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

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

List of Figures

Introduction

THE PROBLEM

The company “XYZ” a company that 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.

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.

PESUDOCODE

1. Start
2. # importing packages
   - IMPORT datetime
3. # 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
4. 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

5. 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)
6. 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)

7. 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)
8. define function date_vertification(msg)
    - 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
9. 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))

10. 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
11. 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)
12. 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 save 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!”

VERIFICATION FUNCTIONS

date_vertification()

```
def date_verification(msg: str) -> str:
    """A function that uses recursion to make sure that the entered date is in a correct format...
    Keyword arguments:
    msg (str) -- the message that should be displayed...
    Return: A string which contains a correct date format...
    """
    date = input(msg)
    splitted_date = date.split(date[2] if len(date) > 3 else " ")
    if len(splitted_date) != 3:
        print("Enter a valid format of the date..!")
        return date_verification(msg)
    month, date, _ = splitted_date
    if int(month) > 12:
        print("Enter a valid month..!")
        return date_verification(msg)
    if int(date) > 31:
        print("Enter a valid date..!")
        return date_verification(msg)
    return date
```

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.

project_status_vertification()

```
def project_status_verification(
    msg: str = "Project Status (ongoing/completed/on hold) : ",
    update_status: bool = False,
) -> (str, list, int):
    """An function that uses recursion to make sure that an input is enter as required
    Keyword arguments:
    msg -- The message that should be displayed to the user to get the project status input
    update_status -- Whether to update the status count
    Return: Tuple[The state enter by the user,
                  the statistic list used to track the project status count,
                  the index of the enter state
                ]
    """
    project_state = str(input(msg)).replace(" ", "").lower()
    if project_state not in possible_inputs:
        print("The entered project status is incorrect...")
        return project_status_verification()
    if update_status:
        statistics_list[possible_inputs.index(project_state)] += 1
    return (
        project_state,
        statistics_list,
        possible_inputs.index(project_state),
    )
```

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]

project_code_vertification()

```
def project_code_verification(msg: str, project_codes: list) -> str:
    """Project Code Verification function with the use of recursion...
    Keyword arguments:
    msg (str) -- The message that is displayed and ask the user to enter the project code
    project_codes (list) -- The list of project codes that already exists
    Return: (str) of a project code that doesnt already exist...
    """
    project_code = str(input(msg))
    if project_code in project_codes:
        print("Project Code already exists..!")
        return project_code_verification(msg, project_codes)
    return project_code

def check_if_int(msg) -> int:
    try:
        return int(input(msg))
    except:
        print("The msg entered was not an integer")
        return check_if_int(msg)
```

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.

check_if_int()

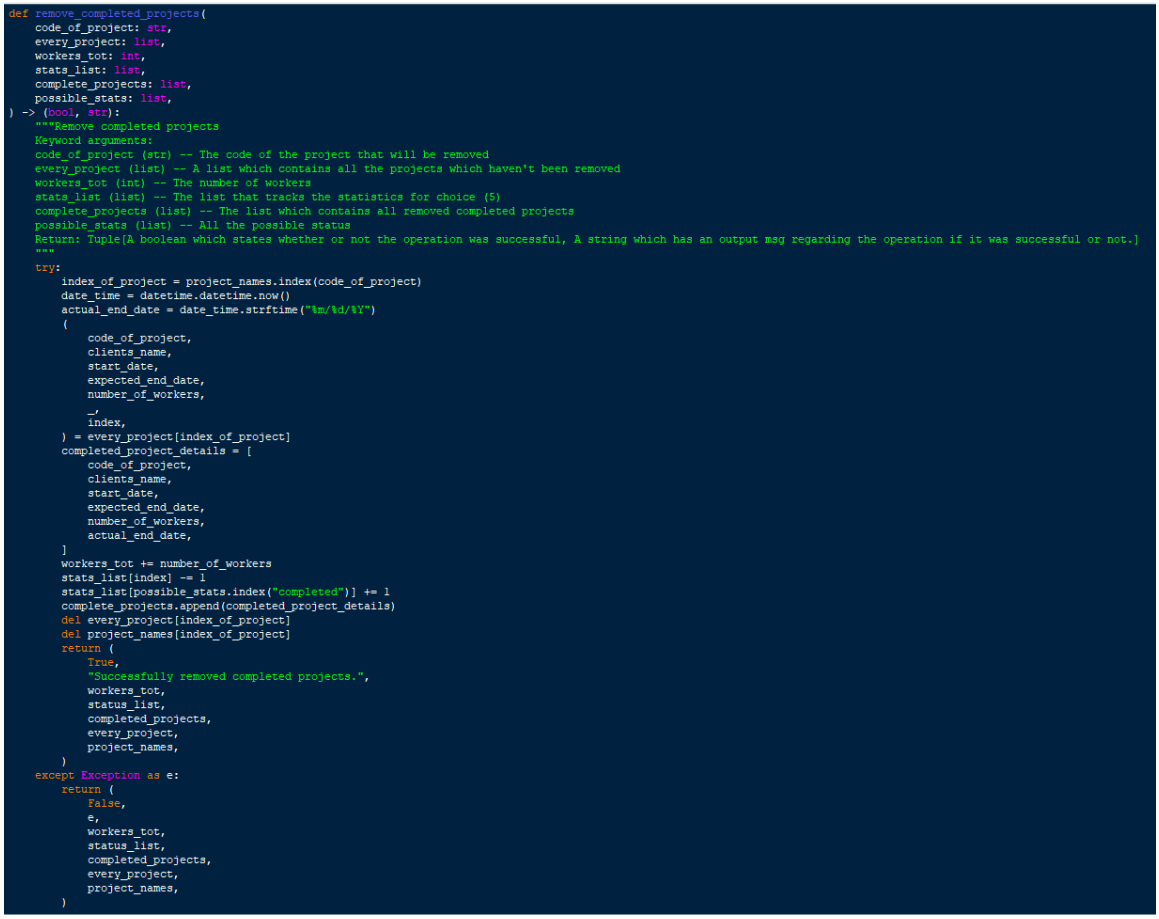
```
def check_if_int(msg) -> int:
    try:
        return int(input(msg))
    except:
        print("The msg entered was not an integer")
        return check_if_int(msg)
```

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).

MAIN FUNCTIONS

remove_completed_projects ()



```python
def remove_completed_projects(
    code_of_project: str,
    every_project: list,
    workers_tot: int,
    stats_list: list,
    complete_projects: list,
    possible_stats: list,
) -> (bool, str):
    """Remove completed projects
    Keyword arguments:
    code_of_project (str) -- The code of the project that will be removed
    every_project (list) -- A list which contains all the projects which haven't been removed
    workers_tot (int) -- The number of workers
    stats_list (list) -- The list that tracks the statistics for choice (5)
    complete_projects (list) -- The list which contains all removed completed projects
    possible_stats (list) -- All the possible status
    Return: Tuple[A boolean which states whether or not the operation was successful, A string which has an output msg regarding the operation if it was successful or not.]
    """
    try:
        index_of_project = project_names.index(code_of_project)
        date_time = datetime.datetime.now()
        actual_end_date = date_time.strftime("%m/%d/%Y")
        (
            code_of_project,
            clients_name,
            start_date,
            expected_end_date,
            number_of_workers,
            _,
            index,
        ) = every_project[index_of_project]
        completed_project_details = [
            code_of_project,
            clients_name,
            start_date,
            expected_end_date,
            number_of_workers,
            actual_end_date,
        ]
        workers_tot += number_of_workers
        stats_list[index] -= 1
        stats_list[possible_stats.index("completed")] += 1
        complete_projects.append(completed_project_details)
        del every_project[index_of_project]
        del project_names[index_of_project]
        return (
            True,
            "Successfully 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,
        )
```

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)

create_project()

```
def create_project(
    status_list: list,
    index: int,
    code_of_project: str,
    clients_name: str,
    start_date: str,
    expected_end_date: str,
    number_of_workers: str,
    project_status: str,
    workers_tot: int,
    project_names:list,
    all_projects:list
) -> (bool, str):
    """This function creates a new project
    Keyword arguments:
    status_list (list) -- The list that is used for project statistics
    index (index) -- The index of the project status in the status list
    code_of_project (str) -- The code of the project
    clients_name (str) -- The project's clients name
    start_date (str) -- The start date of the project
    expected_end_date (str) -- The expected end date of the project
    number_of_workers (str) -- The number of workers required for the project
    project_status (str) -- The status of the project out of (ongoing,on hold, completed)
    workers_tot (int) -- total number of workers in the organization
    project_names (list) -- a list that contains all the project codes
    all_projects (list) -- a list that contains all the projects
    Return: Tuple[
        A boolean which shows if the function successfully executed or not,
        The message which will be displayed to the user
    ]
    """
    try:
        status_list[index] += 1
        project_data = [
            code_of_project,
            clients_name,
            start_date,
            expected_end_date,
            number_of_workers,
            project_status,
            index,
        ]
        if project_status == "ongoing" and (number_of_workers > workers_tot):
            return (False,"There is not enough workers",workers_tot)
        if project_status == "ongoing":
            workers_tot -= number_of_workers
        project_names.append(code_of_project)
        all_projects.append(project_data)
        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)
```

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, start date of the project, expected end date of the project, status of the project from ("ongoing"," on hold"," completed"), 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)

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.

```python
def update_project_details(
    status_list: list,
    index: int,
    previous_index: int,
    code_of_project: str,
    clients_name: str,
    start_date: str,
    expected_end_date: str,
    number_of_workers: str,
    project_status: str,
    current_workers: int,
    workers_tot: int,
    previous_project_status: str,
) -> (bool, str):
    """An function that updates the project details
    Keyword arguments:
    status_list (list) -- The list that is used for project statistics
    index (int) -- The index of the new project status in the status_list
    previous_index (int) -- The index of the previous project status in the status_list
    code_of_project (str) -- The project code of the project
    clients_name (str) -- The (updated / usual) client name
    start_date (str) -- The (updated / usual) start date
    expected_end_date (str) -- The (updated / usual) end date
    number_of_workers (str) -- The (updated / usual) number of workers
    project_status (str) -- The (updated / usual) project status
    current_workers (int) -- The number of workers before the project was updated
    workers_tot (int) -- The total number of workers
    previous_project_status (str) -- The previous project status
    Return: Tuple[
        A boolean which shows if the function successfully executed or not,
        The message which will be displayed to the user
    ]
    """
    try:
        if project_status == "ongoing":
            if number_of_workers > workers_tot + (
                current_workers if previous_project_status == "ongoing" else 0
            ):
                return (False, "Workers chosen are too much", workers_tot)
        if project_status == "ongoing":
            workers_tot -= number_of_workers
        if previous_project_status == "ongoing":
            workers_tot += current_workers
        status_list[index] += 1
        status_list[previous_index] -= 1
        project_data = [
            code_of_project,
            clients_name,
            start_date,
            expected_end_date,
            number_of_workers,
            project_status,
            index,
        ]
        index = project_names.index(code_of_project)
        all_projects[index] = project_data
        return (True, "Project details updated successfully", workers_tot)
    except Exception as e:
        return (False, e, workers_tot)
```

Figure 8

HELPER FUNCTIONS

main()

```
def menu(
    redirect: bool = False,
    to: int = None,
    name_of_the_company: str = company_name,
    msg: str = "Enter your choice: ",
) -> str:
    """This function finds the choice that should be displayed to the user next...

    Keyword arguments:
    redirect (bool) -- A boolean to know whether or not the user will be redirected
    to (int) -- The choice that user will be redirected to
    name_of_the_company (str) -- The companies name that will be displayed
    msg (str) -- The message that will be displayed to the user asking their next choice

    Return: (str) the next choice which the user has chosen...
    """
    main_menu = f"""
     {name_of_the_company}
     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. Updates details on ongoing projects.
     5. Project Statistics.
     6. Exit
    """
    print("Redirecting..." if redirect else main_menu)
    return to if redirect else str(input(msg))
```

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.

EXPLANATION

Add a new project to existing projects.

```
            XYZ Company
            Add a new project

Project Code : 10
Clients Name : Ranuga
Start Date (MM/DD/YYYY) : 12/21/2008
Expected end date (MM/DD/YYYY) : 12/21/2012
Numbers of Workers : 10
Project Status (ongoing/completed/on hold) : onhold
Do you want to save the project(Yes/No)? yes
Successfully created a new project (True)
```

Figure 10

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”.

Removing a Completed project from existing projects

```
Enter your choice: 2

      XYZ Company
      Remove Completed Project

Project Code : 10
Do you want to save the project (Yes/ No)? yes
Successfully removed completed projects. (True)
```

Figure 11

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`.

Adding new workers to the available worker's group

```
Enter your choice: 3

      XYZ Company
      Add new Workers

Number Workers to Add : 100
Do you want to add ? (Yes / No) yes
Workers added successfully..!
```

Figure 12

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.

Updating Details to an ongoing project

```
Enter your choice: 4

      XYZ Company
      Update Project Details

Project Code : 10
Clients Name : Devin
Start Date (MM/DD/YYYY) : 06/16/2017
Expected end date (MM/DD/YYYY) : 06/16/2020
Numbers of Workers : 25
Project Status (ongoing/completed/on hold) : ongoing
Do you want to update the project details (Yes/No)?yes
Project details updated successfully (True)
```

Figure 13

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`

Project Statistics

```
Enter your choice: 5

      XYZ Company
      Project Statistics

Number of ongoing projects : 1
Number of completed projects : 0
Number of onhold projects : 0
Number of available workers : 75
Do you want to add the project (Yes/No)?yes
Redirecting...

            XYZ Company
            Add a new project

Project Code : 0
```

Figure 14

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.

Exiting the Program

```
elif choice == "6":
    print('Exiting Program...')
    execute = False
```

```
while execute:
    choice = menu(
```

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.
2. 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.")
3. 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.
4. 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
5. 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.
6. 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

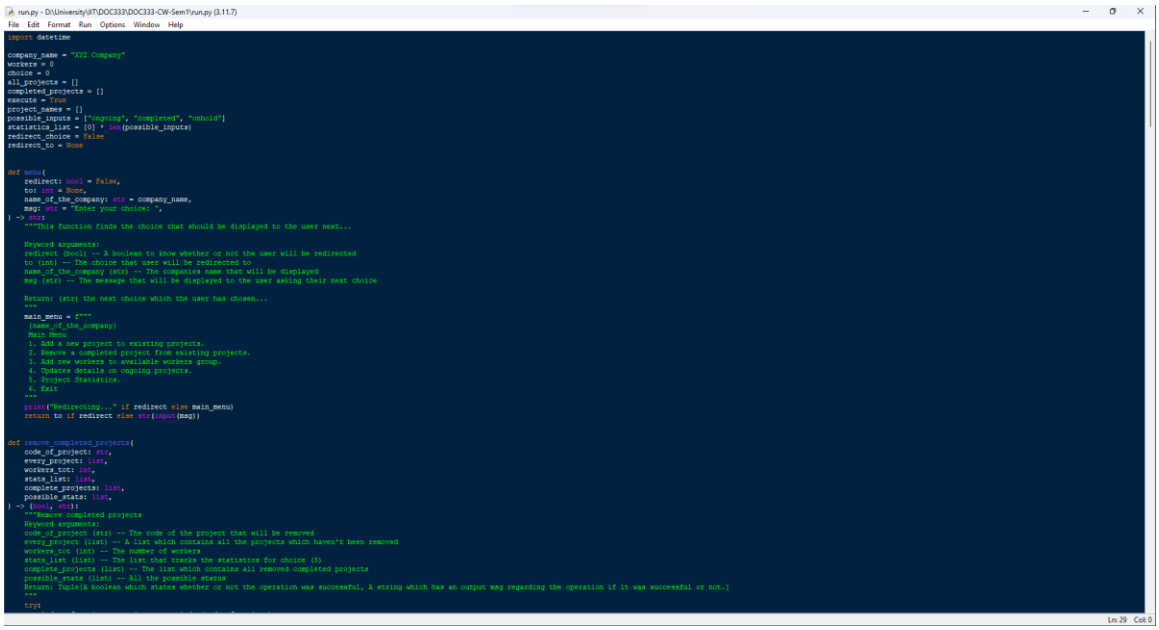


Figure 15

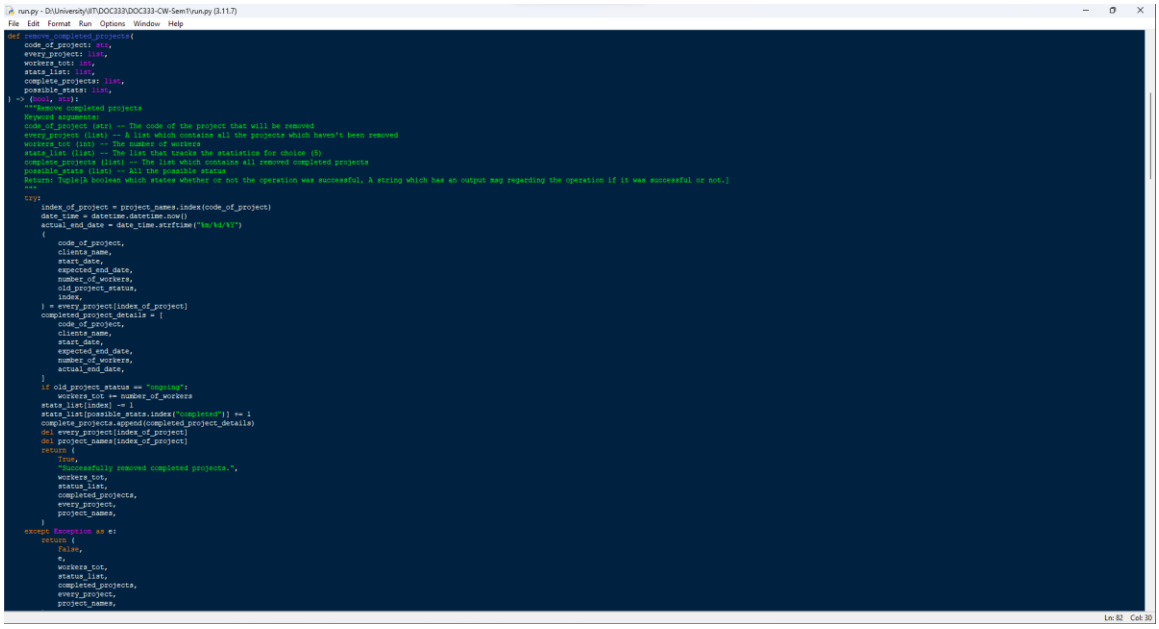


Figure 16

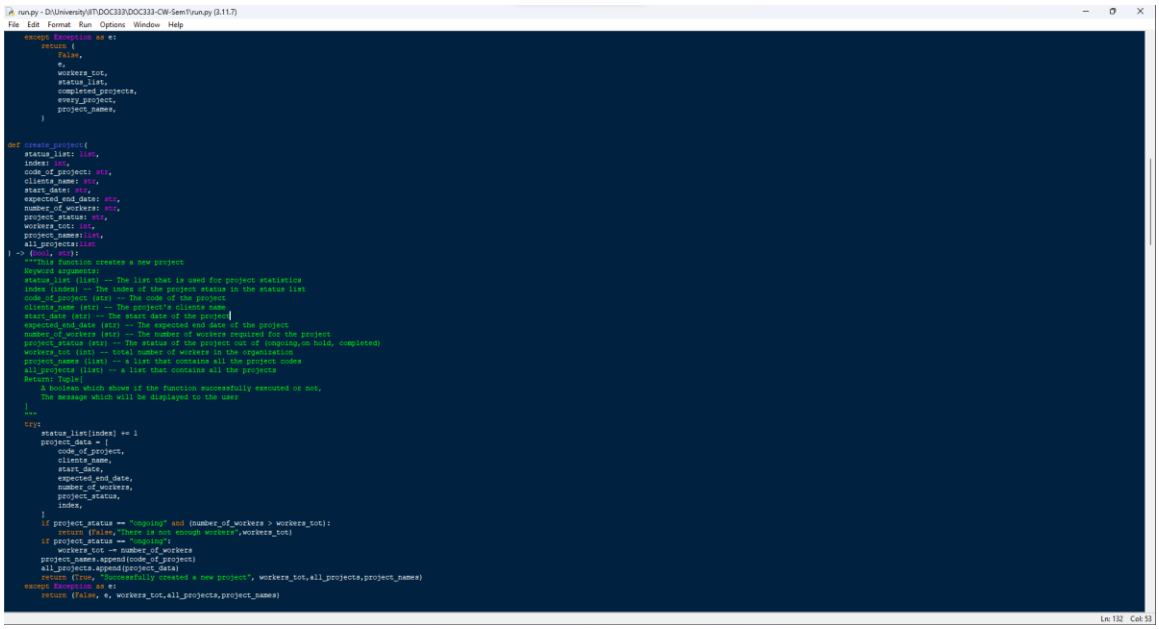


Figure 17

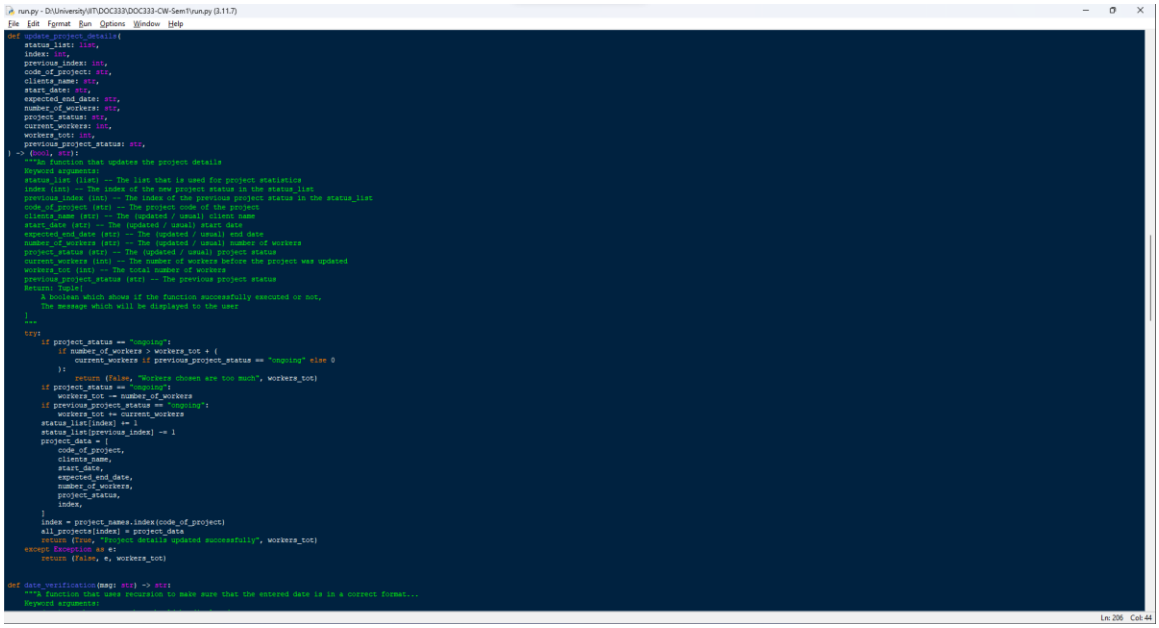


Figure 18

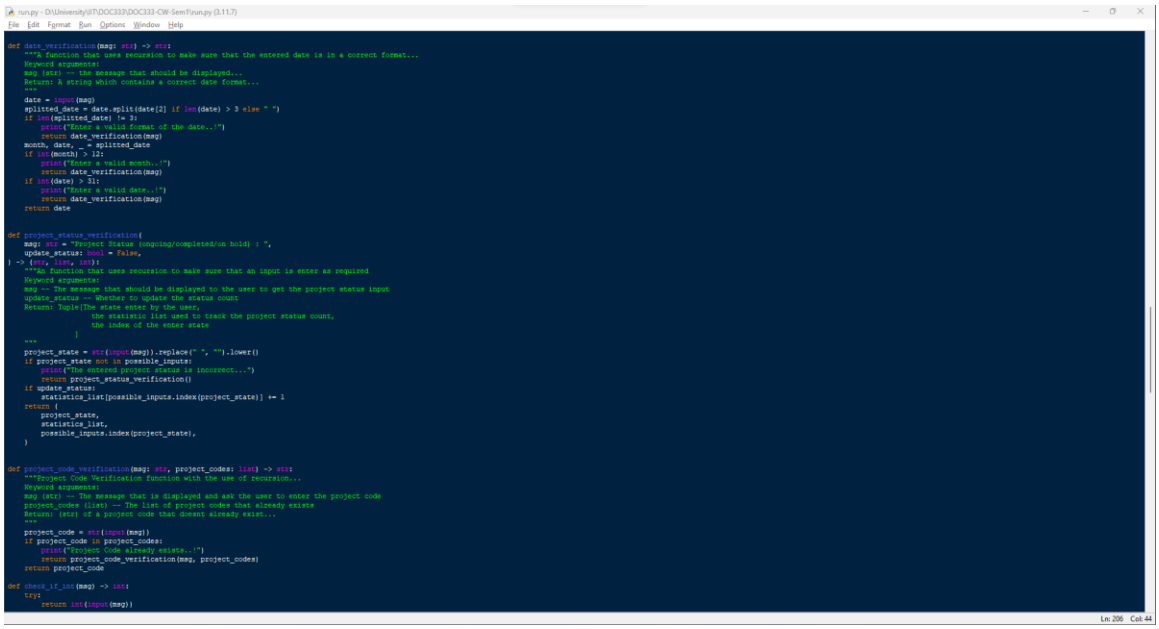


Figure 19

Figure 20

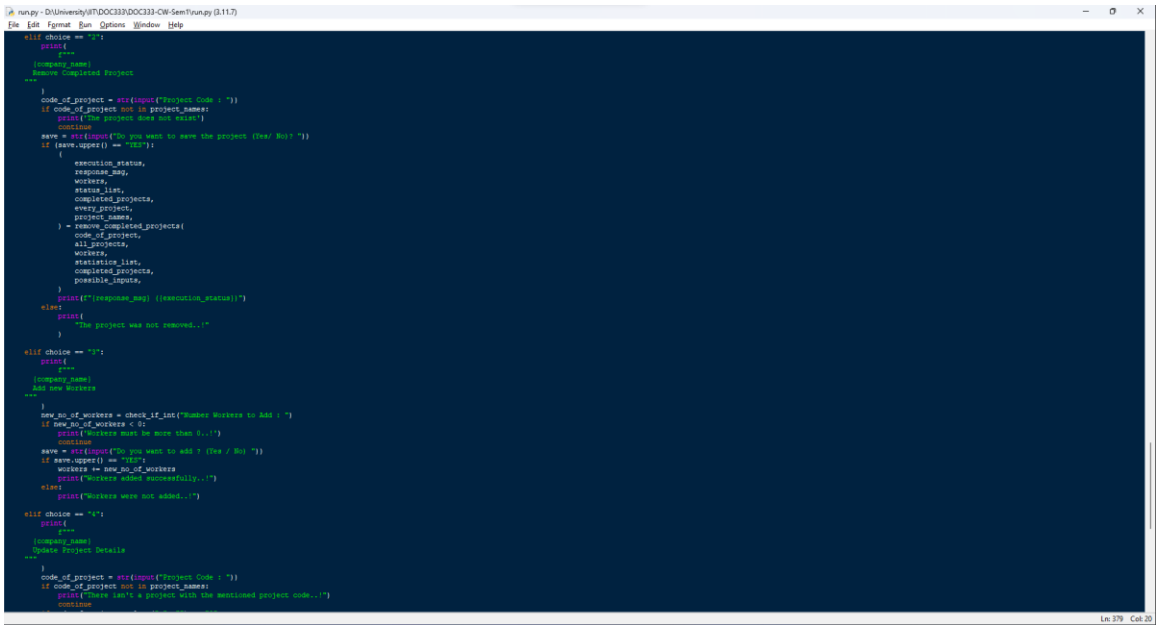


Figure 21

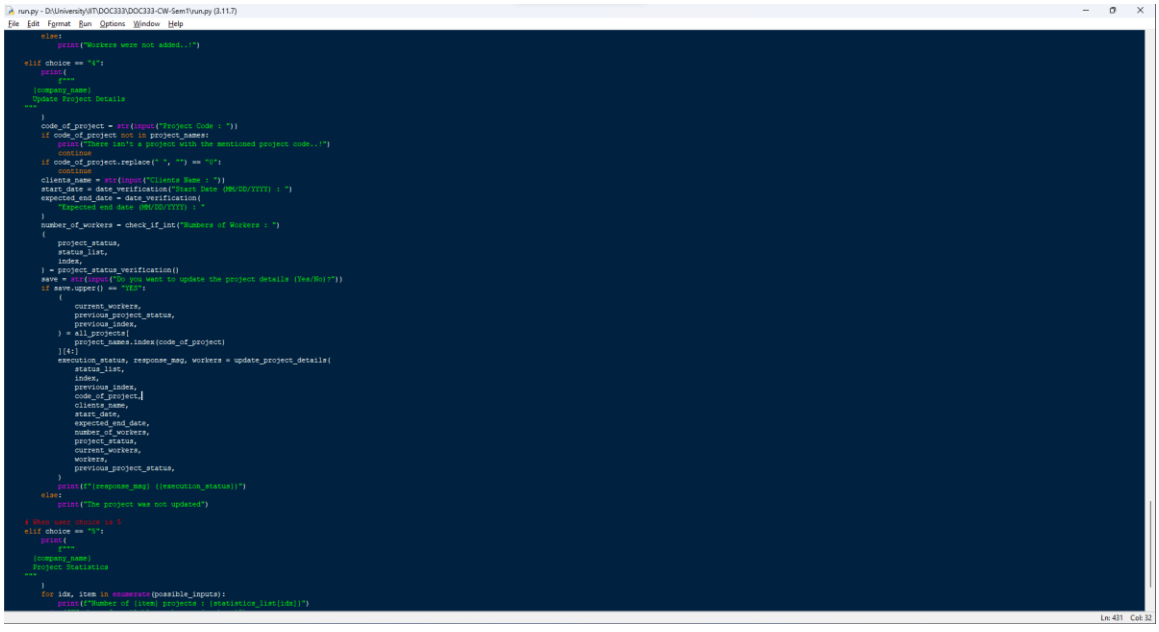


Figure 22

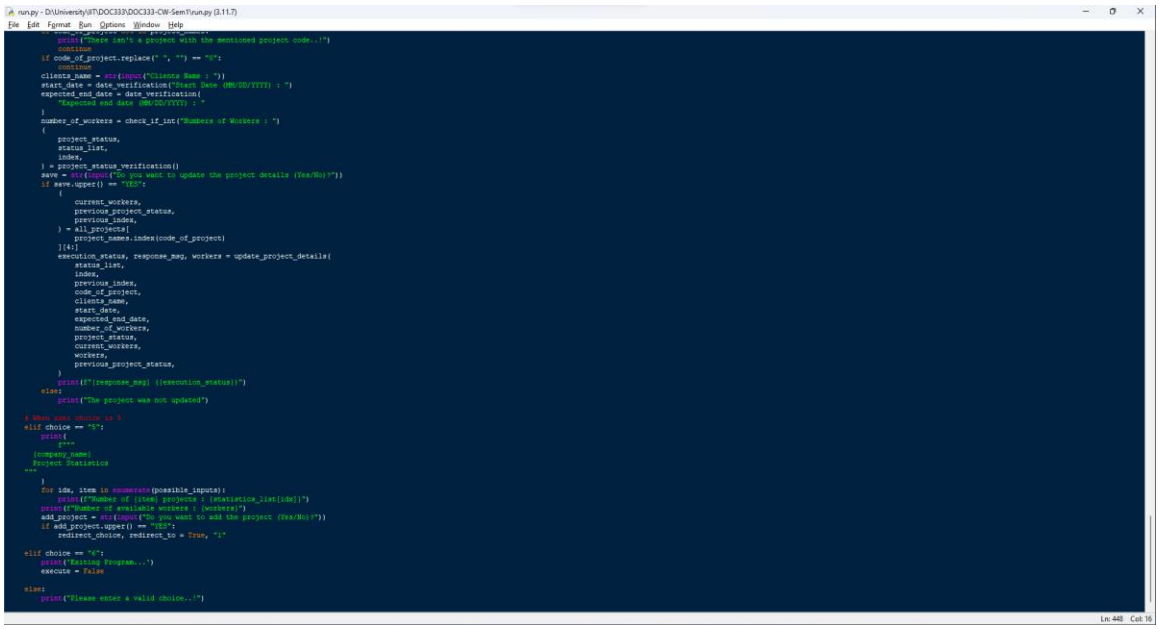


Figure 23

Test Cases

MAIN MENU

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

```
     XYZ Company
     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. Updates details on ongoing projects.
     5. Project Statistics.
     6. Exit

Enter your choice: 1

            XYZ Company
            Add a new project

Project Code :
```

Figure 24

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

```
     XYZ Company
     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. Updates details on ongoing projects.
     5. Project Statistics.
     6. Exit

Enter your choice: 2

      XYZ Company
      Remove Completed Project

Project Code :
```

Figure 25

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

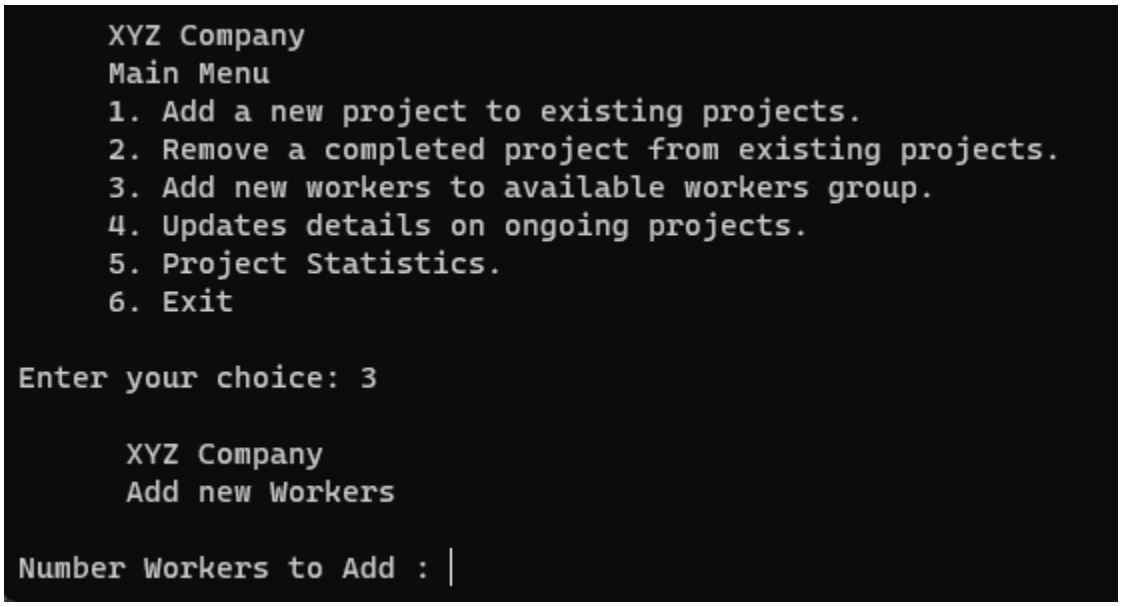


Figure 26

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

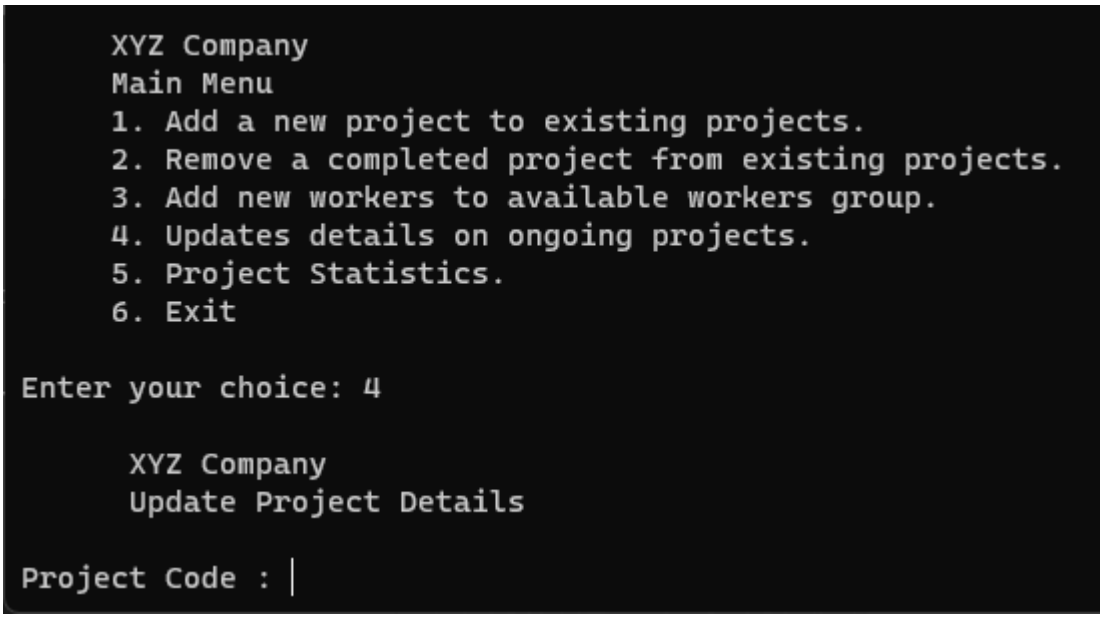


Figure 27

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

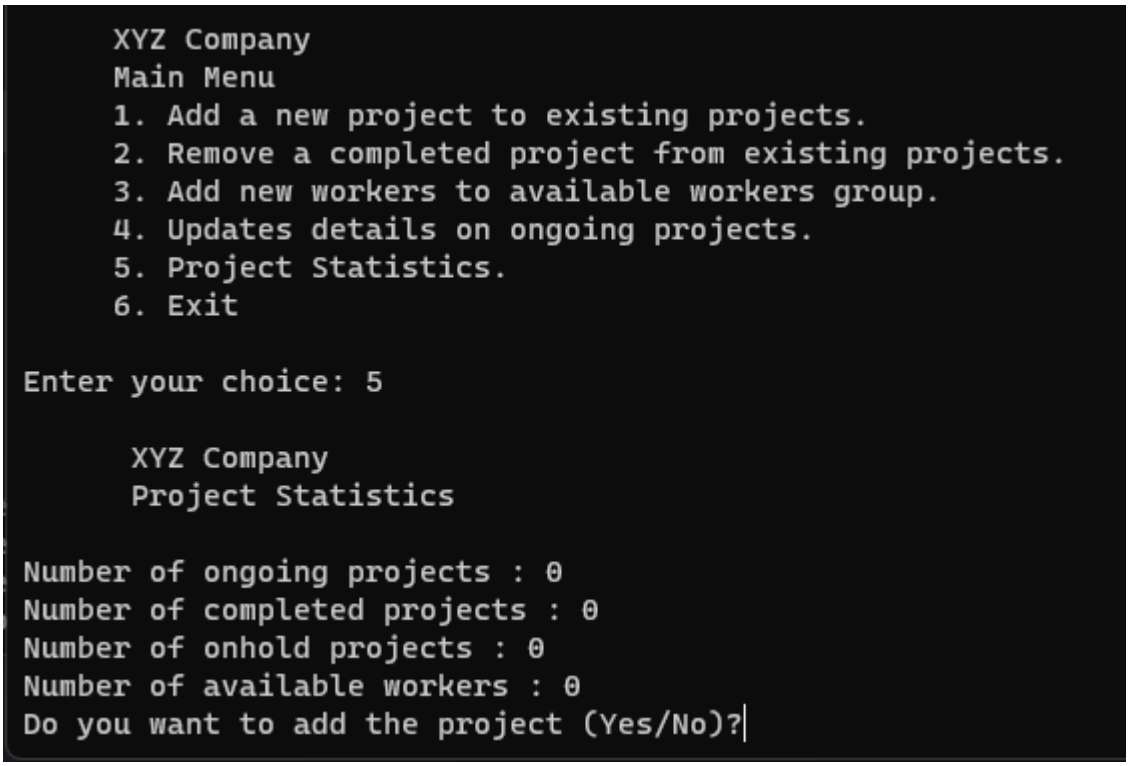


Figure 28

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

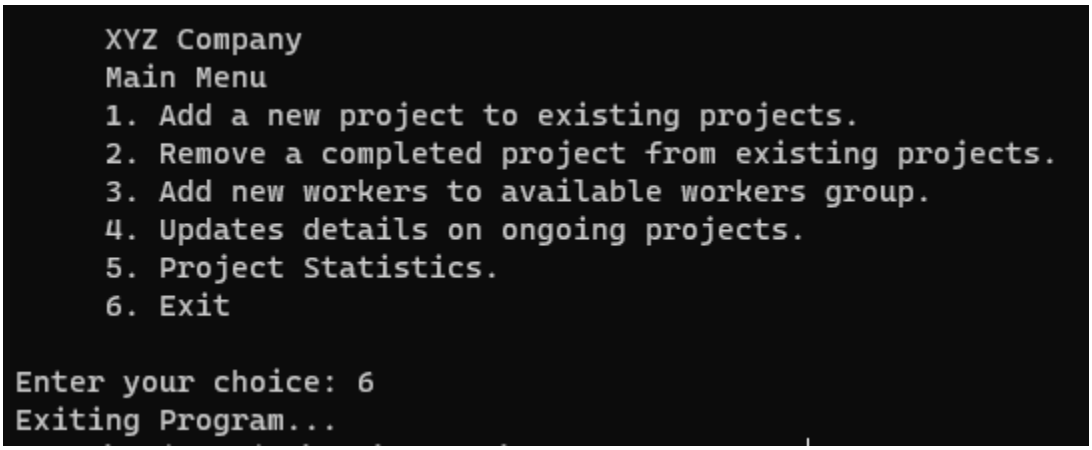


Figure 29

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

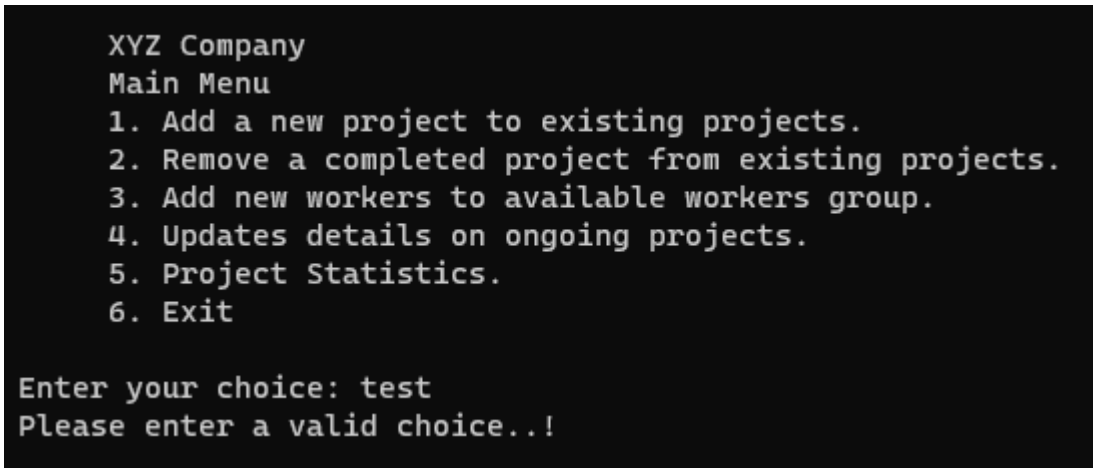


Figure 30

ADD A NEW PROJECT

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 31

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

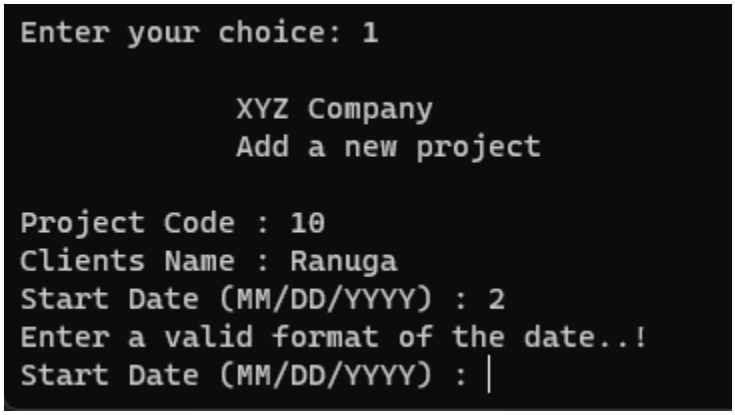


Figure 32

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

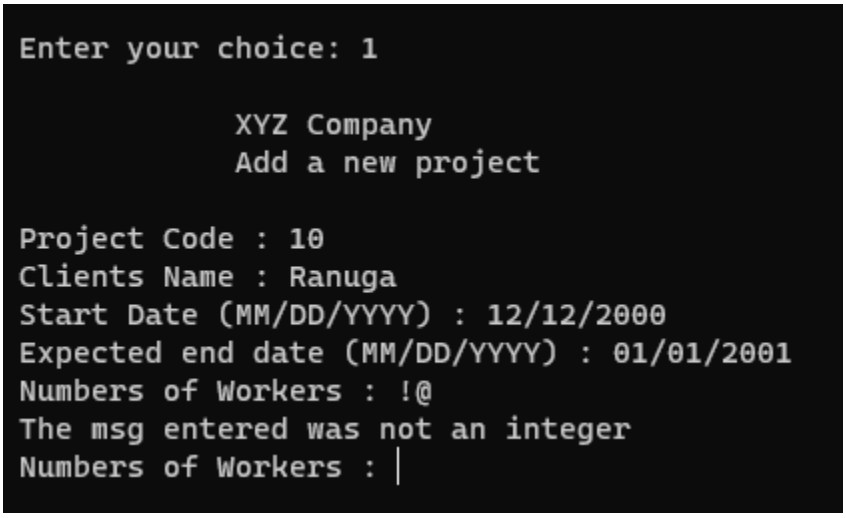


Figure 33

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

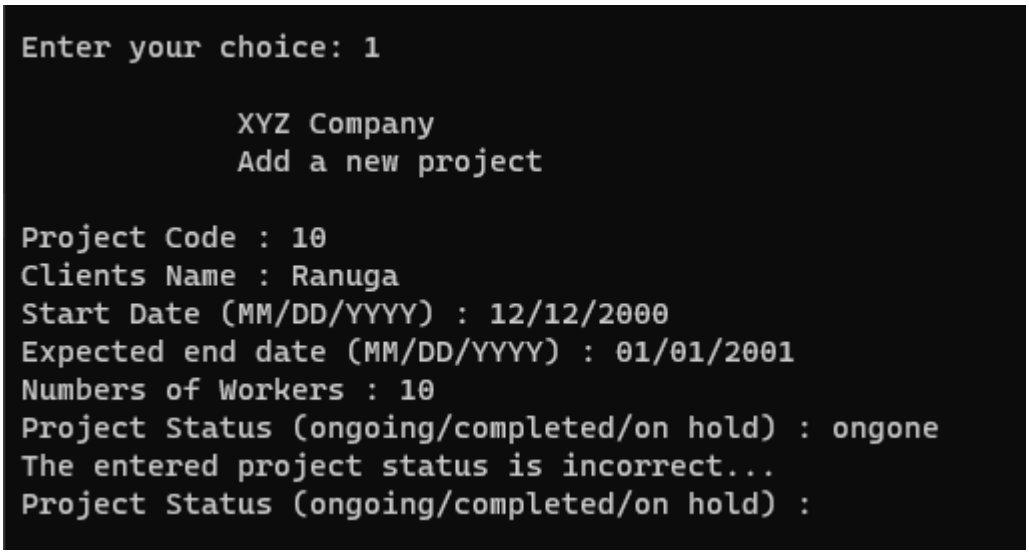


Figure 34

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

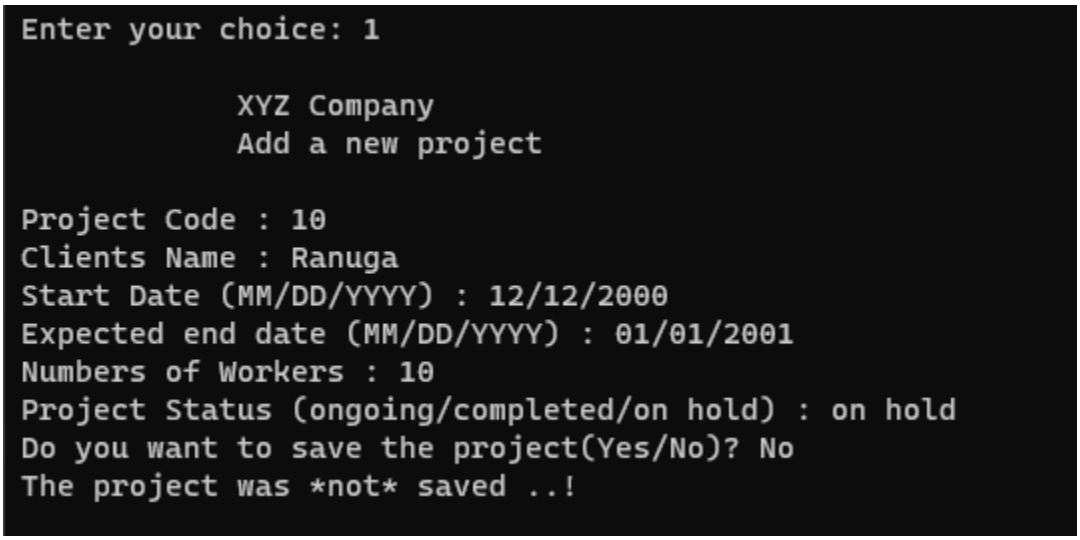


Figure 35

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

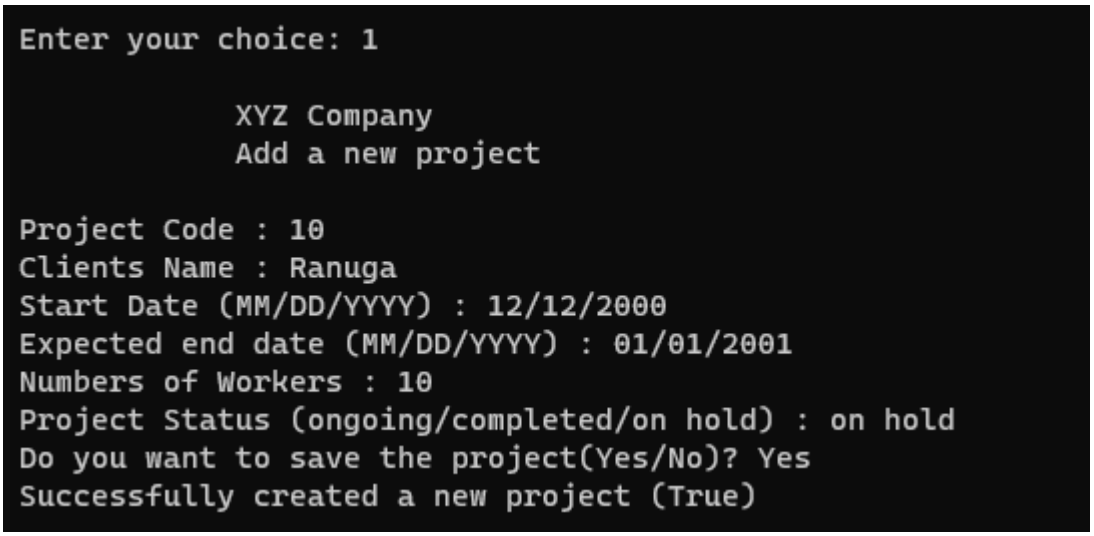


Figure 36

REMOVE A PROJECT

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

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

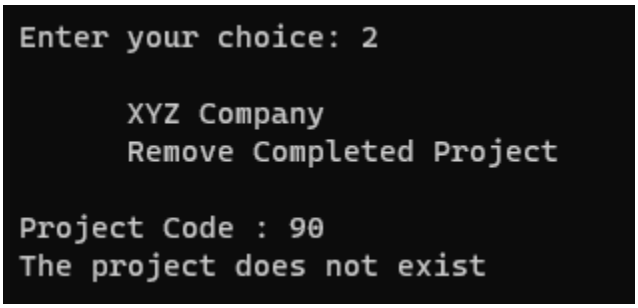


Figure 37

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

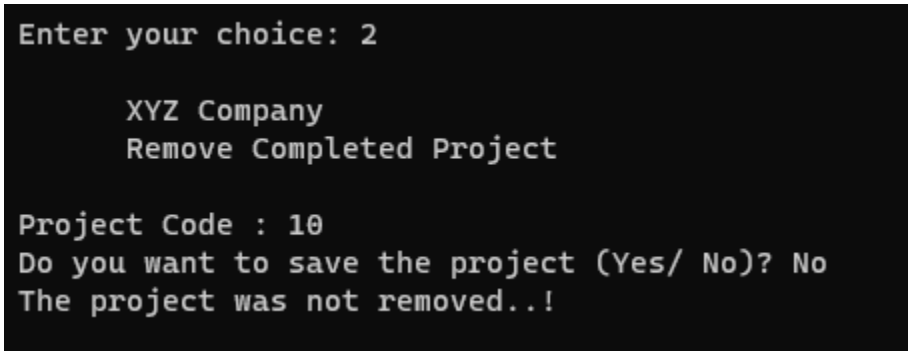


Figure 38

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

```
Enter your choice: 2

      XYZ Company
      Remove Completed Project

Project Code : 10
Do you want to save the project (Yes/ No)? yes
Successfully removed completed projects. (True)
```

Figure 39

ADD NEW WORKERS

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

```
Enter your choice: 3

      XYZ Company
      Add new Workers

Number Workers to Add : -10
Workers must be more than 0..!
```

Figure 40

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

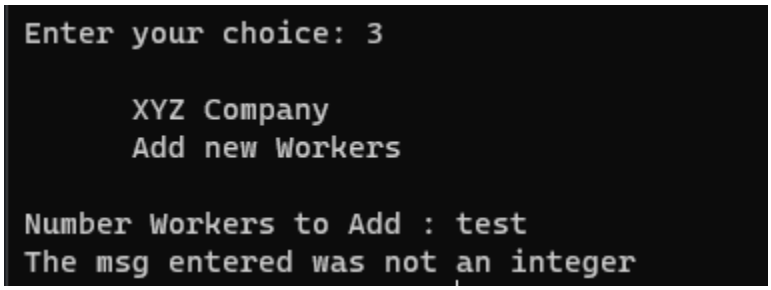


Figure 41

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

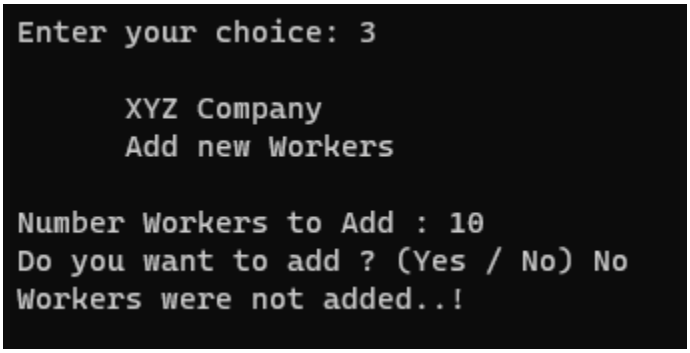


Figure 42

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

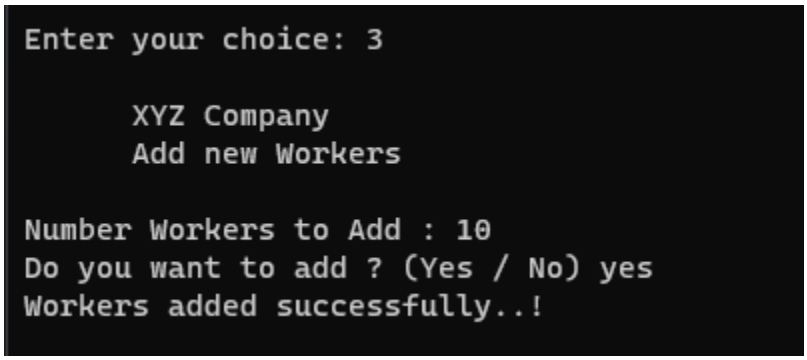


Figure 43

UPDATE PROJECT DETAILS

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

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

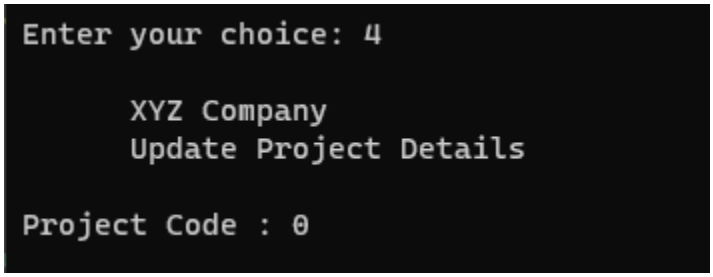


Figure 44

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

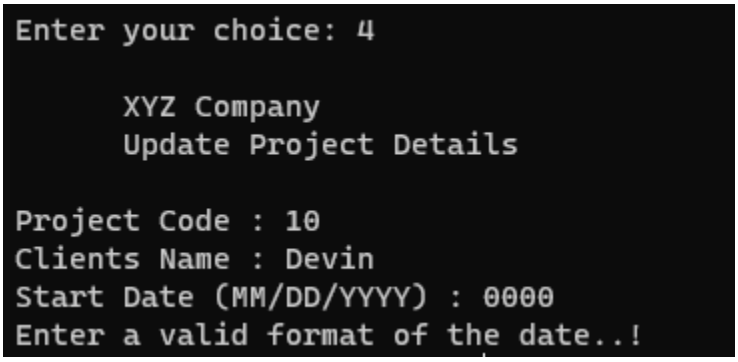


Figure 45

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

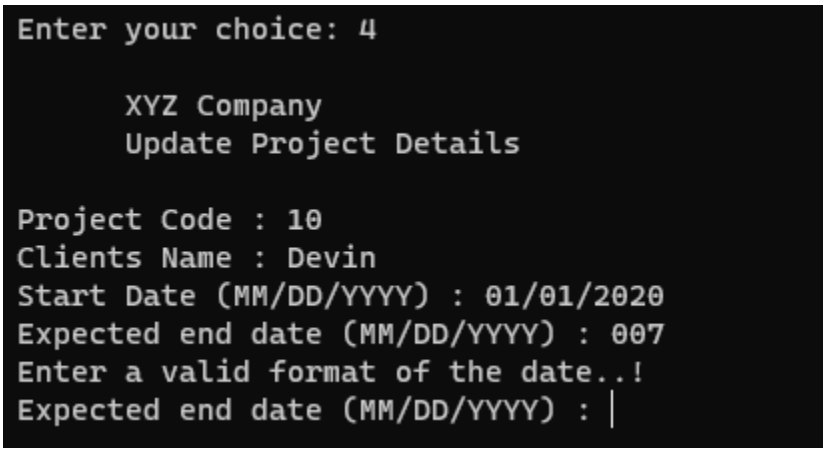


Figure 46

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

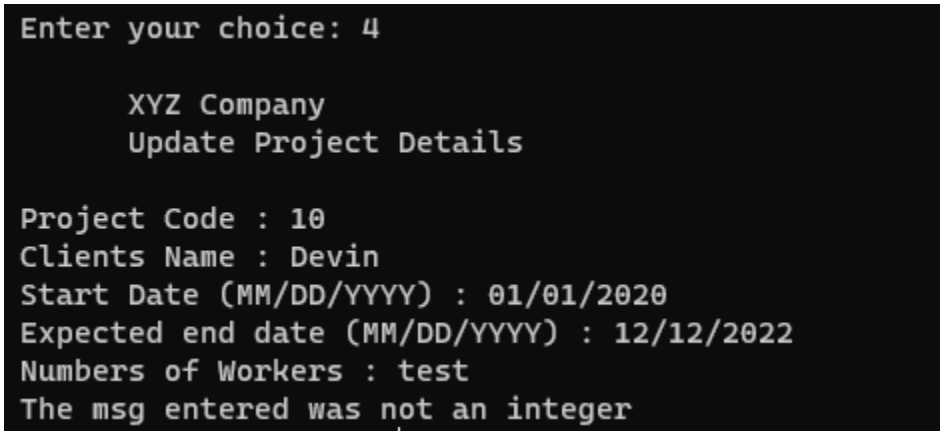


Figure 47

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

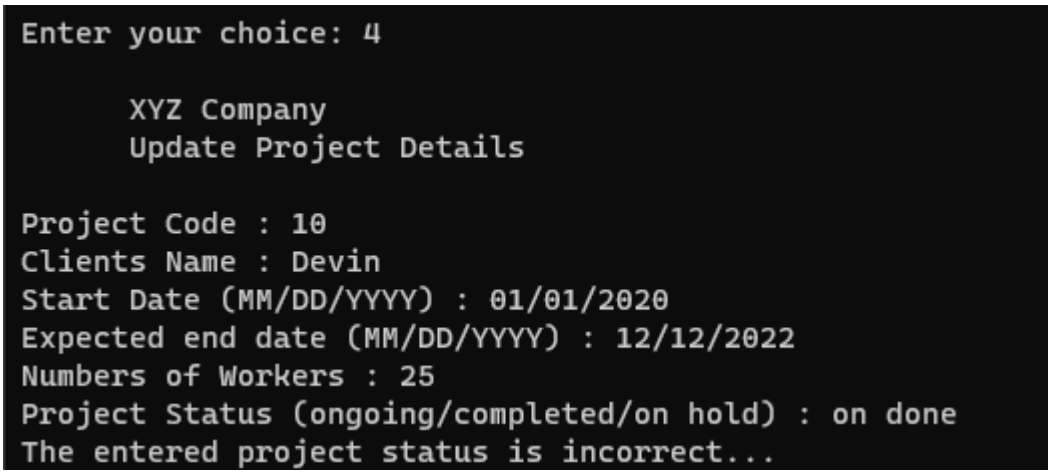


Figure 48

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

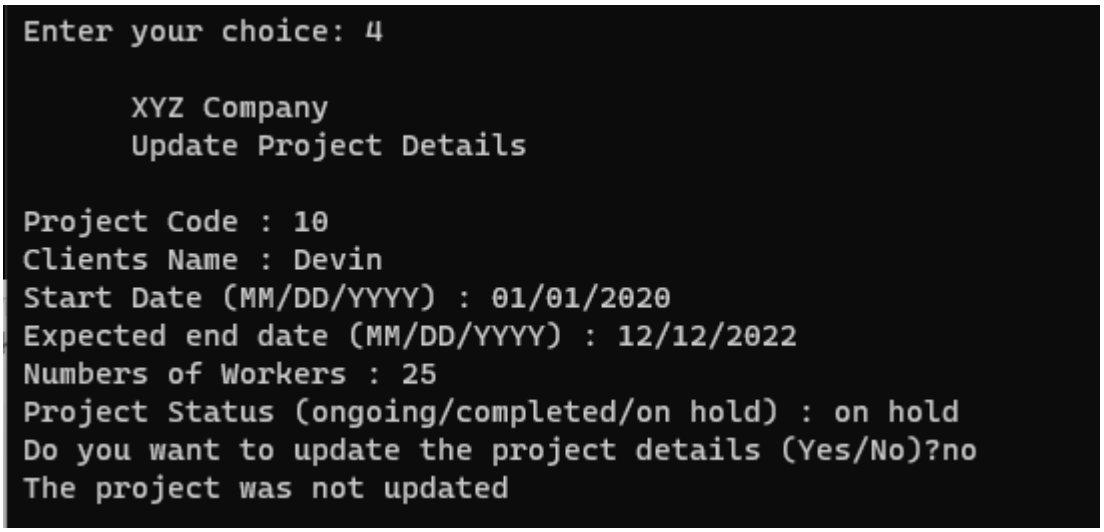


Figure 49

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

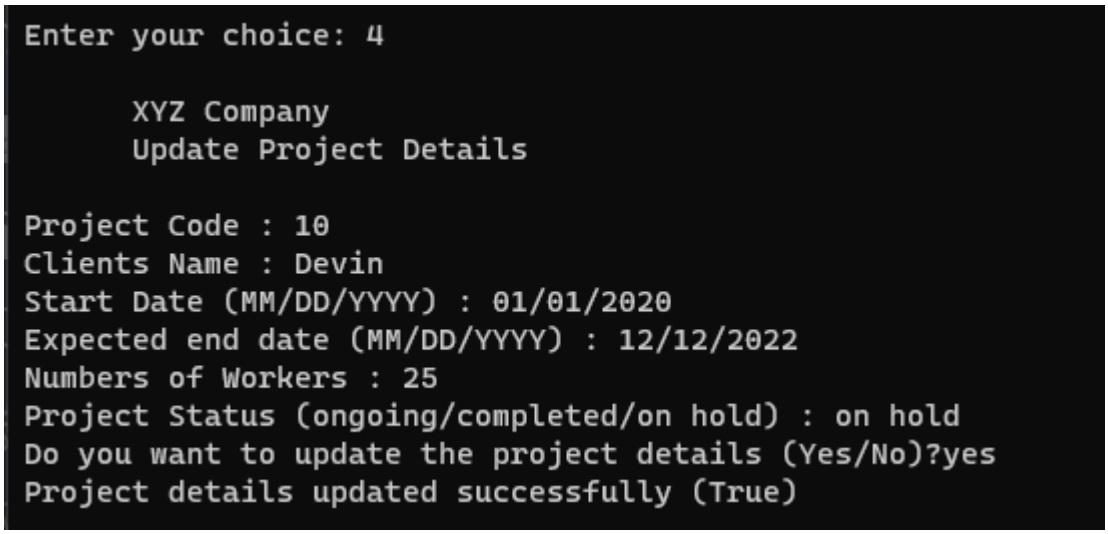


Figure 50

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

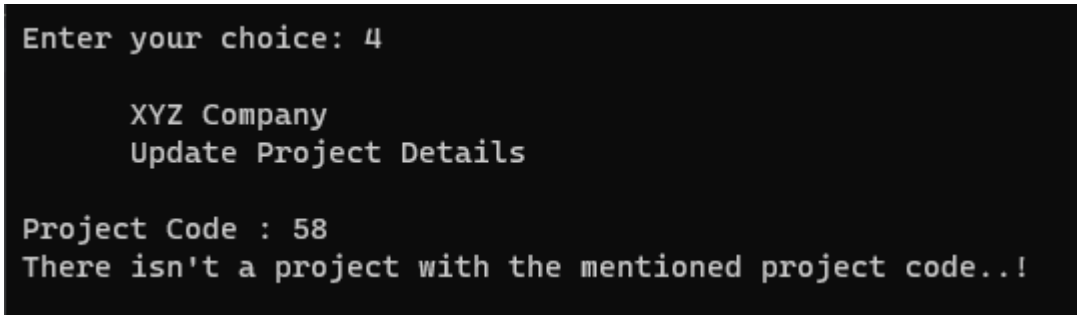


Figure 51

PROJECT STATISTICS

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

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

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

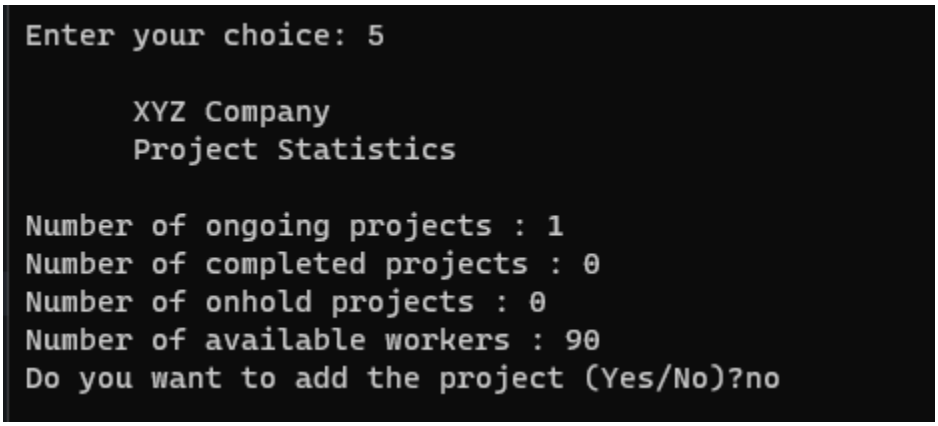


Figure 52

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

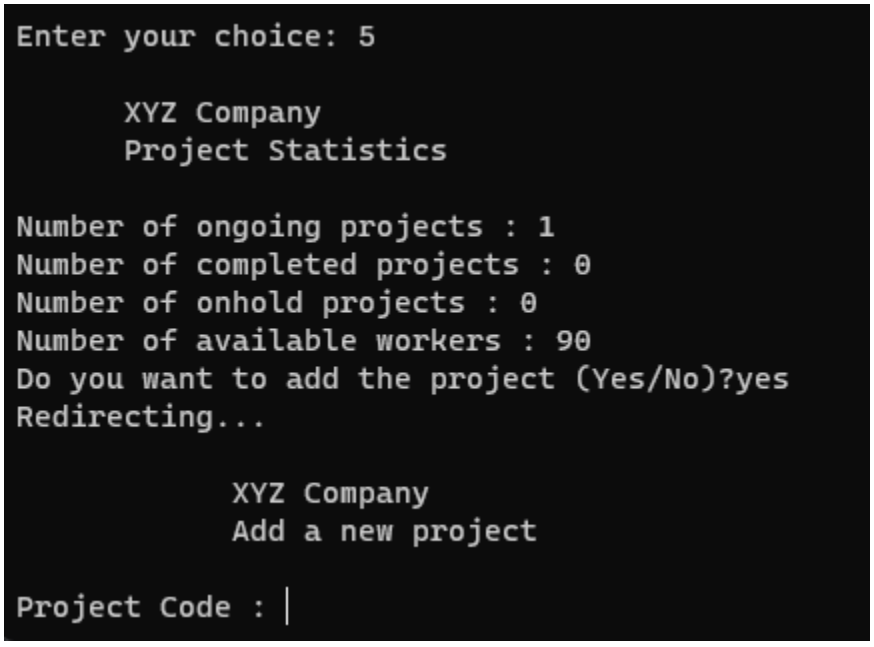


Figure 53

EXIT

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

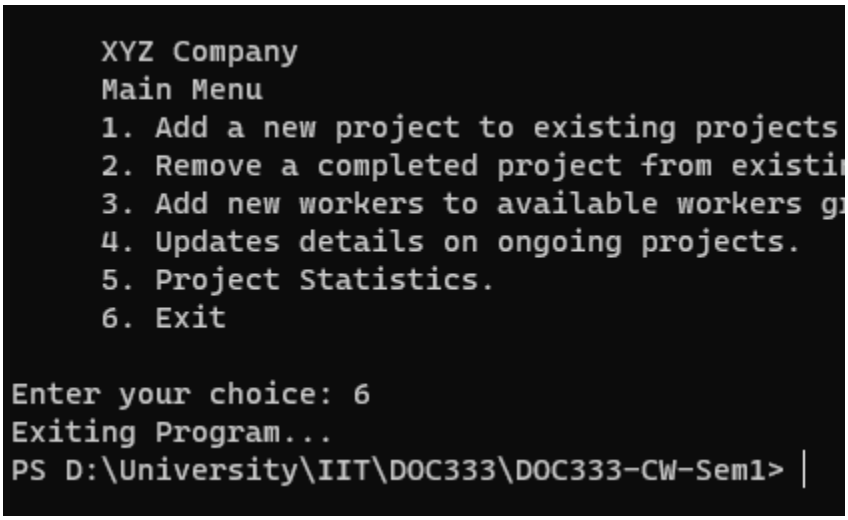


Figure 54

Conclusion

In conclusion, the solution that was implemented can manage XYZ Corporations 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 and 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.