Gleb Lyubin

Abstract

This document contains my reflection of the project, as well as the pseudocode for displaying the menus and function headers.

design and reflection

McDermott’s system documentation

Table of Contents

**Pseudocode3**

**Python Function Headers5**

**Reflection**………………………………………………………………………………………………………………………………………….**7**

## Develop outlines of each menu item (and any sub menus) creating a pseudo-code representation of the display\_main\_menu and any display\_sub\_menu\_name functions.

display\_login\_screen(Username, Password)

Start

Print “McDermott’s Employee Software”

Print “Please login with your credentials”

Print “Username: “

Username 🡸INPUT()

Print “Password”

Password 🡸INPUT()

End

display\_main\_menu (admin, main\_menu\_choice)

Start

If admin = TRUE then

Print “McDermott’s Employee Software”

Print ”Select current action”  
Print “[1] Clock-in”

Print “[2] Clock-out”

Print “[3] Manage Employees”

Print “[4] Calculate Pay”

Else

Print “McDermott’s Employee Software”

Print ”Select current action”  
Print “[1] Clock-in”

Print “[2] Clock-out”

End if

main\_menu\_choice 🡸INPUT()

End

display\_employee\_management(manage\_employee)

Start

Print “McDermott’s Employee Software”

Print “[1] Add new employee”

Print “[2] Remove existing employee”

Print “[3] Edit Employee”

manage\_employee 🡸 INPUT()

End

display\_employee\_edit(employee\_id)

Start

Print “McDermott’s Employee Software”

Print “Input Employee ID”

employee\_ id 🡸 INPUT()

END

display\_clock\_in(clock\_in)

Start

Print “McDermott’s Employee Software”

Print “Enter the clock-in time (24hr format)”

clock\_in 🡸 INPUT()

End

display\_clock\_out(clock\_out)

Start

Print “McDermott’s Employee Software”

Print “Enter the clock-out time (24hr format)”

clock\_out 🡸 INPUT()

End

display\_select\_day(current\_day)

Start

Print “McDermott’s Employee Software”

Print “Enter the day”

Print “[1] Monday”

Print “[2] Tuesday”

Print “[3] Wednesday”

Print “[4] Thursday”

Print “[5] Friday”

Print “[6] Saturday”

Print “[7] Sunday”

current\_day 🡸 INPUT()

End

## Write python function headers (definitions) for each function, describing each of the input parameters (name and purpose) and then describe the output value (valid range and purpose) returned by the function when it is called.

def clear\_console():  
 pass #clears the text  
#end def  
  
def write\_to\_csv(filename, data, emp\_id, col): #filename = name of the csv file, data = data to write

#emp\_id = employee id (row), col = column to write in  
 pass # writes the specified text to a csv file  
#end def  
  
def read\_from\_csv(filename): # filename = name of the csv file  
 pass #returns an array of values  
#end def  
  
def calculate\_after\_tax(pre\_tax, emp\_id): #pre\_tax = income before tax, emp\_id = id of the employee to calculate income  
 pass # returns income after tax. Must not be less than 0  
#end def  
  
def calculate\_after\_deductions(gross, emp\_data): #gross = gross income, emp\_data = employee #record   
 pass # returns income after deductions. Must be >= 0  
#end def  
  
def validate\_data(data, acceptable\_data): #data = data to be validated, acceptable\_data = array of #acceptable values  
 pass # returns a boolean value  
#end def  
  
def clock\_in(clock\_in\_time, emp\_data): # clock\_in\_time = time in a 24hr format, emp\_data = #employee record

pass # writes the clock-in time to another csv file

#end def

def clock\_out(clock\_out\_time, day, emp\_data): # clock\_out\_time = time in 24hr format, day = day of #the week, emp\_data = employee data

pass #calculates the total work time and writes the data to the payroll file

#end def  
  
def calculate\_gross(emp\_data, hours): # emp\_data = employee record, hours = array of different #types of hours  
 pass # returns a float of pre-tax income. Must be >= 0  
#end def  
  
def authorise\_employee(username, password): # username = username entered, password = password entered  
 pass # returns true if authorised, false if not  
#end def

### def read\_emp\_data(filename, emp\_id, delimeter=”,”): #filename = name of the file to read from

### # emp\_id = row, delimeter = used to separate list elements. Default is set to ‘,’

### pass

### #end def

### def calc\_hours(emp\_data): # emp\_data = employee record

### pass # returns an array of sorted hours into categories

### #end def

### def remove\_employee(emp\_id): # emp\_id = id of the employee to remove

### pass # erases the employee records and re-formats the files

### #end def

### def add\_new\_employee(fname, lname, trate, role, superannuation, insurance): # fname = first name #of the new employee, lname = last name of the new employee, role = position of the new #employee, superannuation = selected superannuation, trate = tax rate, insurace = selected #insurance

### pass # adds the new employee record to all csv files

### #end def

### def delete\_lines\_except(filename, line\_num): # filename = name of the file, line\_num = number of #the line to save

### pass # deletes all lines in a text file except for the specified line

### #end def

### def validate\_hours\_decimal(hour): # hour = hour in decimal format

### pass # returns true if the decimal representing the hour is valid

### #end def

### def validate\_hours (hour): # hour = hour in the 24hr format

### pass # returns true if the hour is a real hour

### #end def

### def list\_to\_string(list): # appends all list elements into a string

### pass # returns a string containing all the list elements

### #end def

### def append\_to\_csv(filename, data): # filename = name of the file, data = data to be appended

### pass # appends the data to the specified text file

### #end def

## your reflections upon your own performance against each of the dot-points in this outline, including suggestions that you would try to incorporate into the next version of this application.

I think I did well to address the dot-points in the outline. The program is fully capable of managing employees without having to use external text files. Employees are offered a secure way of clocking in and out.

In the next version of the application, I would give users the opportunity to change their username and password. A database would be much better suited for this task, so I would also opt for using SQLite.

Additionally, I would allow the administrator to change some of the values stated in the task description, such as the pay rates. In this version, I simply hard-coded the values which is a terrible practice; however, I was assuming that there would be no need to change those values.

In the current version, it is difficult to read the values when selecting ‘display all employee data’. In future iterations, I would like the data to appear in an easy to read table or have a graphical user interface.

The current version of the program uses a lot of iteration constructs, which I feel could be replaced by more efficient alternatives. I did use the enumerate() function in python in some cases to hopefully increase the performance, but it is still not ideal. While the iteration constructs do not significantly impact the performance now, if more employees were added, the program would be much slower.

In some places, I have used global variables which are considered to be a bad practice in python. I would refactor my code to eliminate such variables to improve the scalability of the project.

Some of the algorithms are extremely inefficient, such as the one used in calc\_hours() which uses many iteration and selection constructs, and could definitely be compressed. I hope that I can improve my Python skills to code more efficient algorithms as time goes on.

Taking on an object-oriented approach for this project would be ideal, and would help solve the issue of scalability through administrator and team member classes.

Overall, I think I’ve developed a good prototype for a small restaurant, but the software does quickly show limitations when the users decide to customise different aspects, such as the role and day specific pay rates or login credentials. I consider it a good foundation in terms of broad design for the future iterations of the program that would account for scalability.