

Figure 1

Foundation Certificate for Higher Education

Module: DOC 333 Introduction to Programming Principles

Module Leader: Mr. Sudarshan Welihinda

Assignment Type: Individual

Submission Date: 2023.12.11

Student ID: 20231264

Student Name: Ranuga Disansa Belpa Gamage

Student Email: [ranuga.20231264@iit.ac.lk](mailto:ranuga.20231264@iit.ac.lk)

Acknowledgement

I want to express my appreciation to those who have helped complete this report.

I have been able to complete this report because of the Academic advisor Mr. Sudarshan Welihinda for their invaluable support, mentorship, and feedback and the faculty members of Doc 333 (Introduction to Programming Principles).

Additionally, I would like to thank my family members for their unwilling encouragement and help throughout the report.

Thank you.

Table of Contents

[1. Introduction 6](#_Toc153146390)

[1.1 The Problem 6](#_Toc153146391)

[1.2 The Solution 6](#_Toc153146392)

[2. Algorithm 7](#_Toc153146393)

[2.1 PesudoCode 7](#_Toc153146394)

[2.3 Verification Functions 12](#_Toc153146395)

[2.3.1 date\_vertification() 12](#_Toc153146396)

[2.3.3 project\_code\_vertification() 14](#_Toc153146397)

[2.3.4 check\_if\_int() 14](#_Toc153146398)

[2.4 Main Functions 14](#_Toc153146399)

[2.4.1 remove\_completed\_projects() 14](#_Toc153146400)

[2.4.2 create\_project() 17](#_Toc153146401)

[2.4.3 update\_project\_details() 18](#_Toc153146402)

[2.5 Helper Functions 20](#_Toc153146403)

[2.5.1 main() 20](#_Toc153146404)

[2.6 Data Structures 20](#_Toc153146405)

[2.6.1 Introduction to Data Structures 20](#_Toc153146406)

[2.6.2 Data Structures Used in the Program 20](#_Toc153146407)

[2.7 Explanation 22](#_Toc153146408)

[2.7.1 Add a new project to existing projects. 22](#_Toc153146409)

[2.7.2 Removing a Completed project from existing projects 23](#_Toc153146410)

[2.7.3 Adding new workers to the available worker's group 23](#_Toc153146411)

[2.7.4 Updating Details to an ongoing project 24](#_Toc153146412)

[2.7.5 Project Statistics 24](#_Toc153146413)

[2.7.6 Exiting the Program 25](#_Toc153146414)

[3 Assumptions 25](#_Toc153146415)

[4 Python Code 26](#_Toc153146416)

[5 Test Cases 32](#_Toc153146417)

[5.1 Main Menu 32](#_Toc153146418)

[5.1.1 Test 1 32](#_Toc153146419)

[5.1.2 Test 2 32](#_Toc153146420)

[5.1.3 Test 3 33](#_Toc153146421)

[5.1.4 Test 4 34](#_Toc153146422)

[5.1.5 Test 5 35](#_Toc153146423)

[5.1.6 Test 6 36](#_Toc153146424)

[5.1.7 Test 7 37](#_Toc153146425)

[5.2 Add a New Project 37](#_Toc153146426)

[5.2.1 Test 1 37](#_Toc153146427)

[5.2.2 Test 2 38](#_Toc153146428)

[5.2.3 Test 3 38](#_Toc153146429)

[5.2.4 Test 4 39](#_Toc153146430)

[5.2.5 Test 5 40](#_Toc153146431)

[5.2.6 Test 6 41](#_Toc153146432)

[5.3 Remove a Project 42](#_Toc153146433)

[5.3.1 Assuming the project below has been added 42](#_Toc153146434)

[5.3.2 Test 1 42](#_Toc153146435)

[5.3.3 Test 2 43](#_Toc153146436)

[5.3.4 Test 3 43](#_Toc153146437)

[5.4 Add new Workers 44](#_Toc153146438)

[5.4.1 Test 1 44](#_Toc153146439)

[5.4.2 Test 2 44](#_Toc153146440)

[5.4.3 Test 3 45](#_Toc153146441)

[5.4.4 Test 4 45](#_Toc153146442)

[5.5 Update Project Details 46](#_Toc153146443)

[5.5.1 Assuming the project below has been added 46](#_Toc153146444)

[5.5.2 Test 1 46](#_Toc153146445)

[5.5.3 Test 2 47](#_Toc153146446)

[5.5.4 Test 3 47](#_Toc153146447)

[5.5.5 Test 4 48](#_Toc153146448)

[5.5.6 Test 5 49](#_Toc153146449)

[5.5.7 Test 6 49](#_Toc153146450)

[5.5.8 Test 7 50](#_Toc153146451)

[5.5.9 Test 8 51](#_Toc153146452)

[5.6 Project Statistics 52](#_Toc153146453)

[5.6.1 Assuming the project below has been added from Choice 3 52](#_Toc153146454)

[5.6.2 Assuming the project below has been added from Choice 1 52](#_Toc153146455)

[5.6.3 Test 1 52](#_Toc153146456)

[5.6.4 Test 2 53](#_Toc153146457)

[5.7 Exit 54](#_Toc153146458)

[5.7.1 Test 1 54](#_Toc153146459)

[6. Conclusion 54](#_Toc153146460)

List of Figures

[Figure 1 0](file:///D:\University\IIT\DOC333\DOC333-CW-Sem1\Report\20231264.docx#_Toc153143619)

[Figure 2 12](#_Toc153143620)

[Figure 3 13](#_Toc153143621)

[Figure 4 14](#_Toc153143622)

[Figure 5 14](#_Toc153143623)

[Figure 6 15](#_Toc153143624)

[Figure 7 17](#_Toc153143625)

[Figure 8 19](#_Toc153143626)

[Figure 9 20](#_Toc153143627)

[Figure 10 21](#_Toc153143628)

[Figure 11 21](#_Toc153143629)

[Figure 12 22](#_Toc153143630)

[Figure 13 22](#_Toc153143631)

[Figure 14 23](#_Toc153143632)

[Figure 15 23](#_Toc153143633)

[Figure 16 24](#_Toc153143634)

[Figure 17 24](#_Toc153143635)

[Figure 18 25](#_Toc153143636)

[Figure 19 25](#_Toc153143637)

[Figure 20 26](#_Toc153143638)

[Figure 21 27](#_Toc153143639)

[Figure 22 27](#_Toc153143640)

[Figure 23 28](#_Toc153143641)

[Figure 24 28](#_Toc153143642)

[Figure 25 29](#_Toc153143643)

[Figure 26 29](#_Toc153143644)

[Figure 27 30](#_Toc153143645)

[Figure 28 31](#_Toc153143646)

[Figure 29 31](#_Toc153143647)

[Figure 30 32](#_Toc153143648)

[Figure 31 33](#_Toc153143649)

[Figure 32 34](#_Toc153143650)

[Figure 33 35](#_Toc153143651)

[Figure 34 36](#_Toc153143652)

[Figure 35 37](#_Toc153143653)

[Figure 36 37](#_Toc153143654)

[Figure 37 38](#_Toc153143655)

[Figure 38 38](#_Toc153143656)

[Figure 39 39](#_Toc153143657)

[Figure 40 40](#_Toc153143658)

[Figure 41 41](#_Toc153143659)

[Figure 42 42](#_Toc153143660)

[Figure 43 43](#_Toc153143661)

[Figure 44 43](#_Toc153143662)

[Figure 45 44](#_Toc153143663)

[Figure 46 44](#_Toc153143664)

[Figure 47 45](#_Toc153143665)

[Figure 48 45](#_Toc153143666)

[Figure 49 46](#_Toc153143667)

[Figure 50 47](#_Toc153143668)

[Figure 51 47](#_Toc153143669)

[Figure 52 48](#_Toc153143670)

[Figure 53 48](#_Toc153143671)

[Figure 54 49](#_Toc153143672)

[Figure 55 50](#_Toc153143673)

[Figure 56 51](#_Toc153143674)

[Figure 57 51](#_Toc153143675)

[Figure 58 53](#_Toc153143676)

[Figure 59 53](#_Toc153143677)

[Figure 60 54](#_Toc153143678)

# 1. Introduction

## 1.1 The Problem

The company “XYZ” which undertakes large housing construction projects needs an information system to maintain details of the projects they undertake. This system should keep details of all ongoing projects, the details and the number of workers available to assign to a new project. Before undertaking a project, the company makes sure they have enough workers if not the company doesn’t undertake the project. Once the project is finished it is taken out of the ongoing projects and assigned to completed projects and the workers are released when the project is completed.

## 1.2 The Solution

The solution that is created is for the problem to maintain the details of projects the construction company named “XYZ” undertakes. The solution is an information system built using Python programming language.

# Algorithm

The solution which is implemented by Python Programming language is stated below in the form of algorithm steps and with an explanation of how each aspect of the program functions.

## 2.1 PesudoCode

1. Start
2. # importing packages

* IMPORT datetime

1. # initialization of variables

* SET company\_name TO “XYZ Company.”
* SET workers TO 0
* SET choice TO 0
* SET all\_projects TO []
* SET completed\_projects TO []
* SET execute TO True
* SET project\_names TO []
* SET possible\_inputs TO [“ongoing”, “completed”, “on hold”]
* SET statistics\_list TO [0] \* len(possible\_inputs)
* SET redirect\_choice TO False
* SET redirect\_to TO None

1. define function menu(redirect,to,company\_name,msg):

* SET main\_menu TO company\_name + ”””

Main Menu

1. Add a new project to existing projects.
2. Remove a completed project from existing projects
3. Add new workers to available workers group
4. Update details on ongoing projects
5. Project Statistics
6. Exit

“””

* OUTPUT “Redirecting…” if redirect is True else main\_menu
* RETURN to if redirect is True else INPUT user choice

1. define function remove\_completed\_projects(code\_of\_project,every\_project,workers\_tot,stats\_list,complete\_projects,possible\_stats):

* TRY
* SET index\_of\_project TO project\_names.index(code\_of\_project)
* SET date\_time TO datetime.datetime.now()
* SET actual\_end\_date TO date\_time.strftime(“%m/%d/%Y”)
* SET code\_of\_project,clients\_name,start\_date,expected\_end\_date,number\_of\_workers,\_,index TO every\_project[index\_of\_project]
* SET completed\_project\_details TO [code\_of\_project,clients\_name,start\_date,expected\_end\_date,number\_of\_workers,actual\_end\_date]
* IF old\_project\_status EQUALS “ongoing”
* SET workers\_tot TO workers\_tot + num\_of\_workers
* SET stats\_list[index] TO stats\_list[index] - 1
* SET stats\_list[possible\_stats.index(“completed”)] TO stats\_list[possible\_stats.index(“completed”)] + 1
* APPEND completed\_project\_details TO completed\_projects
* DELETE every\_project[index\_of\_project]
* DELETE project\_names[index\_of\_project]
* RETURN (True, “Successful removed completed projects.”,workers\_tot,status\_list,completed\_projects,every\_project,project\_names)
* EXCEPT Exception as e
* RETURN (False,e,workers\_tot,status\_list,completed\_projects,every\_project,project\_names)

1. define function create\_project(status\_list,index,code\_of\_project,clients\_name,start\_date,expected\_end\_date,number\_of\_workers,project\_status,workers\_tot,project\_names,all\_projects)

* TRY
* SET status\_list[index] TO status\_list[index] + 1
* SET project\_date TO [code\_of\_project,clients\_name,start\_date,expected\_end\_date,number\_of\_workers,project\_status,index]
* IF project\_status EQUALS “ongoing” and number\_of\_workers > workers\_tot
* RETURN (False,” There is not enough workers”, workers\_tot,all\_projects,project\_names)
* IF project\_status EQUALS “ongoing”
* SET workers\_tot TO workers\_tot – number\_of\_workers
* APPEND code\_of\_project TO project\_names
* APPEND project\_data to all\_projects
* RETURN (True, “Successfully created a new project”, workers\_tot,all\_projects,project\_names)
* EXCEPT Exception as e
* RETURN (False, e, workers\_tot,all\_projects,project\_names)

1. define function update\_project\_details(status\_list,index,previous\_index,code\_of\_project,clients\_name,start\_date,expected\_end\_date,number\_of\_workers,project\_status,current\_workers,workers\_tot,previous\_project\_status)

* TRY
* IF number\_of\_workers > workers\_tot + (current\_workers if previous\_project\_stautus EQUALS “ongoing” else 0) AND project\_status EQUALS “ongoing”
* RETURN (False, “Workers chosen are too much”, workers\_tot)
* IF project\_status EQUALS “ongoing”
* SET workers\_tot TO workers\_tot – number\_of\_workers
* IF previous\_project\_status EQUALS “ongoing”
* SET workers\_tot TO workers\_tot + current\_workers
* SET status\_list[index] TO status\_list[index] + 1
* SET status\_list[previous\_index] TO status\_list[previous\_index] – 1
* SET project\_data TO [code\_of\_project,clients\_name,start\_date,expected\_end\_date,number\_of\_workers,project\_status,index]
* SET index TO project\_names.index(code\_of\_project)
* SET all\_projects[index] TO project\_data
* RETURN (True, “Project details updated successfully”, workers\_tot)
* EXCEPT Exception as e:
* RETURN (False,e,workers\_tot)

1. define function date\_vertification(msg)

* TRY
* SET date TO INPUT(msg)
* SET splitted\_date TO date.split(date[2] if len(date) > 3 else “ “)
* IF len(splitted\_date) != 3
* OUTPUT “Enter a valid format of the date…!”
* SET month TO splitted\_date[0]
* SET date TO splitted\_date[1]
* IF month > 12
* OUTPUT “Enter a valid month! “
* RETURN date\_vertification(msg)
* IF date > 31
* OUTPUT “Enter a valid date !”
* RETURN date\_vertification(msg)
* RETURN date
* EXCEPT
* RETURN date\_vertification(msg)

1. define function project\_status\_vertification(msg,update\_status)

* SET project\_state TO INPUT(msg).replace(“ “,””).lower()
* IF project\_state NOT IN possible\_inputs:
* OUTPUT “The entered project status is incorrect”
* IF update\_status IS True
* SET statistics\_list[possible\_inputs.index(project\_state)] TO statistics\_list[possible\_inputs.index(project\_state)] + 1
* RETURN (project\_state,statistics\_list,possible\_inputs.index(project\_state))

1. define function project\_code\_vertification(msg,project\_codes)

* SET project\_code TO INPUT(msg)
* IF project\_code IN project\_codes
* OUTPUT “Project code already exists”
* RETURN project\_code\_vertification(msg,project\_codes)
* RETURN project\_code

1. define function check\_if\_int(msg)

* TRY
* RETURN int(INPUT(msg)
* EXCEPT
* OUTPUT “The msg entered was not an integer”
* RETURN check\_if\_int(msg)

1. while execute

* SET choice TO menu(redirect\_choice,redirect\_to)
* SET redirect\_choice TO False
* SET redirect\_to TO None
* IF choice EQUALS “1”
* OUTPUT company\_name + “Add a new project”
* SET code\_of\_project TO project\_code\_vertification(“Project Code : “, project\_names)
* IF code\_of\_project EQUALS “0”
* CONTINUE
* SET clients\_name TO INPUT(“Clients Name : “)
* SET start\_date TO date\_vertification(“Start Date (MM/DD/YYYY) : ”)
* SET expected\_end\_date TO date\_vertification(“Expected end date (MM/DD/YYYY)”)
* SET number\_of\_workers TO check\_if\_int(“Numbers of Workers : “)
* SET project\_status, status\_list, index TO project\_status\_vertification()
* SET save to INPUT(“Do you want to save the project (Yes/No)”)
* IF save.upper() EQUALS “YES”
* SET execution\_status, response\_msg, workers TO create\_project(status\_list,index,code\_of\_project,clients\_name,start\_date,expected\_end\_date,number\_of\_workers,project\_status,workers,all\_projects,statistics\_list,possible-Inputs)
* OUTPUT response\_msg + execution\_status
* ELSE:
* OUTPUT “The project was \*not\* saved !”
* ELSEIF choice EQUALS “2”
* OUTPUT company\_name + “Remove completed project”
* SET code\_of\_project TO INPUT(“Project Code : “)
* IF code\_of\_project NOT IN project\_names
* OUTPUT “The project does not exist”
* CONTINUE
* SET save TO INPUT(“Do you want to remove the project (Yes/No) ? “)
* IF save.upper() EQUALS “YES”
* SET Execution\_status,response\_msg,workers,status\_list,completed\_projects,every\_project, project\_names TO remove\_completed\_projects(code\_of\_project,all\_projects,workers,statistics\_list,completed\_projects,possible\_inputs)
* OUTPUT response\_msg + execution\_status
* ELSE
* OUTPUT “The project was not removed”
* ELSEIF choice EQUALS “3”
* OUTPUT company\_name + “Add new workers”
* SET new\_no\_of\_workers TO check\_if\_int(“Number Workers to Add : “)
* IF new\_no\_of\_workers < 0
* OUTPUT “Workers must be more than 0”
* IF save.upper() EQUALS “YES”
* SET workers TO workers + new\_no\_of\_workers
* ELSE
* OUTPUT “Workers were not added”
* ELSEIF choice EQUALS “4”
* OUTPUT company\_name + “Update Project Details”
* SET code\_of\_project TO INPUT(“Project Code : “)
* IF code\_of\_project NOT IN project\_names
* OUTPUT “There isn’t a project with the mentioned project code…!”
* IF code\_of\_project.replace(“ “,””) EQUALS “0”
* CONTINUE
* SET clients\_name TO INPUT(“Clients Name : “)
* SET start\_date TO date\_vertification(“Start Date (MM/DD/YYYY) : “)
* SET expected\_end\_date TO date\_vertification(“Excepected End Date (MM/DD/YYYY) : “)
* SET number\_of\_workers TO check\_if\_int(“Numbers of Workers : “)
* SET project\_status,status\_list,index TO project\_status\_vertification()
* SET save TO INPUT(“Do you want to update the project details (Yes/No)”)
* IF save.upper() EQUALS “YES”
* SET current\_workers,previous\_project\_status,previous\_index TO all\_projects[project\_names.index(code\_of\_project)][4:]
* SET execution\_status,response\_msg,workers TO update\_project\_details(status\_list,index,previous\_index,code\_of\_project,client\_name,start\_date,expected\_end\_date,number\_of\_workers,project\_status,current\_workers,workers,previous\_project\_status)
* OUTPUT response\_msg + execution\_status
* ELSE
* OUTPUT “The project was not updated”
* ELSEIF choice EQUALS “5”
* OUTPUT company\_name + “Project Statistics”
* FOR idx, item IN enumerate(possible\_inputs)
* OUTPUT “Number of “ + item + “projects : “ + statistics\_list[idx]
* OUTPUT “Number of Available Workers: “ + workers
* SET add\_project TO INPUT(“Do you want to add the project”)
* IF add\_project.upper() EQUALS “YES”
* SET redirect\_choice, redirect\_to TO True, “1”
* ELSEIF choice EQUALS “6”
* SET execute TO False
* ELSE
* OUTPUT “Please enter a valid choice!”

## 2.3 Verification Functions

### 2.3.1 date\_vertification()

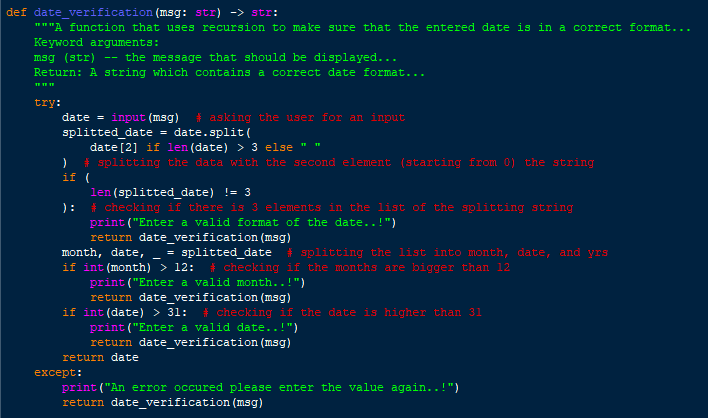


Figure 2

`date\_vertification()` is a function that uses recursion to make sure the entered date is in the correct format. It has 1 argument which is `msg` which is the message that is displayed to the user. A string containing the correct date format is returned. The function works by first asking the user for a date, and then the data is split by the second character (3’rd letter), for example, “12**/**21/2008” The second element which is “/” will be used to split and it is taken in consideration that the string may be smaller than 3 letters so if it is then an empty string will be used. Then the length of the spliced list is checked and if it is not 3 then the `date\_vertification()` function calls itself (recursion). Then after that, the month and date are extracted from the list. Finally, the month and date are checked if they are higher than 12 and higher than 31 respectively, and if all the arguments are passed then the date is returned.

2.3.2 project\_status\_vertification()

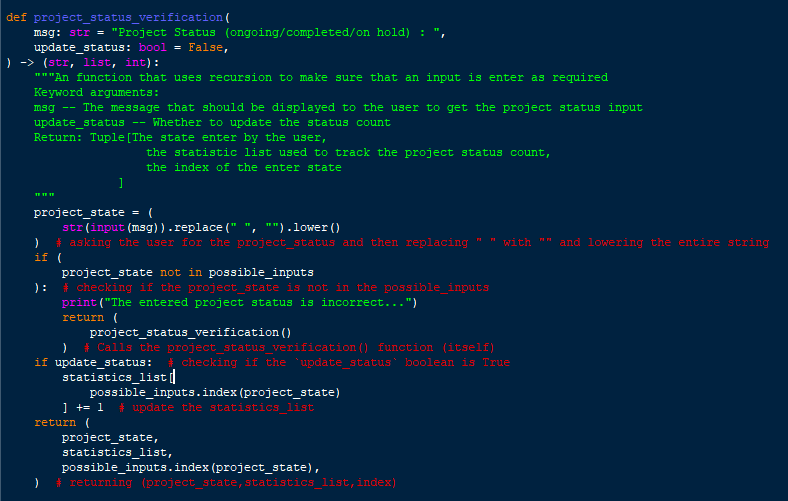


Figure 3

The `project\_status\_vertification()` function checks whether the project status that is entered is allowed and if not it uses recursion to make sure that the user enters the allowed status. It has 2 arguments which is `msg` which is the message that should be displayed to the user and `update\_status` which is a Boolean argument that if True the statistics\_list is updated with the status that was entered. First, the user has displayed a message which they need to respond to then the “ “ (blank spaces) are replaced with “and the entire message is lowered, then it is checked if the project\_state entered is not in the possible\_inputs list, if it is not then the function is calling itself (recursion) and if the project\_state is in the possible\_input the `statistics\_list` is updated and then the following is returned : (project\_state, statistics\_list, possible\_inputs.index(project\_state)) => (The project state, the statistics list that is used for the choice `5`, the index of the project state in the list possible\_inputs), an example would be: [“ongoing”,[2,1,5],1]

### 2.3.3 project\_code\_vertification()

A computer screen shot of text

Description automatically generated

Figure 4

`project\_code\_vertification()` is a function which uses recursion to make sure the project\_code entered does not exist already. The parameters are `msg` which is the message that should be displayed to the user and `project\_codes` which is the list of project\_codes where the function checks if the entered project code exists or not, and finally if the project code does not exist it is returned.

### 2.3.4 check\_if\_int()

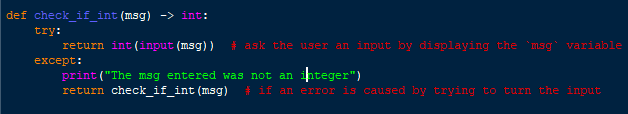


Figure 5

`check\_if\_int()` function uses recursion with having 1 parameter `msg` which is what is displayed to the user then the message is displayed and the function tries to return the message by trying to convert the inputted data into an integer and if an error is caused then a message saying “The msg entered was not an integer” is displayed and then the function calls itself (recursion).

## 2.4 Main Functions

### 2.4.1 remove\_completed\_projects()

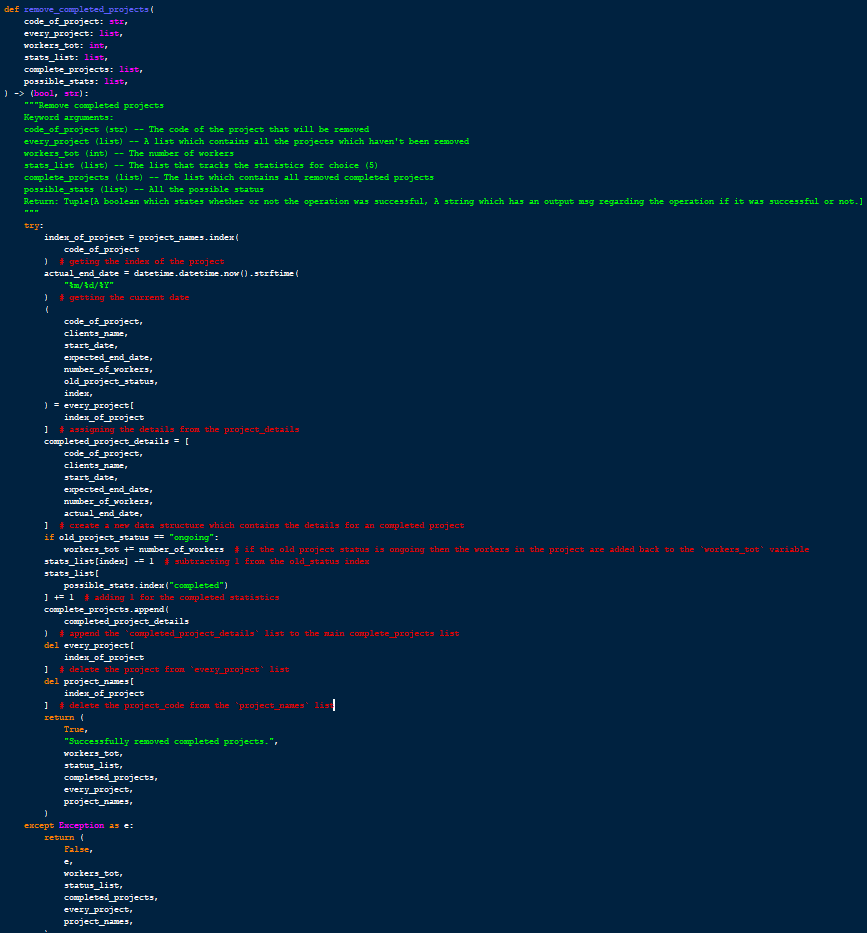


Figure 6

The `remove\_completed\_projects()` function has parameters which are code\_of\_project,every\_project,workers\_tot,statst\_list,complete\_projects, and possible\_stats which respectively contain the code of the project to be removed, list which contains all the projects which haven’t been removed, number of workers, statistics list which tracks the project statuses for choice `5`, list which contains all removed completed projects, all the possible status. The function gets the current date using `datetime` library and gets `code\_of\_project`, `clients\_name`, `start\_date`, ` expected\_end\_date `,`number\_of\_workers,` index` from the every\_project specific project code, then a list called `completed\_project\_details` is created using: [code\_of\_project,clients\_name,start\_date,expected\_end\_date,number\_of\_workers,actual\_end\_date] then the workers that were in the project if the status was “ongoing” is added back to the workers\_tot. Then the statistics list is updated by subtracting the old status and adding to the completed column, then the `completed\_project\_details` is added to the `completed\_projects` list, and the data is deleted from `every\_project` and `project\_names`, a tuple is returned which contains: (a Boolean which states whether the function was successfully or not, a message that will be displayed to the user, workers\_tot,status\_list,completed\_projects,every\_project,project\_names). there is a try and except just in case an error is caused in turn return (False, error, workers\_tot,status\_list,completed\_projects,every\_project,project\_names)

### 2.4.2 create\_project()

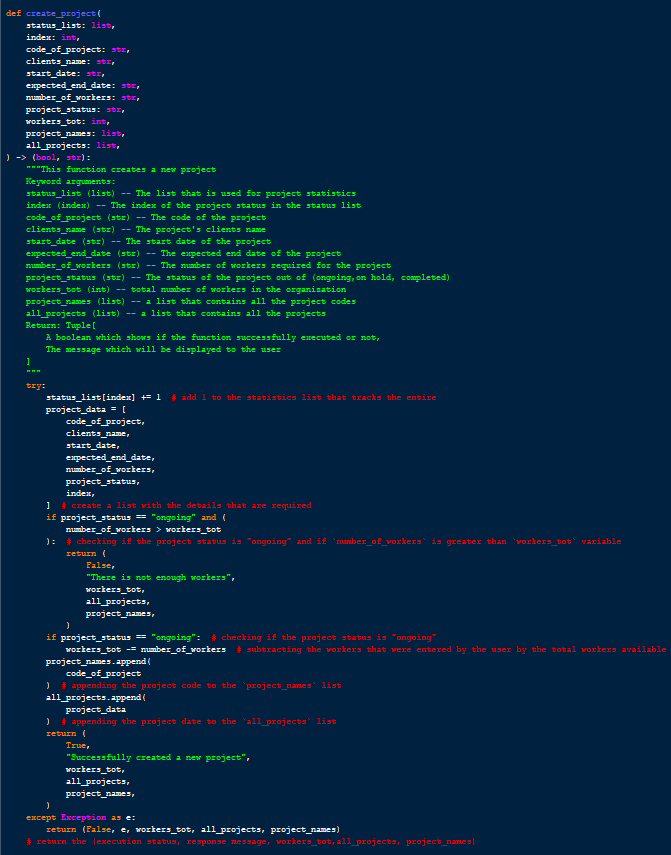


Figure 7

The function `create\_project()` contains the functionality to create a new project, with parameters of status\_list, index,code\_of\_project,clients\_name,start\_date,expected\_end\_date,number\_of\_workers,project\_status,workers\_tot,project\_names,all\_projects and they contain the following in respective order: list which contains the project statistics, index of the project status entered by the user in the status\_list, code of the project, clients name in the project, the start date of the project, expected end date of the project, status of the project from (“ongoing”,” on hold”,” completed”), the total number of workers available, list of all the project codes, contains all the projects. The function first updates the status\_list for the specific project status that was entered by the user, then a list is created with the following information: code\_of\_project,clients\_name,start\_date,expected\_end\_date,number\_of\_workers,project\_status, index then with an if statement we check whether the status is “ongoing” and if the there are enough workers to be assigned and if there isn’t enough then (False, “There is not enough workers”, workers\_tot,all\_projects,project\_names) is returned if not we check whether the project\_status is “ongoing” and if it is then we subtract the number of workers from the total worker count, then the `code\_of\_project` is appended to `project\_names` and `project\_data` is appended to `all\_projects` and finally (True, “Successfully created a new project”, workers\_tot, all\_project,project\_names) is returned, there is an try and except just in case an error is caused in turn return (False, error, workers\_tot,all\_projects,project\_names)

### 2.4.3 update\_project\_details()

The function `update\_project\_details()` updates the project details of an already existing project, it takes 12 which are: status\_list, index,previous\_index,code\_of\_project,clients\_name,start\_date,expected\_end\_date,number\_of\_workers,project\_status,current\_workers,worekrs\_tot,previous\_project\_status which contain the following: the list that contains the project statistics data, index of the new project status in the `status\_list`, previous\_index which contains the previous project status in the `status\_list`, the project code of the project, the updated client name, the updated start date, the updated expected end date, the updated number of workers, the workers total, the preiovious\_project\_status. First using an if statement the program checks whether there are enough workers and it is only checked for “ongoing” projects, then next if the project\_status is “ongoing” the number\_of\_workers is subtracted by the workers total, and if the previous\_project\_status is “ongoing” then the workers before they were updated are added to the workers total, then the statistics list (status\_list) is updated and then the project details are updated, finally (True, “Project details updated successfully”, workers\_tot) is returned and in case of an error then (False, e,workers\_tot) is returned.

A screen shot of a computer program

Description automatically generated

Figure 8

## 2.5 Helper Functions

### 2.5.1 main()

A screenshot of a computer program

Description automatically generated

Figure 9

The `menu()` function displays the main\_menu and asks the user to enter their choice. But also, it allows the user to be redirected with the use of the parameters `redirect` and `to` where the `redirect` parameter is a Boolean and if True then instead of the main and the returned value if the `to` parameter instead of asking the user for an input.

## Data Structures

### Introduction to Data Structures

The data structure that was used in the program was a list which is a built-in abstract data type with a sequence or collection of elements, which may be any data type. The data inside a list starts with the index of ‘0’. Their syntax is ‘[item1, item2.

,...]’. Lists can be edited or modified compared to similar data types, such as sets.

### Data Structures Used in the Program

A variety of variables were created to store different types of data, the following are the variables and their purpose.

#### 2.6.2.1 all\_projects

The list that contains all the projects that have been added by the 1st choice and that haven’t been removed using the 2nd choice. It uses a nested loop structure with the structure of: [code\_of\_project, clients\_name,start\_date,expected\_end\_date,number\_of\_workers,project\_status, index]

many projects it would be displayed in the following manner: [

[1.” Ranuga”,”12/12/2008”,”01/01/2009”,10,”ongoing”,0],

[2,” Devin”,”06/16/2017”,”05/12/2020”,20,”onhold”2],

[3,” Joe”,”01/01/2000”,”10/05/2010”,”50”,” completed,1],

…]

A blue screen with white text

Description automatically generated

Figure 10

#### 2.6.2.2 completed\_projects

This list contains all the projects that were removed from the 2nd choice, it uses the nested loop concept to store the data with each list that is stored inside the completed\_project list being in the following structure: [code\_of\_project, clients\_name,start\_date,expected\_end\_date,number\_of\_workers,actual\_end\_date]

examples of many projects being stored: [

[5,” Tharun”,”10/10/2015”,”12/12/2015”,10,”01/25/2016”],

[6,” Sandaru”,”02/10/2013”,”04/10/2017”,100,”05/14/2017”],

[7,” Kalin”,”05/18/1990”,”04/19/2000”,250,”06/29/2000”],

…]

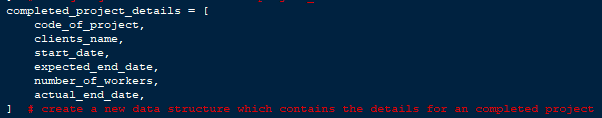


Figure 11

#### 2.6.2.3 project\_names

project\_names list contains all of the project\_codes of the projects that are added by the 1st choice, it is stored in the following order: [project\_code1, project\_code2,…]. This is used to find the index of a project in `all\_projects`, it serves as a link to help the program find the project details just by using the project.

#### 2.6.2.4 possible\_inputs

The list possible\_inputs does not change over time and it contains a constant list of values of: [“ongoing”,” completed”,” onhold”], this list was made so that the statuses can be scaled for example adding another status would not take must more effort. `possible\_inputs` contain all the statuses which are allowed within the program.



Figure 12

#### 2.6.2.5 statistics\_list

The statistics\_list is directly dependent on `possible\_inputs` it has the same no. of elements as the `possible\_inputs` list, it is made with the scalability of new statuses in mind. It is used to track the statistics of each project\_status listed in the `possible\_inputs` list.

## 2.7 Explanation

### 2.7.1 Add a new project to existing projects.

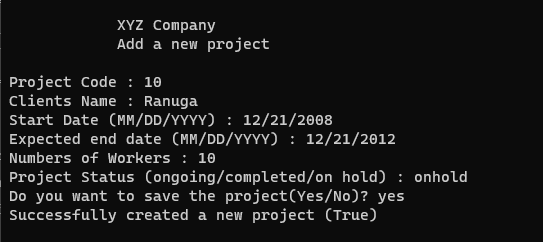


Figure 13

When the user enters the choice `1` the program asks the user for the Project Code which is directed through a function `project\_code\_vertification()` which makes sure that the project code is unique, for the start date and expected end date, function `date\_vertification()` is used which makes sure the date entered is in the correct format of MM/DD/YYYY. When asking the user for the project status it is passed through a function `project\_status\_vertification()` which checks whether the status entered was either “ongoing”, “completed” or “on hold”.

### 2.7.2 Removing a Completed project from existing projects

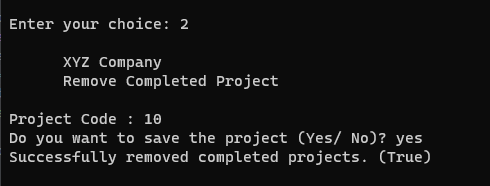


Figure 14

When the user enters the choice `2` the program asks for the project code which needs to be removed, and it makes sure that the project code exists in the list `project\_names`.

### 2.7.3 Adding new workers to the available worker's group

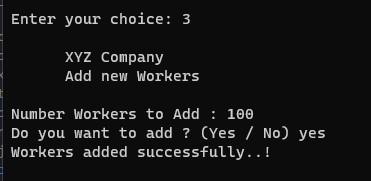


Figure 15

When the user chooses the choice `3` the user is asked for the number of workers that they want to add and the function `check\_if\_it()` is used to make sure the user enters an integer value.

### 2.7.4 Updating Details to an ongoing project

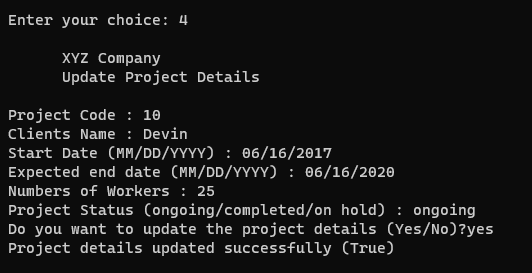


Figure 16

When the user chooses the choice `4` the user is asked for the project code and other details such as Clients Name, Start Date, Expected end Date, Number of workers, and Project Status which uses functions such as `date\_vertification()`, `check\_if\_int()` and `project\_status\_vertification()`, and this updates the data in the main list `all\_projects`

### 2.7.5 Project Statistics

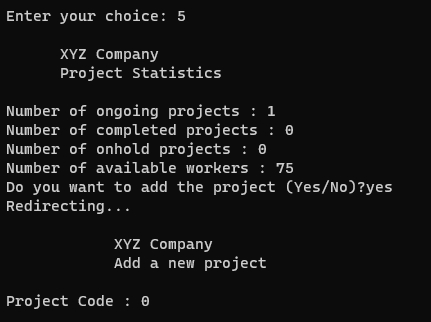


Figure 17

When the user chooses the choice `5` it displays the project statistics such as ongoing, completed, hold projects and the available workers. It also allows the user to be redirected to the 1st choice.

### 2.7.6 Exiting the Program

A blue screen with red text

Description automatically generated

Figure 18



Figure 19

When the user chooses the choice `6` then the program exits by changing the variable `execute` which is the condition that is given to the while loop in turn looping until `execute` is False.

# Assumptions

The listed below are assumptions that were made about the solution when making the program:

1. Unique Project Code

It was assumed that the project code must be unique throughout all other projects with either ongoing or on-hold status, but the project code can be one which was already completed.

1. When removing a project, the current date should be stored.

It was assumed that the date on which the user removes a project needs to be stored as stated in the “DOC 333 CW Specification”, due to that reason the package “datetime” was imported to get the date on the 2nd option of removing a completed project. (“Remove a completed project from existing projects.”)

1. Only workers are assigned to ongoing projects.

It was assumed that only ongoing projects need to be assigned workers, the other statuses do not get assigned workers when they are created, but if they are updated to ongoing then workers are assigned to those projects.

1. The date format was assumed to be (MM/DD/YYYY)

It was assumed that the Date format that should be used is MM/DD/YYYY, and it is followed throughout the program.

1. There were 0 workers initially.

It is assumed that there are no workers initially when the program starts so it is required for the user to add workers before adding an `ongoing` project.

1. Worker Assignment

It is assumed that only ongoing projects need to be assigned workers and that on hold and completed projects do not require any assigned workers.

# Python Code

A computer screen shot of a blue screen

Description automatically generated

Figure 20

A screenshot of a computer

Description automatically generated

Figure 21

A screenshot of a computer

Description automatically generated

Figure 22

A screenshot of a computer program

Description automatically generated

Figure 23

A screenshot of a computer program

Description automatically generated

Figure 24

A screenshot of a computer program

Description automatically generated

Figure 25

A screenshot of a computer

Description automatically generatedFigure 26

A screenshot of a computer

Description automatically generated

Figure 27

A screenshot of a computer

Description automatically generated

Figure 28A screenshot of a computer program

Description automatically generated

Figure 29

# Test Cases

## 5.1 Main Menu

### 5.1.1 Test 1

The following test cases are the main menu, it checks for the first choice in the menu.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Choice | “1” | Displays Add a new project | Displays Add a new project | Pass |

A screenshot of a computer program

Description automatically generated

Figure 30

### 5.1.2 Test 2

The following test cases are the main menu, it checks for the second choice in the menu.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Choice | “2” | Displays remove a completed project | Displays remove a completed project | Pass |

A screenshot of a computer program

Description automatically generated

Figure 31

### 5.1.3 Test 3

The following test cases are the main menu, it checks for the third choice in the menu.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Choice | “3” | Displays add a new worker to the available worker's group | Displays add a new worker to the available worker's group | Pass |

A screenshot of a computer program

Description automatically generated

Figure 32

### 5.1.4 Test 4

The following test cases are the main menu, it checks for the fourth choice in the menu.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Choice | “4” | Displays updated details on ongoing projects | Displays updated details on ongoing projects | Pass |

A screenshot of a computer program

Description automatically generated

Figure 33

### 5.1.5 Test 5

The following test cases are the main menu, it checks for the fifth choice in the menu.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Choice | “5” | Displays Project Statistics | Displays Project Statistics | Pass |

A screenshot of a computer program

Description automatically generated

Figure 34

### 5.1.6 Test 6

The following test cases are the main menu, it checks for the sixth choice in the menu.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Choice | “6” | Exits the Program | Exits the Program | Pass |

A screen shot of a black screen

Description automatically generated

Figure 35

### 5.1.7 Test 7

The following test cases are the main menu, it checks what is the response given by the program when an unknown input is given.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Choice | “test” | “Please enter a valid choice..!” | “Please enter a valid choice..!” | Pass |

A screen shot of a black screen

Description automatically generated

Figure 36

## 5.2 Add a New Project

### 5.2.1 Test 1

The following test cases are for the 1st choice (“Add a new project to existing projects.”)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Project Code | “0” | Exit and Go to the Main Menu | Exit and Go to the Main Menu | Pass |



Figure 37

### 5.2.2 Test 2

The following test cases are for the 1st choice (“Add a new project to existing projects.”)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Project Code | “10” | Continue | Continue | Pass |
| Clients Name | “Ranuga” | Continue | Continue | Pass |
| Start Date | “2” | Error Raised, asking the user to enter the date again… | Error Raised, asking the user to enter the date again… | Pass |

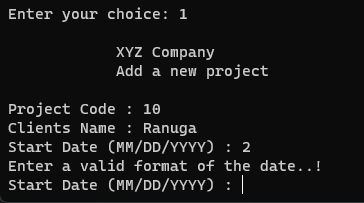


Figure 38

### 5.2.3 Test 3

The following test cases are for the 1st choice (“Add a new project to existing projects.”)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Project Code | “10” | Continue | Continue | Pass |
| Clients Name | “Ranuga” | Continue | Continue | Pass |
| Start Date | “12/12/2000” | Continue | Continue | Pass |
| Expected End Date | “01/01/2001” | Continue | Continue | Pass |
| Number of Workers | “!@” | “The msg entered was not an integer” | “The msg entered was not an integer” | Pass |

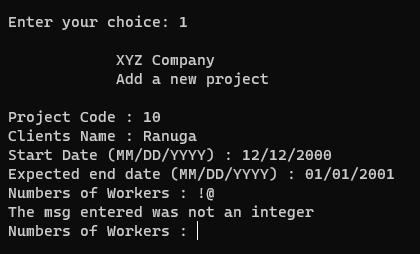


Figure 39

### 5.2.4 Test 4

The following test cases are for the 1st choice (“Add a new project to existing projects.”)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Project Code | “10” | Continue | Continue | Pass |
| Clients Name | “Ranuga” | Continue | Continue | Pass |
| Start Date | “12/12/2000” | Continue | Continue | Pass |
| Expected End Date | “01/01/2001” | Continue | Continue | Pass |
| Number of Workers | 10 | Continue | Continue | Pass |
| Project Status | “ongone” | “The entered project status is incorrect...” | “The entered project status is incorrect...” | Pass |

A computer screen with white text

Description automatically generated

Figure 40

### 5.2.5 Test 5

The following test cases are for the 1st choice (“Add a new project to existing projects.”)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Project Code | “10” | Continue | Continue | Pass |
| Clients Name | “Ranuga” | Continue | Continue | Pass |
| Start Date | “12/12/2000” | Continue | Continue | Pass |
| Expected End Date | “01/01/2001” | Continue | Continue | Pass |
| Number of Workers | 10 | Continue | Continue | Pass |
| Project Status | “on hold” | Continue | Continue | Pass |
| Save | “No” | “The project was \*not\* saved ..!” | “The project was \*not\* saved ..!” | Pass |

A screenshot of a computer program

Description automatically generated

Figure 41

### 5.2.6 Test 6

The following test cases are for the 1st choice (“Add a new project to existing projects.”)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Project Code | “10” | Continue | Continue | Pass |
| Clients Name | “Ranuga” | Continue | Continue | Pass |
| Start Date | “12/12/2000” | Continue | Continue | Pass |
| Expected End Date | “01/01/2001” | Continue | Continue | Pass |
| Number of Workers | 10 | Continue | Continue | Pass |
| Project Status | “on hold” | Continue | Continue | Pass |
| Save | “Yes” | “Successfully created a new project (True)” | “Successfully created a new project (True)” | Pass |

A computer screen with white text

Description automatically generated

Figure 42

## 5.3 Remove a Project

### 5.3.1 Assuming the project below has been added

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Project Code | “10” | Continue | Continue | Pass |
| Clients Name | “Ranuga” | Continue | Continue | Pass |
| Start Date | “12/12/2000” | Continue | Continue | Pass |
| Expected End Date | “01/01/2001” | Continue | Continue | Pass |
| Number of Workers | 10 | Continue | Continue | Pass |
| Project Status | “on hold” | Continue | Continue | Pass |
| Save | “Yes” | “Successfully created a new project (True)” | “Successfully created a new project (True)” | Pass |

### 5.3.2 Test 1

The following test cases are for the 2nd choice (“Remove Completed Project”)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Project Code | “90” | “The project does not exist..!” | “The project does not exist..!” | Pass |

A screen shot of a computer

Description automatically generated

Figure 43

### 5.3.3 Test 2

The following test cases are for the 2nd choice (“Remove Completed Project”)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Project Code | “10” | Continue | Continue | Pass |
| Save | “No” | “The project was not removed…!” | “The v was not removed…!” | Pass |

A screen shot of a computer program

Description automatically generated

Figure 44

### 5.3.4 Test 3

The following test cases are for the 2nd choice (“Remove Completed Project”)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Project Code | “10” | Continue | Continue | Pass |
| Save | “Yes” | “Successfully removed completed projects.” | “Successfully removed completed projects.” | Pass |

A screen shot of a black screen

Description automatically generated

Figure 45

## 5.4 Add new Workers

### 5.4.1 Test 1

The following test cases are for the 3rd choice (“Add new Workers”)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Number of Workers | -10 | “Workers must be more than 0..!” | “Workers must be more than 0..!” | Pass |

A black screen with white text

Description automatically generated

Figure 46

### 5.4.2 Test 2

The following test cases are for the 3rd choice (“Add new Workers”)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Number of Workers | “test” | “The msg entered was not an integer” | “The msg entered was not an integer” | Pass |

A screenshot of a computer screen

Description automatically generated

Figure 47

### 5.4.3 Test 3

The following test cases are for the 3rd choice (“Add new Workers”)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Number of Workers | 10 | Continue | Continue | Pass |
| Save | No | ” Workers were not added...!” | ” Workers were not added...!” | Pass |

A screenshot of a computer screen

Description automatically generated

Figure 48

### 5.4.4 Test 4

The following test cases are for the 3rd choice (“Add new Workers”)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Number of Workers | 10 | Continue | Continue | Pass |
| Save | Yes | “Workers added successfully...!” | “Workers added successfully...!” | Pass |

A screenshot of a computer screen

Description automatically generated

Figure 49

## 5.5 Update Project Details

### 5.5.1 Assuming the project below has been added

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Project Code | “10” | Continue | Continue | Pass |
| Clients Name | “Ranuga” | Continue | Continue | Pass |
| Start Date | “12/12/2000” | Continue | Continue | Pass |
| Expected End Date | “01/01/2001” | Continue | Continue | Pass |
| Number of Workers | 10 | Continue | Continue | Pass |
| Project Status | “on hold” | Continue | Continue | Pass |
| Save | “Yes” | “Successfully created a new project (True)” | “Successfully created a new project (True)” | Pass |

### 5.5.2 Test 1

The following test cases are for the 4th choice (“Update Project Details”)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Project Code | 0 | Exit and Go to the Main Menu | Exit and Go to the Main Menu | Pass |

A black screen with white text

Description automatically generated

Figure 50

### 5.5.3 Test 2

The following test cases are for the 4th choice (“Update Project Details”)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Project Code | “10” | Continue | Continue | Pass |
| Clients Name | “Devin” | Continue | Continue | Pass |
| Start Date | “0000” | “Enter a valid format of the date…!” | “Enter a valid format of the date…!” | Pass |

A screen shot of a computer

Description automatically generated

Figure 51

### 5.5.4 Test 3

The following test cases are for the 4th choice (“Update Project Details”)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Project Code | “10” | Continue | Continue | Pass |
| Clients Name | “Devin” | Continue | Continue | Pass |
| Start Date | “01/01/2020” | Continue | Continue | Pass |
| Expected End Date | “007” | “Enter a valid format of the date…!” | “Enter a valid format of the date…!” | Pass |

A screen shot of a computer

Description automatically generated

Figure 52

### 5.5.5 Test 4

The following test cases are for the 4th choice (“Update Project Details”)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Project Code | “10” | Continue | Continue | Pass |
| Clients Name | “Devin” | Continue | Continue | Pass |
| Start Date | “01/01/2020” | Continue | Continue | Pass |
| Expected End Date | “12/12/2022” | Continue | Continue | Pass |
| Number of Workers | “test” | “The msg entered was not an integer” | “The msg entered was not an integer” | Pass |

A screen shot of a computer

Description automatically generated

Figure 53

### 5.5.6 Test 5

The following test cases are for the 4th choice (“Update Project Details”)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Project Code | “10” | Continue | Continue | Pass |
| Clients Name | “Devin” | Continue | Continue | Pass |
| Start Date | “01/01/2020” | Continue | Continue | Pass |
| Expected End Date | “12/12/2022” | Continue | Continue | Pass |
| Number of Workers | 25 | Continue | Continue | Pass |
| Project Status | “on done” | “The entered project status is incorrect…” | “The entered project status is incorrect…” | Pass |

A computer screen with white text

Description automatically generated

Figure 54

### 5.5.7 Test 6

The following test cases are for the 4th choice (“Update Project Details”)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Project Code | “10” | Continue | Continue | Pass |
| Clients Name | “Devin” | Continue | Continue | Pass |
| Start Date | “01/01/2020” | Continue | Continue | Pass |
| Expected End Date | “12/12/2022” | Continue | Continue | Pass |
| Number of Workers | 25 | Continue | Continue | Pass |
| Project Status | “on hold” | Continue | Continue | Pass |
| Save | “no” | “The project was not updated” | “The project was not updated” | Pass |

A computer screen with white text

Description automatically generated

Figure 55

### 5.5.8 Test 7

The following test cases are for the 4th choice (“Update Project Details”)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Project Code | “10” | Continue | Continue | Pass |
| Clients Name | “Devin” | Continue | Continue | Pass |
| Start Date | “01/01/2020” | Continue | Continue | Pass |
| Expected End Date | “12/12/2022” | Continue | Continue | Pass |
| Number of Workers | 25 | Continue | Continue | Pass |
| Project Status | “on hold” | Continue | Continue | Pass |
| Save | “yes” | “Project details updated successfully (True)” | “Project details updated successfully (True)” | Pass |

A screen shot of a computer

Description automatically generated

Figure 56

### 5.5.9 Test 8

The following test cases are for the 4th choice (“Update Project Details”)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Project Code | “58” | “There isn’t a project with the mentioned project code…!” | “There isn’t a project with the mentioned project code…!” | Pass |

A black screen with white text

Description automatically generated

Figure 57

## 5.6 Project Statistics

### 5.6.1 Assuming the project below has been added from Choice 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Number of Workers | 100 | Continue | Continue | Pass |
| Save | “Yes” | “Workers added successfully” | “Workers added successfully” | Pass |

### 5.6.2 Assuming the project below has been added from Choice 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Project Code | “10” | Continue | Continue | Pass |
| Clients Name | “Ranuga” | Continue | Continue | Pass |
| Start Date | “12/12/2000” | Continue | Continue | Pass |
| Expected End Date | “01/01/2001” | Continue | Continue | Pass |
| Number of Workers | 10 | Continue | Continue | Pass |
| Project Status | “on going” | Continue | Continue | Pass |
| Save | “Yes” | “Successfully created a new project (True)” | “Successfully created a new project (True)” | Pass |

### 5.6.3 Test 1

The following test cases are for the 5th choice (“Project Statistics”)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Add Project | “No” | Exit and Go to the Main Menu | Exit and Go to the Main Menu | Pass |

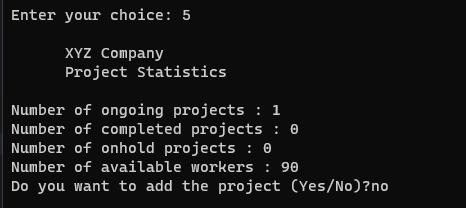


Figure 58

### 5.6.4 Test 2

The following test cases are for the 5th choice (“Project Statistics”)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Add Project | “Yes” | Redirect to Choice 1 | Redirect to Choice 1 | Pass |

A black screen with white text

Description automatically generated

Figure 59

## 5.7 Exit

### 5.7.1 Test 1

The following test cases are for the 6th choice (“Exit”)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Input Entered | Expected Outcome | Actual Outcome | Results |
| Choice | 6 | “Exiting Program…” | “Exiting Program…” | Pass |

A screen shot of a black screen

Description automatically generated

Figure 60

# 6. Conclusion

In conclusion, the solution that was implemented can manage XYZ Corporation's Projects, by adding new projects, updating details of ongoing projects and removing completed projects. With the ability to add new workers to the company as well as see statistics of how all the projects are going including the workers available. The code has been tested thoroughly and has many verification steps and in turn, it should meet up to the standards that XYZ Corporation needs to continue their operations seamlessly and smoothly.