY-TIME RECORD SYSTEM", 120, 13)
            mainMenu()
            break
        else:
            print(withLeftMargin("Invalid Input! Try Again!"))
            continue

def exitScreen(title, width, height):
    """
    The function `exitScreen` displays a goodbye message on the screen with a specified title, width,
    and height.
    """
    header(title, width, height)
    print("")
    print("")
    print("")
    print("                                                     GOODBYE!")
    print("")
    print("")
    print("")
    print(withMargin("========================================================================================================"))
    sys.exit()


""" The code below is checking if the current module is being run as the main program. If it is, then it
calls the `main()` function.
"""
if __name__ == "__main__":
    main()
```