BSc (Hons) Artificial Intelligence and Data Science

Module: CM1601

Programming Fundamentals

Individual Coursework Report

Module Leader: Ms. Sachinthani Perera

RGU Student ID : 2410212

IIT Student ID : 20233136

Student Name : Mathusha Kannathasan

Table of Contents

1) TABLE OF FIGURES

2) INTRODUCTION TO TECHEXPO

Sarah has organized an annual technology showcase event “TechExpo”. Here the innovative tech projects are presented to industry professionals and enthusiasts. The following code will indicate the imported functions, global functions, and the welcome banner of this competition.

I. PYTHON_CODE

```
################################################################################
import time
import random

# Initialize variables to store selected_projects and totalScore
selected_projects = {}
totalScore = {}

# Initialize an empty list to store projects_list
projects_list = []

print("\n*********TechExpo*********")
time.sleep(2)
print("----Welcome to TechExpo----")
time.sleep(2)
print()
################################################################################
```

These codes indicate the introduction part of the TechExpo event management system. The code imports “Random”, and “Time” for the various functions within the code. It will take a minute rest while processing the codes, and choosing the items randomly.

Furthermore, selected_projects and totalScore are initialized as empty dictionaries, and projects_list as empty lists. These will used to store data about the projects. The welcome banner will be displayed at the beginning of the program only.

3) INTRODUCTION TO FUNCTIONS IN THE SYSTEM

Different functions were created using the keyword “def” in Python. There is a main function main_menu. This includes 9 sub-functions, some use arguments while others do not. 1-APD for adding project details, 2-DPD for deleting unwanted projects permanently, 3-UPD for updating the project details, 4-VPD viewing the project details in ascending order based on project ID, 5-SPD saving the entered project details to the text file, 6-RSS for randomly selecting a project in each category, AWP for displaying the winners’ details, VAP for visualizing the award-winning projects using “*”, EXIT for terminating the program. Once we call the main_menu, the loop will start.

APD () - Adding Project Details

DPD () - Deleting Project Details

UPD () - Updating Project Details

VPD () - Viewing Project Details

SPD () - Saving Project Details to Text File

RSS () - Random Spotlight Selection

AWP (selected_projects) - Recording Awards and Recognitions

VAP (totalScore) - Visualizing Award-Winning Projects

EXIT () - Exiting the Program

After once we input option “6” then we can’t do the APD, DPD, UPD, VPD, and SPD. Because RSS, AWP, VAP, and EXIT are included in option “6” respectively. Therefore, we have to do the other options before inputting the option “6”. After each function, it will display the menu and choice again.

I. PYTHON_CODE

```
################################################################################
def main_menu():
    while True:
        print("--Select your choice from the following menu:")
        print()
        time.sleep(2)
        print("1. Adding Project Details (APD)")
        print("2. Deleting Project Details (DPD)")
        print("3. Updating Project Details (UPD)")
        print("4. Viewing Project Details (VPD)")
        print("5. Saving Project Details to Text File (SPD)")
        print("6. Random Spotlight Selection (RSS)")
        print("   Recording Awards and Recognitions (AWP)")
        print("   Visualizing Award-Winning Projects (VAP)")
        print("   Exiting the Program (EXIT)")
```

```
        print()
        time.sleep(1)
        choice = input("Enter your choice (1 to 6): ")

        match choice:

            case "1":
                print()
                APD()
            case "2":
                print()
                DPD()
            case "3":
                print()
                UPD()
            case "4":
                print()
                VPD()
            case "5":
                print()
                SPD()
            case "6":
                print()
                RSS()  #Call RSS()
                print()
                AWP(selected_projects)  # Call AWP(selected_projects)
                print()
                VAP(totalScore)  # Call VAP(totalScore)
                print()
                EXIT()  # Call EXIT()
                break
            case _:
                print("Invalid choice. Please enter the valid input.")
                print()
main_menu()
################################################################################
```

4) FUNCTION FOR ADDING PROJECT DETAILS: APD ( )

I. FLOWCHART

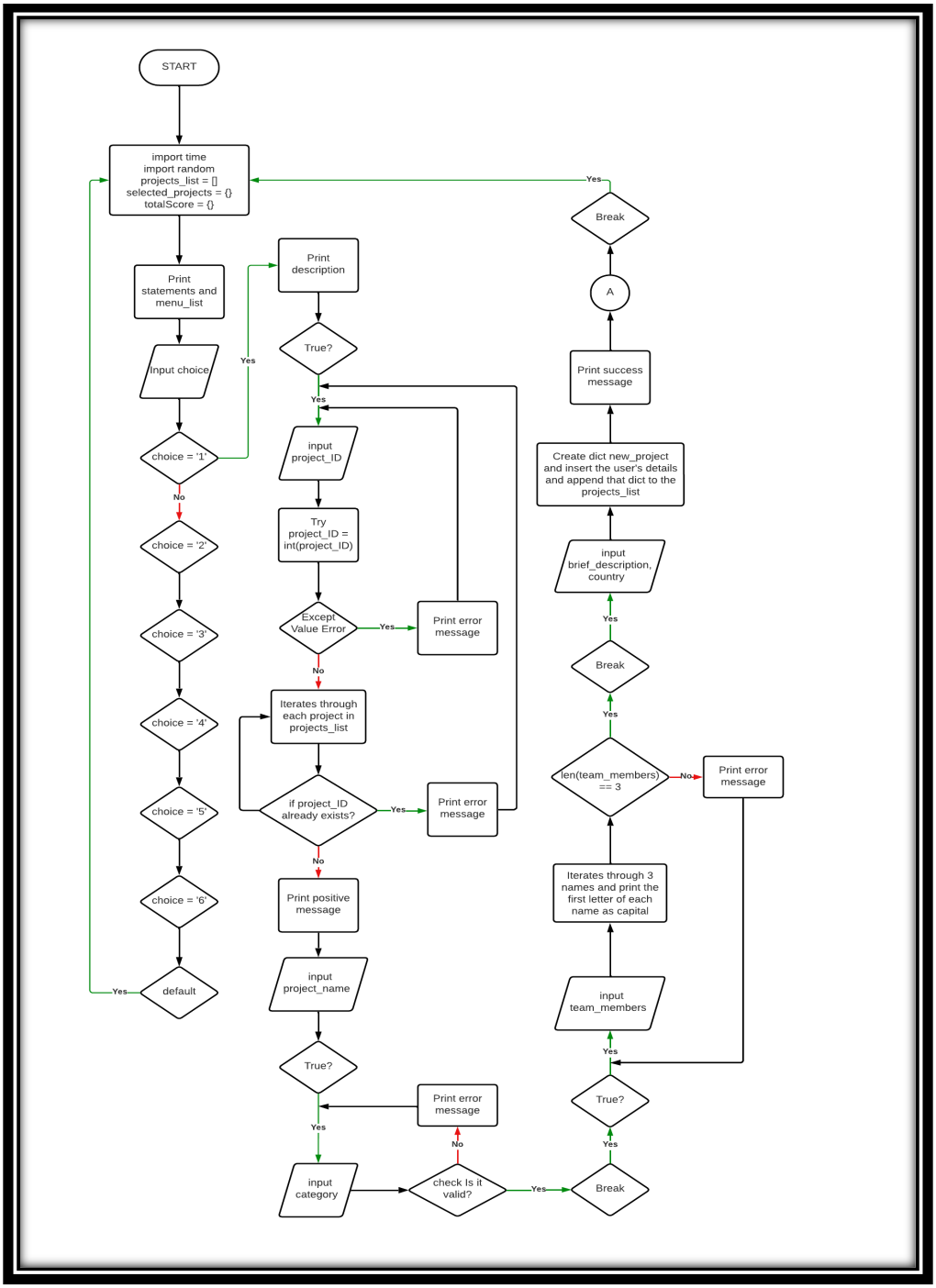


Figure 1 - APD.flowchart

II. PYTHON_CODE

```
##############################################################################
#Adding Project Details (APD)
#Allow the participants to input their project details
def APD():
    time.sleep(1)
    print("--Now you can add your project details--")
    while True:
        time.sleep(1)
        print()
        project_ID = input("Enter your project ID:")
        try:
            project_ID = int(project_ID)
            #Check whether the id already exists
            if project_ID in {project["project_ID"] for project in projects_list}:
                time.sleep(2)
                print()
                print("--The project Id is already exist in the list, please enter a unique ID--")
            else:
                time.sleep(1)
                print("---Hurray!!! Your ID is unique---")
                #Input project details
                project_name = input("Enter your project name:").capitalize()
                #Validate category input
                while True:
                    category = input("Enter the category to which your project belongs
                                     (AI/RT/ML):").upper()
                    if category in {"AI", "ML", "RT"}:
                        break
                    else:
                        print("Invalid category! Please enter (Al/ML/RT).")
                # Input team members' names with validation for exactly 3 members
                while True:
                    team_members = [name.capitalize()
                                    for name in input("Enter your team members' names:").split(",")]
                    if len(team_members) == 3:
                        break
                    else:
                        print("Please enter exactly three team members.")
                brief_description = input("Enter a brief description about the
                project:").capitalize()
                country = input("Enter your country:").capitalize()
                #Create a dictionary to input the user's details
                new_project = {"project_ID": project_ID,
                               "project_name": project_name,
                               "category": category,
                               "team_members": team_members,
                               "brief_description": brief_description,
                               "country": country}
```

```
            # Append the project details to the projects_list
            projects_list.append(new_project)
            time.sleep(2)
            print("\n--Project details have been added successfully--\n")
            SPD()
            break
        time.sleep(1)
    except ValueError:
        print("Invalid input! Please enter a valid integer for project_ID.")
##########################################################################
```

III. DESCRIPTION

When the main_menu () choice is asked to input if the user inputs option “1” – APD (). The function APD is used to add the project details of projects that are submitted for TechExpo. It starts with a display of a welcome banner of addition. The “While” loop is used here to get the details of the project. If the condition is True, then it will sleep for 1 second and let the user input the project_ID, then it will let to Try Except condition, and it will pass to other condition if only the entered ID is an integer again. Else it will raise ValueError with an invalid statement and let the user enter the integer. Then it will iterate through each project in the list (projects_list) and check whether the entered ID already exists. If it already exists then it will allow the user to input the valid ID. The loop will continue until the integer is valid, the correct validation will display by a congrats message.

Moreover, the user can input the project_name, even if the user enters the name in the simple letter the system will change the first letter to capital. Then again a while loop will start to enter the category. Here I assume that there are only 3 categories {“AI”, “ML”, and “RT”}. Therefore, it will break this loop, if only the entered category is in the list. Else it will let the user enter the category again until the category is valid. Then the user can input the team member's name. But I assume as the competition will allow only 3 members and I open a list to include these names, the names will split with a comma by using a built-in function split (“,”) and the first letter of each name will convert as capital. Here also a while loop starts, if only the length of the members is 3, it will break. It will continue until it’s true.

After that user can input a brief description and country name. Here also first letter will convert as capital using the built-in function capitalize (). Then a dictionary “new_project” will be created to add the user-entered details of each new project. Then each dictionary “new_project” is appended to list “projects_list”. After that a success message will display and then SPD() will proceed. Then the loop will break and back to main_menu again.

IV. OUTPUTS

Integer checking, duplication of ID checking, category type checking, and checking the count of team members are implemented in these outputs. When you refer to these screenshots after details have been added successfully it will lead to SPD () for saving the details in the text file and then back to main_menu () again.

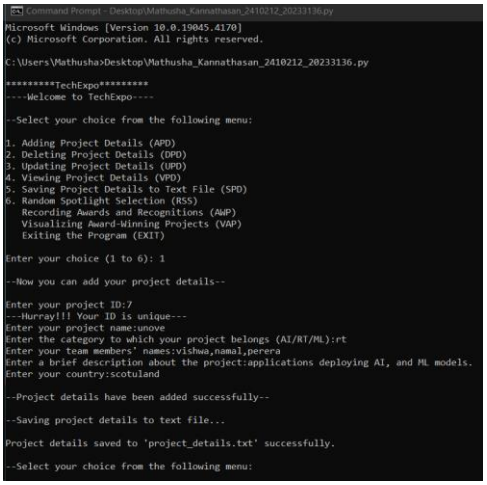


Figure 2 - APD.output1

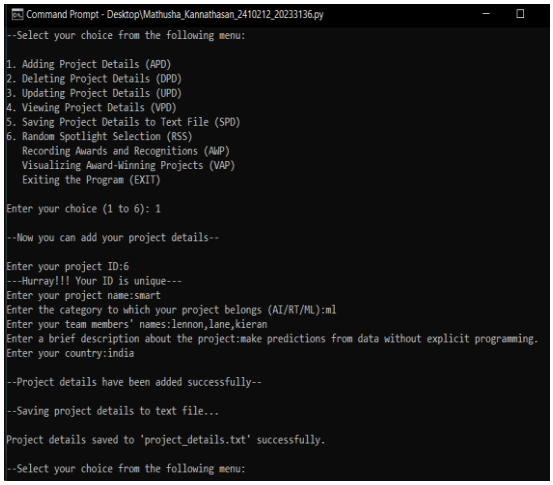


Figure 3 - APD.output2

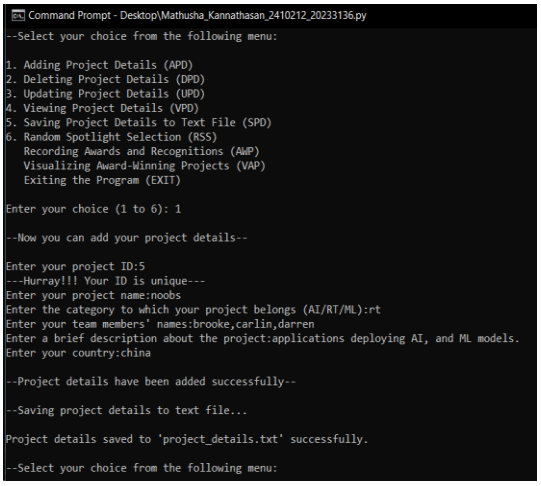


Figure 4 - APD.output3

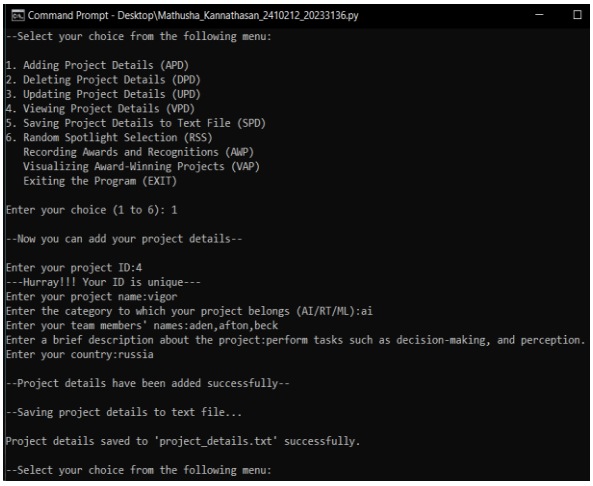


Figure 5 - APD.output4

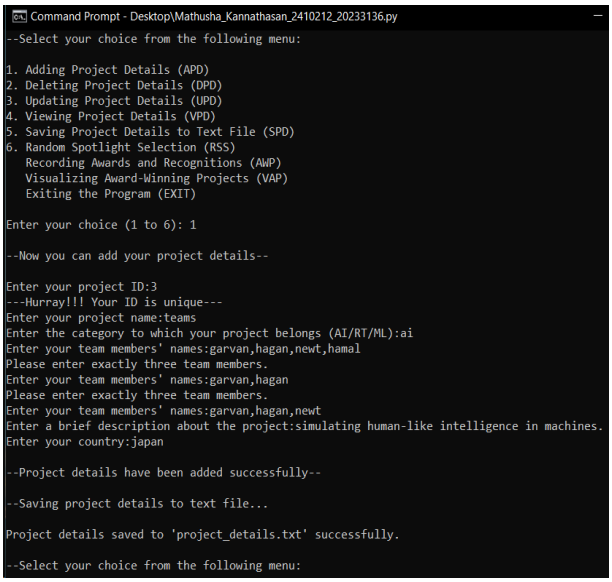


Figure 6 - APD.output5

Figure 7 - APD.output6

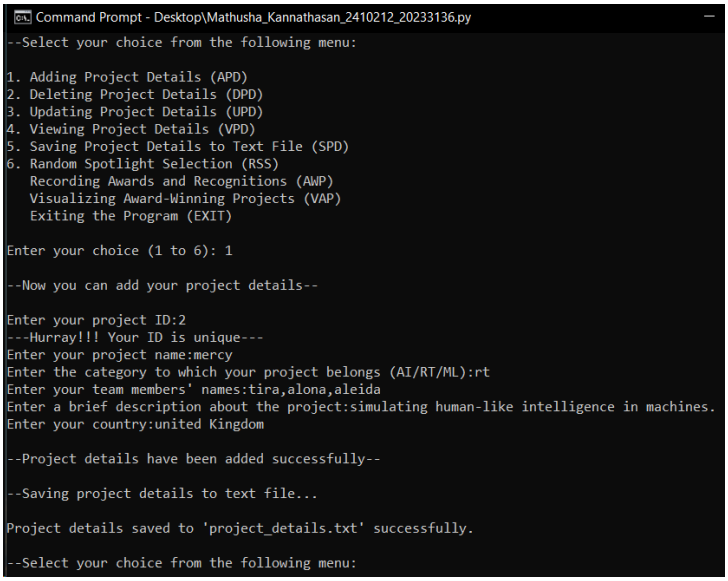


Figure 8 - APD.output7

5) FUNCTION FOR DELETING PROJECT DETAILS: DPD ( )

I. FLOW_CHART

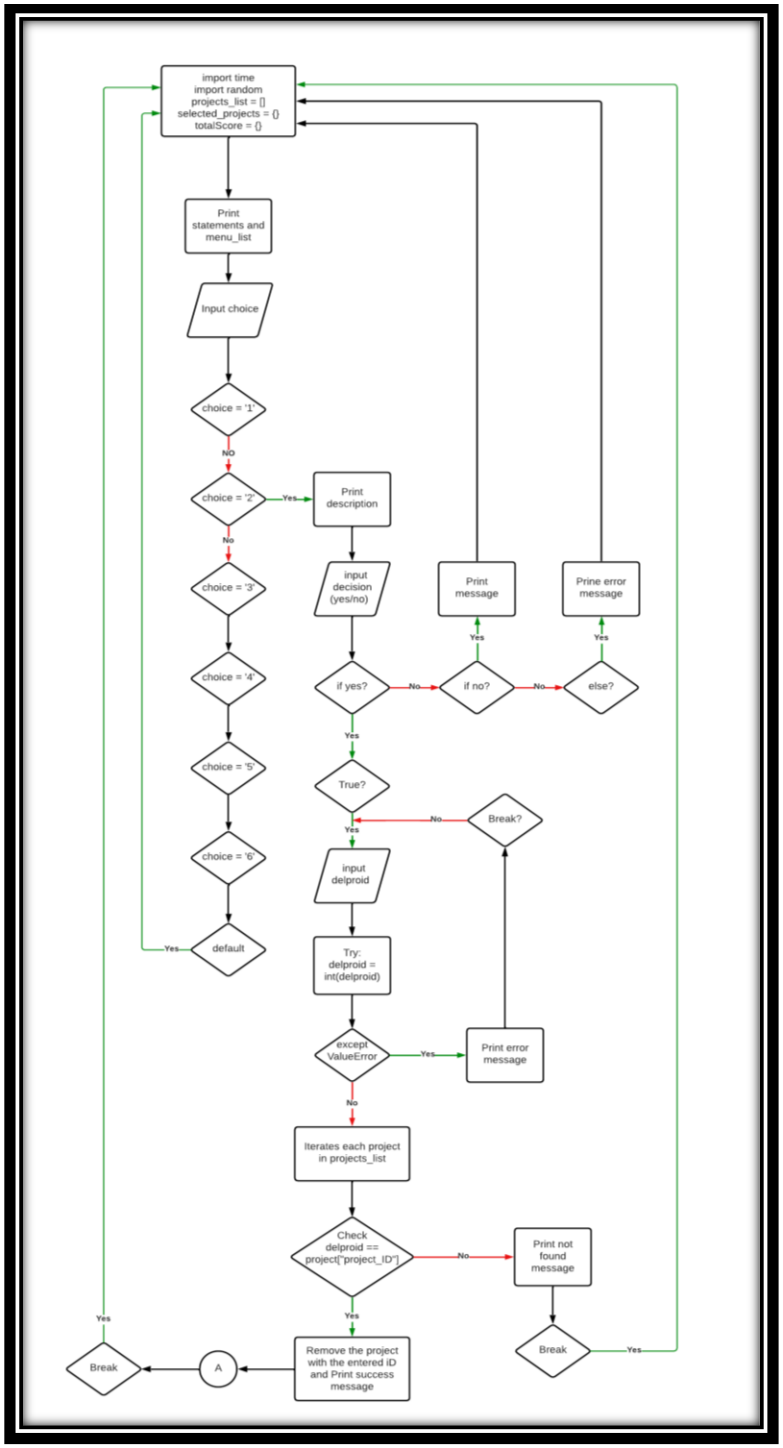


Figure 9 - DPD.flowchart

II. PYTHON_CODE

```
#############################################################################
#Deleting Project Details(DPD)
#Here the user can delete the unwanted project by finding the projects using the project ID

def DPD():
    time.sleep(1)
    print("Think twice.... Will you intend to delete your project?")
    print("--If you delete your project, then you can not participate--")
    print()
    time.sleep(2)
    decision = input("Do you want to delete your project? (yes/no):")
    print()
    if decision.lower() == "yes":
        time.sleep(1)
        while True:
            delproid = input("Enter the project ID you want to delete:")
            try:
                delproid = int(delproid)
                print()
            except ValueError:
                print("Invalid input! Please enter a valid integer for project_ID.")
                print()
                continue
            #Check whether the id already exists
            for project in projects_list:
                if delproid == project["project_ID"]:
                    #Remove the project from the projects list
                    projects_list.remove(project)
                    time.sleep(1)
                    print(f"Project with ID {delproid} has been successfully deleted.")
                    print()
                    SPD()
                    break
            else:
                print(f"Project with ID {delproid} is not found in our list.")
            break
    elif decision.lower() == "no":
        print("You can participate, your project details will not be deleted.")
    else:
        print("Invalid option. Please enter 'yes' or 'no'.")
    time.sleep(1)
    print()
#############################################################################
```

III. DESCRIPTION

When the main_menu () choice is asked to input if the user inputs option “2” – DPD (). The function DPD is used to delete the project details of projects that are submitted for TechExpo. The function starts with displaying the description of deleting. After that user can input the decision (yes/no) and the system will convert the entered input into lowercase. If the decision is yes, the “while” loop will start and let the user input the ID that they want to delete. then it will let to Try Except condition, and it will pass to other condition if only the entered ID is an integer. Else it will raise ValueError with an invalid statement and let the user enter the integer again. Then it will iterate through each project in the list (projects_list) and check whether the entered ID already exists. If it already exists then it will delete the project with the entered ID using the built-in function remove () and display the success message. Otherwise, if it does not exist then it will display the not found message, proceed to the SPD(), and return to main_menu (). If the option == (no / invalid), the system returns to main_menu ().

IV. OUTPUTS

Integer checking for ID, existence checking for ID, and decision checking are implemented in these outputs. When you refer to these screenshots after details have been successfully deleted it will lead to SPD () for saving the details in the text file and then back to main_menu () again. Otherwise, it will directly lead to main_menu ().

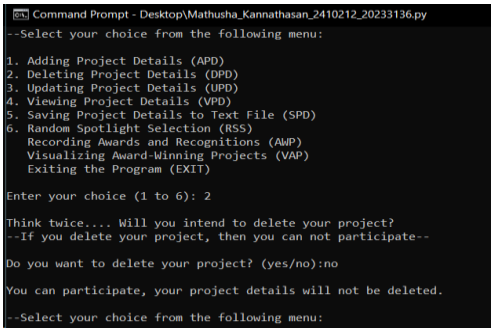


Figure 10 - DPD.output1

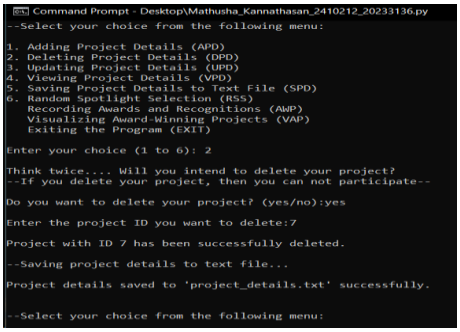


Figure 11 - DPD.output2

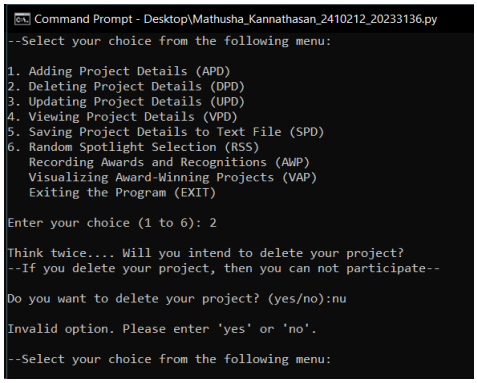


Figure 12 - DPD.output3

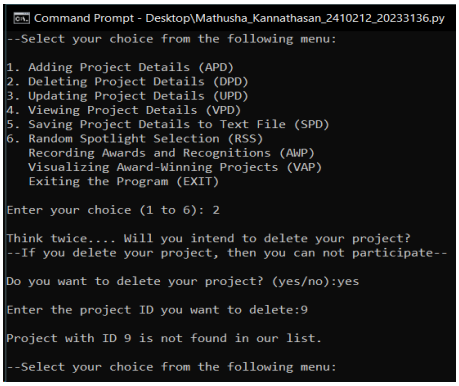


Figure 13 - DPD.output4

6) FUNCTION FOR UPDATING PROJECT DETAILS: UPD ( )

I. FLOW_CHART

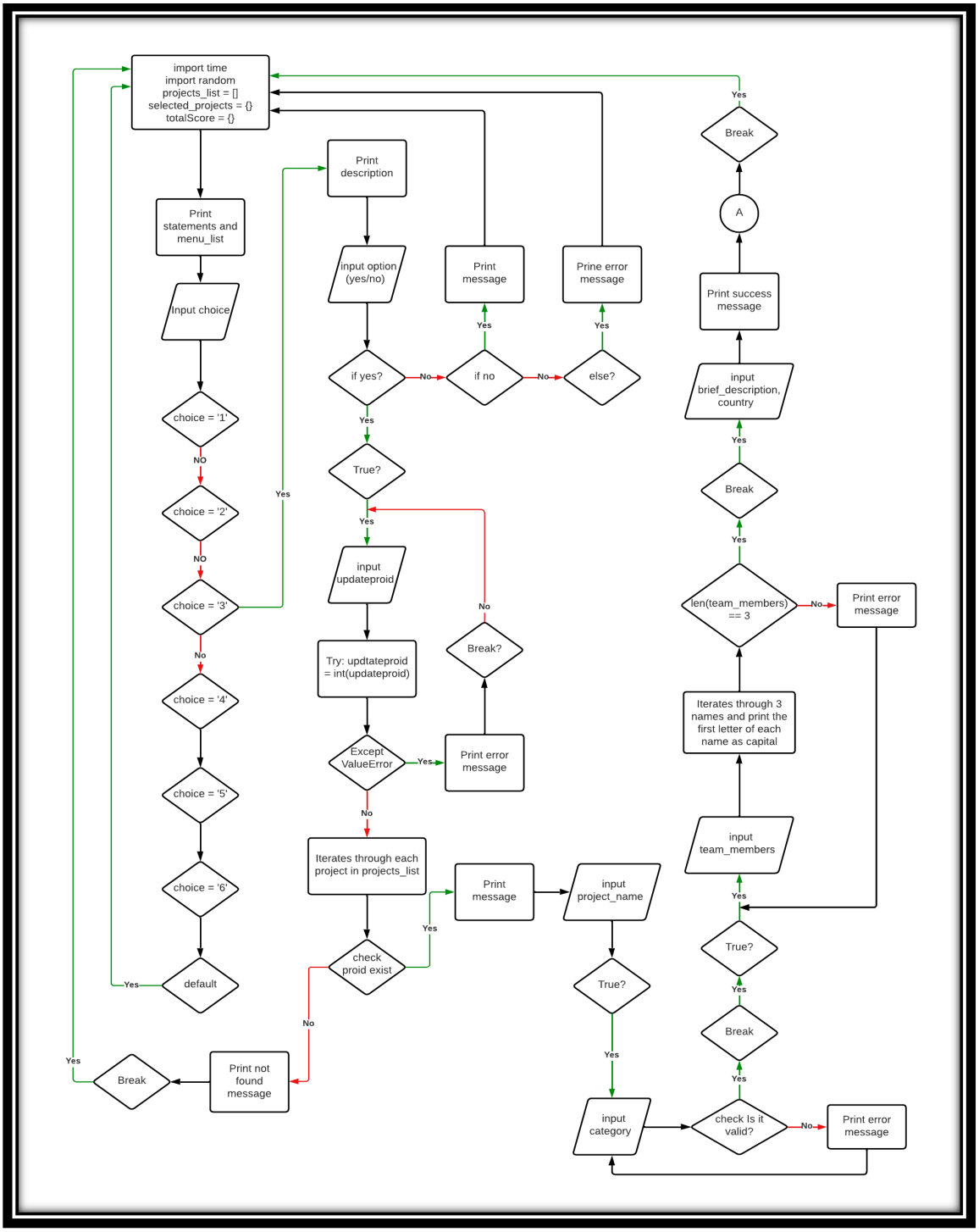


Figure 14 - UPD.flowchart

II. PYTHON_CODE

```
#############################################################################
#Updating Project Details(UPD)
#Here the project details can be updated by finding the projects using the project ID

def UPD():
    time.sleep(1)
    print("Are you really want to update?")
    print("If you update your details, then your previous details will be updated.")
    print()
    time.sleep(1)
    option = input("Do you want to update your project details? (yes/no):")
    print()
    if option.lower() == "yes":
        time.sleep(1)
        while True:
            updateproid = input("Enter the project ID of the project you want to update:")
            try:
                updateproid = int(updateproid)
                print()
            except ValueError:
                print("Invalid input! Please enter a valid integer for project_ID.")
                continue
            #Check whether the ID already exists and update the details
            for project in projects_list:
                if updateproid == project["project_ID"]:
                    print("Now, you can update the details.")
                    print()
                    time.sleep(1)
                    #Update details seperately
                    project["project_name"] = input("Enter the updated project
                                                    name:").capitalize()
                    # Validate category input
                    while True:
                        category = input("Enter the updated category (AI/RT/ML):").upper()
                        if category in ["AI", "RT", "ML"]:
                            project["category"] = category
                            break
                        else:
                            print("Invalid category! Please enter (AI/ML/RT).")
                            #Input team members' names exactly 3 members
                    while True:
                        team_members = [name.capitalize()
                                for name in input("Enter your team members' names: ").split(",")]
                        if len(team_members) == 3:
                            project["team_members"] = team_members
                            break
                        else:
                            print("Please enter exactly three team members.")
```

```
            project["brief_description"] = input("Enter the updated brief description
            about the project:").capitalize()
            project["country"] = input("Enter the updated country:").capitalize()
            print()
            print("Project details has been updated successfully.")
            print()
            SPD()
            break
        else:
            print("Project with ID", updateproid, "is not found in our list.")
        break
    elif option.lower() == "no":
        print("You can participate, your project details will not be updated.")
    else:
        print("Invalid option. Please enter 'yes' or 'no'.")
    time.sleep(1)
    print()
##############################################################################
```

III. DESCRIPTION

When the main_menu () choice is asked to input if the user inputs option "3" – UPD (). The function UPD is used to update the project details of projects that are submitted for TechExpo. This function begins with displaying the update description. After that user can input the option (yes/no) and the system will convert the entered input into lowercase. If the decision is yes, the "while" loop will start and let the user input the ID that they want to update. then it will let to Try Except condition, and it will pass to other condition if only the entered ID is an integer. Else it will raise ValueError with an invalid statement and let the user enter the integer again. Then it will iterate through each project in the list (projects_list) and check whether the entered ID already exists. If it already exists then it will start the project with the entered ID. For that, it will ask the user to update the project name. After that, the code is implemented as a similar flow to the APD function. Until the user updates the country details. Then it will display a successfully updated message and proceed to the function SPD(). If the ID does not exist then it will display a not found message and return to main_menu. If the option = (no / invalid), then the system will return to main_menu.

IV. OUTPUTS

Integer checking for ID, existence checking for ID, decision checking, duplication of ID checking, category type checking, and checking the count of team members are implemented in these outputs. When you refer to these screenshots after details have been successfully updated it will lead to SPD () for saving the details in the text file and then back to main_menu () again. Otherwise, it will directly lead to main_menu ().

```
Command Prompt - Desktop\Mathusha_Kannathasan_2410212_20233136.py
--Select your choice from the following menu:

1. Adding Project Details (APD)
2. Deleting Project Details (DPD)
3. Updating Project Details (UPD)
4. Viewing Project Details (VPD)
5. Saving Project Details to Text File (SPD)
6. Random Spotlight Selection (RSS)
   Recording Awards and Recognitions (AWP)
   Visualizing Award-Winning Projects (VAP)
   Exiting the Program (EXIT)

Enter your choice (1 to 6): 3

Are you really want to update?
If you update your details, then your previous details will be updated.

Do you want to update your project details? (yes/no):nu

Invalid option. Please enter 'yes' or 'no'.

--Select your choice from the following menu:
```

Figure 15 - UPD.output1

```
Command Prompt - Desktop\Mathusha_Kannathasan_2410212_20233136.py
--Select your choice from the following menu:

1. Adding Project Details (APD)
2. Deleting Project Details (DPD)
3. Updating Project Details (UPD)
4. Viewing Project Details (VPD)
5. Saving Project Details to Text File (SPD)
6. Random Spotlight Selection (RSS)
   Recording Awards and Recognitions (AWP)
   Visualizing Award-Winning Projects (VAP)
   Exiting the Program (EXIT)

Enter your choice (1 to 6): 3

Are you really want to update?
If you update your details, then your previous details will be updated.

Do you want to update your project details? (yes/no):no

You can participate, your project details will not be updated.

--Select your choice from the following menu:
```

Figure 16 - UPD.output2

```
Command Prompt - Desktop\Mathusha_Kannathasan_2410212_20233136.py
--Select your choice from the following menu:

1. Adding Project Details (APD)
2. Deleting Project Details (DPD)
3. Updating Project Details (UPD)
4. Viewing Project Details (VPD)
5. Saving Project Details to Text File (SPD)
6. Random Spotlight Selection (RSS)
   Recording Awards and Recognitions (AWP)
   Visualizing Award-Winning Projects (VAP)
   Exiting the Program (EXIT)

Enter your choice (1 to 6): 3

Are you really want to update?
If you update your details, then your previous details will be updated.

Do you want to update your project details? (yes/no):8

Invalid option. Please enter 'yes' or 'no'.

--Select your choice from the following menu:
```

Figure 17 - UPD.output3

```
Command Prompt - Desktop\Mathusha_Kannathasan_2410212_20233136.py
--Select your choice from the following menu:

1. Adding Project Details (APD)
2. Deleting Project Details (DPD)
3. Updating Project Details (UPD)
4. Viewing Project Details (VPD)
5. Saving Project Details to Text File (SPD)
6. Random Spotlight Selection (RSS)
   Recording Awards and Recognitions (AWP)
   Visualizing Award-Winning Projects (VAP)
   Exiting the Program (EXIT)

Enter your choice (1 to 6): 3

Are you really want to update?
If you update your details, then your previous details will be updated.

Do you want to update your project details? (yes/no):yes

Enter the project ID of the project you want to update:8

Project with ID 8 is not found in our list.

--Select your choice from the following menu:
```

Figure 18 - UPD.output4

```
Command Prompt - Desktop\Mathusha_Kannathasan_2410212_20233136.py
--Select your choice from the following menu:

1. Adding Project Details (APD)
2. Deleting Project Details (DPD)
3. Updating Project Details (UPD)
4. Viewing Project Details (VPD)
5. Saving Project Details to Text File (SPD)
6. Random Spotlight Selection (RSS)
   Recording Awards and Recognitions (AWP)
   Visualizing Award-Winning Projects (VAP)
   Exiting the Program (EXIT)

Enter your choice (1 to 6): 3

Are you really want to update?
If you update your details, then your previous details will be updated.

Do you want to update your project details? (yes/no):yes

Enter the project ID of the project you want to update:2

Now, you can update the details.

Enter the updated project name:merdy
Enter the updated category (AI/RT/ML):uk
Invalid category! Please enter (AI/ML/RT).
Enter the updated category (AI/RT/ML):rt
Enter your team members' names: tira
Please enter exactly three team members.
Enter your team members' names: tira,alonae,aleide
Enter the updated brief description about the project:applications deploying AI, and ML models.
Enter the updated country:norway

Project details has been updated successfully.

--Saving project details to text file...

Project details saved to 'project_details.txt' successfully.


--Select your choice from the following menu:
```

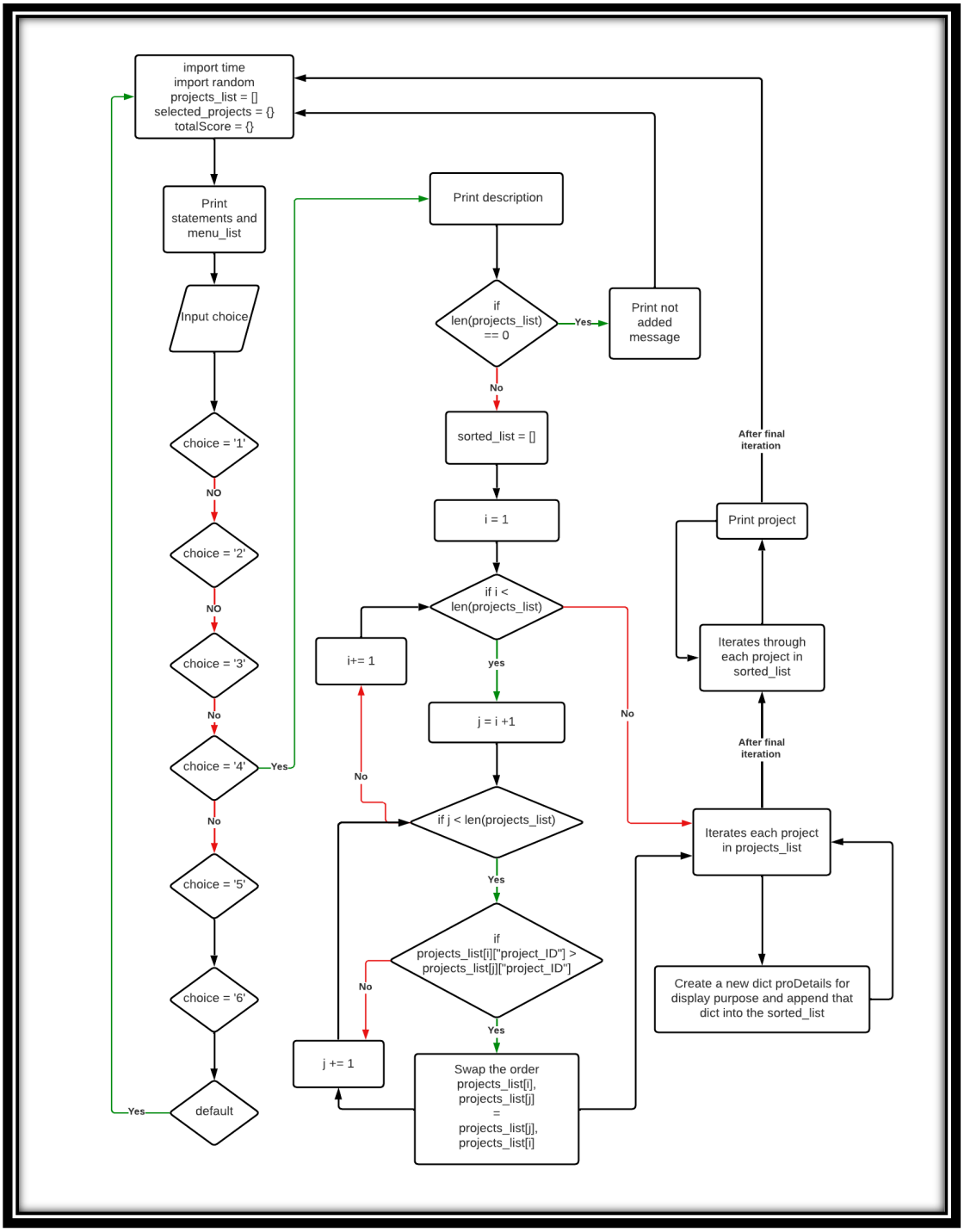
Figure 19 - UPD.output5

```
Command Prompt - Desktop\Mathusha_Kannathasan_2410212_20233136.py
--Select your choice from the following menu:

1. Adding Project Details (APD)
2. Deleting Project Details (DPD)
3. Updating Project Details (UPD)
4. Viewing Project Details (VPD)
5. Saving Project Details to Text File (SPD)
6. Random Spotlight Selection (RSS)
   Recording Awards and Recognitions (AWP)
   Visualizing Award-Winning Projects (VAP)
   Exiting the Program (EXIT)

Enter your choice (1 to 6): 3

Are you really want to update?
If you update your details, then your previous details will be updated.

Do you want to update your project details? (yes/no):yes

Enter the project ID of the project you want to update:ju
Invalid input! Please enter a valid integer for project_ID.
Enter the project ID of the project you want to update:5

Now, you can update the details.

Enter the updated project name:noobs
Enter the updated category (AI/RT/ML):rt
Enter your team members' names: brooke,carlino,darren,funel
Please enter exactly three team members.
Enter your team members' names: brooke,carlino,darren
Enter the updated brief description about the project:applications deploying AI, and ML models.
Enter the updated country:china

Project details has been updated successfully.

--Saving project details to text file...

Project details saved to 'project_details.txt' successfully.


--Select your choice from the following menu:
```

Figure 20 - UPD.output6

7) FUNCTION FOR VIEWING PROJECT DETAILS: VPD ( )

I. FLOW_CHART



II. PYTHON_CODE

```
#############################################################################
#Viewing Project Details(VPD)
#Here the inserted projects will be listed in ascending order based on the project_ID

def VPD():
    time.sleep(1)
    print("--The project details will be displayed in ascending order based on project_ID--")
    time.sleep(2)
    print()
    if len(projects_list) == 0:
        print("The projects has not been added yet now.")
    else:
        sorted_list = []
        for i in range(len(projects_list)): #outer loop
            for j in range(i +1, len(projects_list)): #inner loop
                if projects_list[i]["project_ID"] > projects_list[j]["project_ID"]:
                    #Swap the order in ascending, like min(project_ID), max(project_ID)
                    projects_list[i], projects_list[j] = projects_list[j], projects_list[i]

        for project in projects_list:
            #Format project details for display
            proDetails= {"project_ID": project["project_ID"],
                         "project_name": project["project_name"],
                         "category": project["category"],
                         "team_members": project["team_members"],
                         "brief_description": project["brief_description"],
                         "country": project["country"]}
            sorted_list.append(proDetails)
        time.sleep(1)
        for project in sorted_list:  # Iterate through sorted_list to print each project's details
on a new line
            print(project)
            print()
    time.sleep(1)
    print()
#############################################################################
```

III. DESCRIPTION

When the main_menu () choice is asked to input if the user inputs option “4” – VPD (). The function VPD is used to view the project details of projects that are submitted for TechExpo in ascending order based on the project ID. This function begins with displaying the description of viewing in ascending order. If the length of the projects_list is zero, the system will display the not added message. Otherwise, it will create an empty list “sorted_list” to add the sorted project details. Then it will start to iterate through ‘i’ in the range of len(projects_list). That means it will iterate until the last entered project. Within that inner loop, it will start another loop to compare

the other ‘j’ projects to each other until the last with the “i” project. If project ‘i’s’ project ID > j project’s project ID then it will be generated in ascending order like projects_list[j], projects_list[i]. Like this, it will compare each project separately with the “i” project. After the inner loop is completed, the system will again start the outer loop. Therefore, it will continue until the last project. After the final iteration of ‘i’, another loop will start to iterate through each sorted project in projects_list to find the details and add to a dictionary “proDetails”. After the final iteration, the proDetails will append to sorted_list. After that to maintain space between each sorted project details, it will iterate through each project in sorted_list and print project and a space after that. Then it will back to main_menu again.

IV. OUTPUT

The output for VPD has resulted in the ascending order based on project_ID. Then it will back to main_menu again.

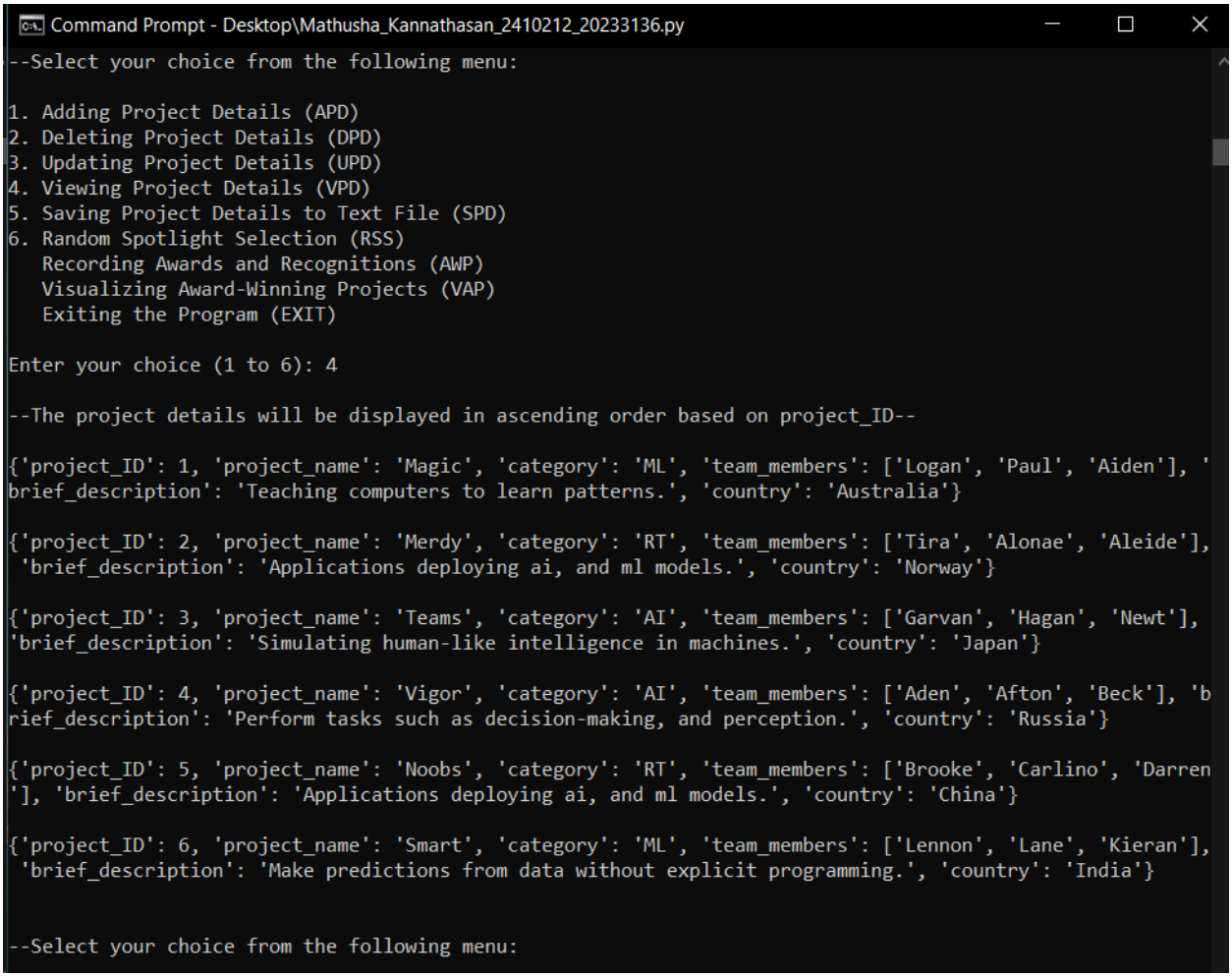


Figure 21 - VPD.output

8) FUNCTION FOR SAVING PROJECT DETAILS TO THE TEXT FILE: SPD ( )

I. FLOW_CHART

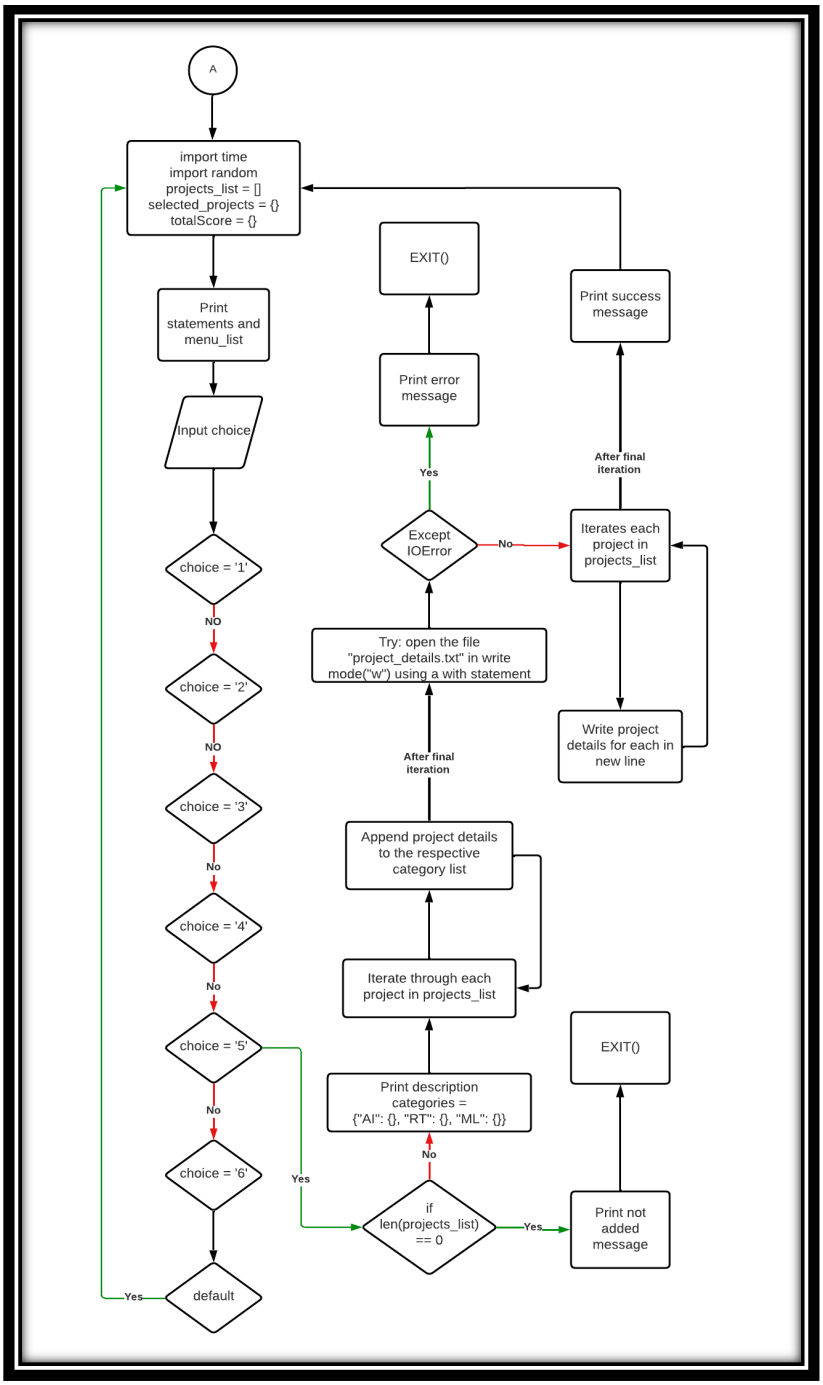


Figure 22 - SPD.flowchart

II. PYTHON_CODE

```
###############################################################################
# Saving Project Details to Text File(SPD)
# Here the project details will be saved into a text file.
def SPD():
    time.sleep(1)
    if len(projects_list) == 0:
        print("The projects have not been added yet.")
        print()
        EXIT()
    else:
        print("--Saving project details to text file...")
        print()
        time.sleep(1)
        # Initialize dictionaries to store project details by category
        categories = {"AI": [], "RT": [], "ML": []}
        for project in projects_list:
            # Append project details to the respective category list
            categories[project["category"]].append(project)
        try:
            with open("desktop/project_details.txt", "w") as file: # Open file in append mode
                for category, projects in categories.items():
                    file.write("\n")
                    file.write(f"********** {category} Projects **********\n")
                    file.write("\n")
                    for project in projects:
                        # Write project details in the text file
                        file.write("---Project Details---\n")
                        file.write(f"Project ID: {project['project_ID']}\n")
                        file.write(f"Project Name: {project['project_name']}\n")
                        file.write(f"Category: {project['category']}\n")
                        # Join team members using ','
                        file.write(f"Team Members: {', '.join(project['team_members'])}\n")
                        file.write(f"Brief Description: {project['brief_description']}\n")
                        file.write(f"Country: {project['country']}\n")
                        file.write("\n")
                print("Project details saved to 'project_details.txt' successfully.")
                print()
        except IOError:
            print("Error: Unable to access the file.")
            print()
            EXIT()
###############################################################################
```

III. DESCRIPTION

When the main_menu () choice is asked to input if the user inputs option “5” – SPD () not only for that but also it is used after APD (), UPD (), and DPD (). The function SPD is used to save the project details of projects that are submitted for TechExpo to a text file ‘project_details.txt’. If the length of projects_list is zero, then will display the not added message. Else it will display a description of saving project details to a text file. A dictionary “categories” is created with empty lists to store projects for each category in it. After that, it iterates through each project through projects_list and appends the project details to the respective category list. Then after the final iteration Try-Except condition will start to check whether the text file can be accessed for writing using the “with” condition because this condition will close the file automatically when the work is done. If not, an IO error will raise and EXIT () the program. Otherwise, it will iterate through each category’s project in projects_list, and write each project’s detail in a new line with the subheading ‘Project Details’ under the main category heading. This will help to read line-by-line purposes. After that, the success message of saving will be displayed. Here I assumed that all categories’ project details were included in a single text file. Therefore, it’s categorized inside a single file. Furthermore, I assumed it was a one-day event, and regarding that, once the user starts RSS () then they cannot redo the earlier function. Therefore when they re-run this program they will see the empty list while they proceed with the function VPD ().

IV. OUTPUT

After saving the details to the text file, it will return to main_menu () again.

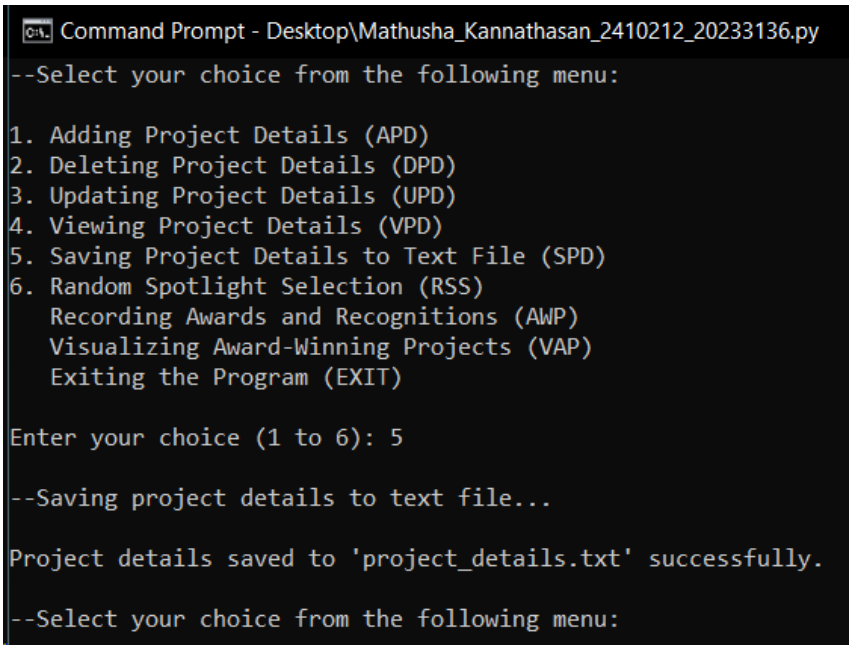


Figure 23 - SPD.output

9) FUNCTION FOR RANDOM SPOTLIGHT SELECTION: RSS ( )

I. FLOW_CHART FOR OPTION ‘6’

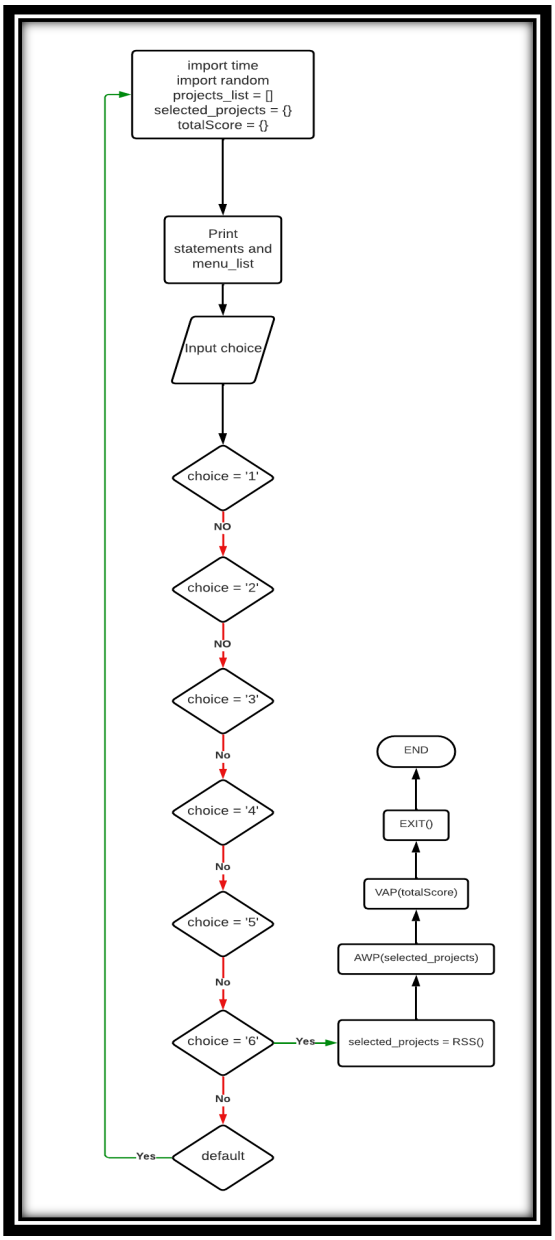


Figure 24 - Option'6'.flowchart

When the main_menu () choice is asked to input if the user inputs option “6” – the continuous function will start. After input ‘6’, the user cannot do other functions above option ‘6’. Option ‘6’ will run in the order of RSS (), AWP (selected_projects), VAP (totalScore), and EXIT () respectively.

II. FLOW_CHART FOR RSS ( )

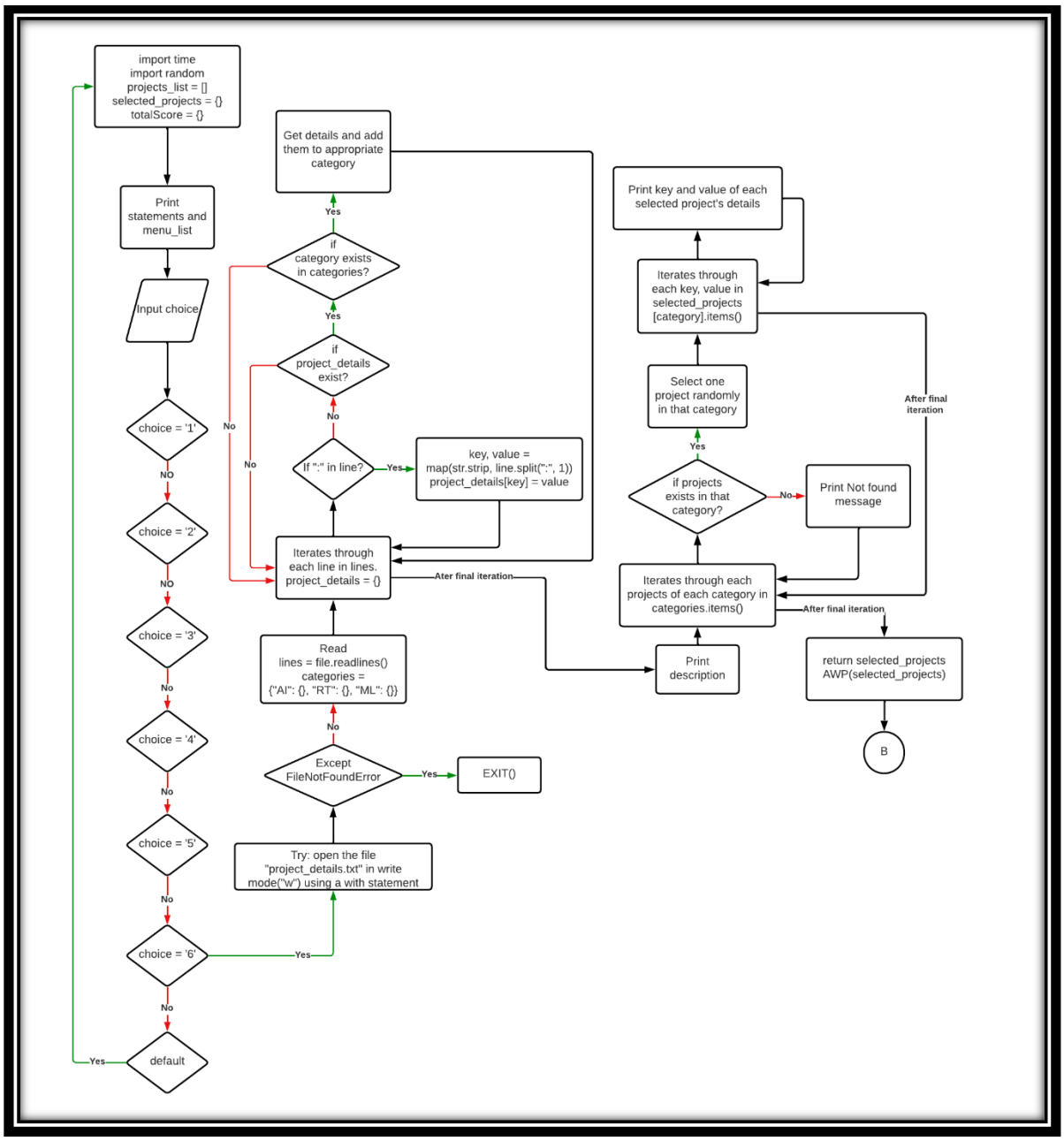


Figure 25 - RSS.flowchart

III. PYTHON_CODE

```
##############################################################################
#Random Spotlight Selection(RSS)
#3 cateogories are assigned. Here One project has been randomly selected from each
category.

def RSS():
   try:
      time.sleep(1)
      with open("desktop/project_details.txt", "r") as file:
         # Read all lines from the file
         lines = file.readlines()

      # Initialize empty dictionaries to store projects for each category
      categories = {"AI": {}, "RT": {}, "ML": {}}

      # Iterate through lines to extract project details
      project_details = {}
      for line in lines:
         # Strip removes any leading whitespace
         line = line.strip()
         # Check if the line contains a colon to split into key-value pairs
         if ":" in line:
            key, value = map(str.strip, line.split(":", 1))
            # Split the current line at the first occurrence of ":" and limit to only one
            # and strip whitespace from both parts
            project_details[key] = value
         else:
            # If a line doesn't contain a colon, it might be the start of a new project section
            # If project details exist, add them to the appropriate category
            if project_details:
               category = project_details.get("Category")
               if category in categories:
                  categories[category][project_details["Project ID"]] = project_details
               # Reset project details dictionary for the next project
               project_details = {}

      # Randomly select one project from each category
      print("--Randomly selected projects will be listed down--")
      print()
      time.sleep(1)
      for category, projects in categories.items():
         if projects:  # Check if there are projects in the category
            selected_projects[category] = random.choice(list(projects.values()))
            print(f"--Randomly Selected Project Details ({category}):")
            # Display randomly selected project details of each category
            for key, value in selected_projects[category].items():
               print(f"{key}: {value}")
            time.sleep(1)
            print()
```

```
            else:
                print(f"No projects found in category {category}.")

        return selected_projects

    except FileNotFoundError:
        print("Project details file not found.")
        EXIT()

    print()
##############################################################################
```

IV. DESCRIPTION

The function RSS is equalized with selected_projects as it returns selected_projects for AWP. RSS is used to do random spotlight selection. That means one project will be selected from each category, as I assumed 3 categories, it will randomly select 3 projects as finalists.

Here the function starts with the Try-Except condition. If the text file 'project_details.txt' cannot be accessed for reading line by line, then it will raise an error FileNotFoundError and terminate the program. Otherwise, it will open the file for reading purposes using the "with" condition because this condition will close the file automatically when the work is done. The file will be read by each line in the file. A dictionary "categories" is created with empty dictionaries to store projects for each category in it. Another empty dictionary project_details is also created. Then it iterates through each line in lines, and if it finds ":" then it splits the line into key-value pairs, where the key is the part before the colon and the value is the part after it. These key-value pairs represent different details of projects. If a line doesn't contain a colon, it assumes it's the start of a new project section. It stores the previous project details (if any) into the appropriate category in the 'categories' dictionary. Again, it will reset project_details to empty start each iteration. After the iteration through the final line, it will iterate through the teaching category and its projects in "categories". It will check the existence of projects in each category, if it does not exist in a category, then it will print a not found message for that category. If it finds projects, then it will choose randomly a project from that category using the import function random. It will be added to the dictionary "selected_projects". After that it will iterate through each key, and value in selected_projects and print the key and value of the selected project under the subtitle of its category. Then again it will iterate through another category like this. Finally, it will return selected_projects and start to attempt the AWP (selected_projects) function.

V. OUTPUT

Exactly one project has been randomly selected for each category by reading the text file line by line. After that, it will lead to AWP ().

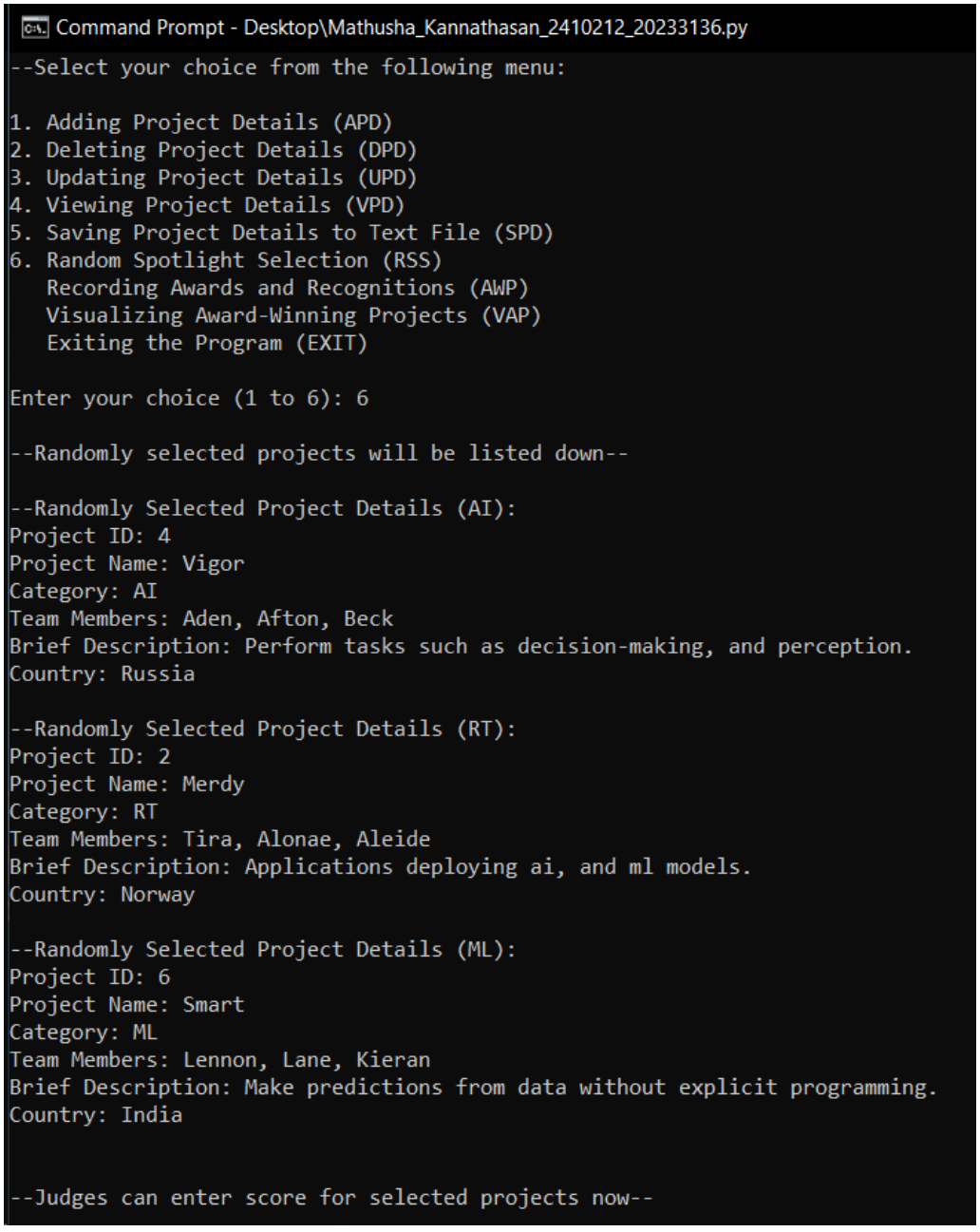


Figure 26 - RSS.output

10) FUNCTION FOR AWARDS WINNING PROJECTS: AWP ( )

I. FLOW_CHART

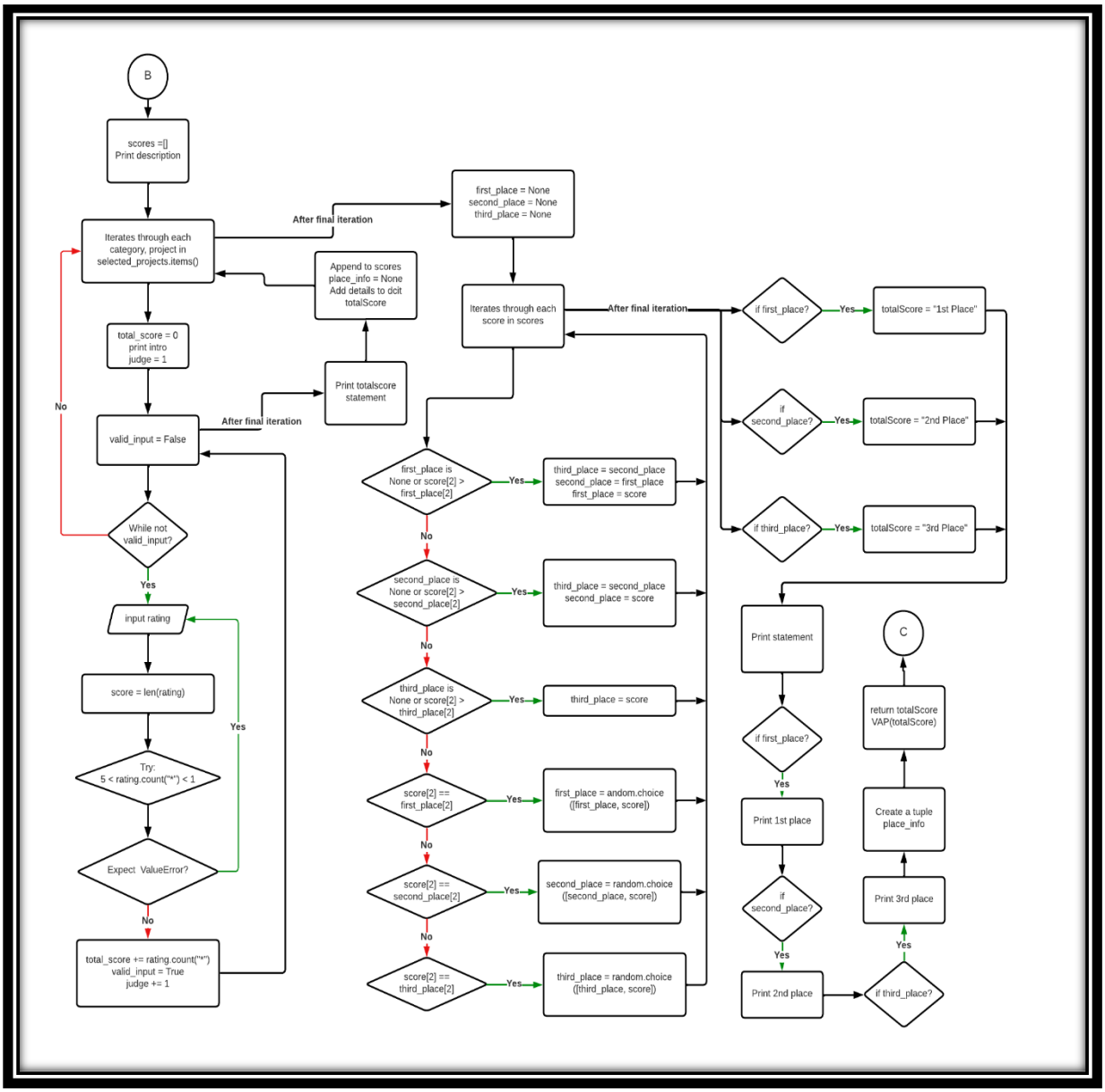


Figure 27 - AWP.flowchart

II. PYTHON_CODE

```
##########################################################################
#Awards Winning Projects(AWP)
#Take points from four judges for each selected project, calculate total scores,
#and determine overall 3 winners for 1st, 2nd, and 3rd place.

def AWP(selected_projects):
    # Initialize scores list
    scores = []
    time.sleep(1)
    print("--Judges can enter score for selected projects now--")
    time.sleep(1)
    # Iterate through selected projects to determine scores
    for category, project in selected_projects.items():
        # Initialize the total score for the current project
        total_score = 0
        print(f"\n---Scoring project of ({category}) with project_ID- {project['Project ID']}:")
        # Take scores from four judges
        for judge in range(1, 5):
            # Initialize flag for valid input
            valid_input = False
            while not valid_input:
                # Prompt the judge for a score using stars
                rating = input(f"Judge {judge}: Enter rating using stars only (1-5):")
                # Validate input and convert stars to a numerical score
                score = len(rating)
                try:
                    if rating.count("*") < 1 or rating.count("*") > 5:
                        raise ValueError
                    total_score += rating.count("*")  #Assign the count of '*' as the score
                    valid_input = True
                except ValueError:
                    print("Invalid input! Please enter between 1 and 5 '*' characters.")
        print(f"Total score for project ({category}) with project_ID- {project['Project ID']} is:
        {total_score}")
        time.sleep(1)
        # Append total score to scores list
        scores.append((category, project['Project ID'], total_score))
        # Store total score in totalScore dictionary along with place_info
        place_info = None  # Initialize place_info
        totalScore[project['Project ID']] = {"Total Score": total_score,
                                             "Place Info": place_info,
                                             "Project Details": project}

    # Initialize variables to store the first, second, and third places
    first_place = None
    second_place = None
    third_place = None
    # Iterate through scores to find the first, second, and third places
```

```
    for score in scores:
        if first_place is None or score[2] > first_place[2]:
            third_place = second_place
            second_place = first_place
            first_place = score
        elif second_place is None or score[2] > second_place[2]:
            third_place = second_place
            second_place = score
        elif third_place is None or score[2] > third_place[2]:
            third_place = score
        # Handle ties
        elif score[2] == first_place[2]:  # Tie with first place
            first_place = random.choice([first_place, score])
        elif score[2] == second_place[2]:  # Tie with second place
            second_place = random.choice([second_place, score])
        elif score[2] == third_place[2]:  # Tie with third place
            third_place = random.choice([third_place, score])
    # Assign places in totalScore dictionary
    if first_place:
        totalScore[first_place[1]]["Place Info"] = "1st Place"
    if second_place:
        totalScore[second_place[1]]["Place Info"] = "2nd Place"
    if third_place:
        totalScore[third_place[1]]["Place Info"] = "3rd Place"
    # Display overall ranking
    print("\n---Overall Ranking:")
    print()
    time.sleep(3)
    if first_place:
        print(f"1st Place: Category-({first_place[0]}) Project_ID-({first_place[1]}) with
{first_place[2]} points")
        time.sleep(2)
        if second_place:
            print(f"2nd Place: Category-({second_place[0]}) Project_ID-({second_place[1]})
with {second_place[2]} points")
            time.sleep(1)
            if third_place:
                print(f"3rd Place: Category-({third_place[0]}) Project_ID-({third_place[1]}) with
{third_place[2]} points")
    # Create a list of place tuples
    place_info = [first_place, second_place, third_place]
    return totalScore
    time.sleep(1)
##############################################################################
```

III. DESCRIPTION

The function AWP is a continuation of RSS. It is used to select 1st three winners of TechExpo by rating using '*' by four judges for each selected project in each category. As I selected 3 projects. They must be the 1st three winners. A judge can vote up to 5 '*' only for a selected project. There a project can get a maximum of 20 stars. Hope this code is lengthy as I did not use any sort of function, I used my algorithm to extract the rank.

Here it initializes a list "scores" to add the scores from judges. Then it will iterate through the project of each category in selected_projects.items(). (.items() method used to return a view object that displays a list of a dictionary's key-value tuple pairs.) It initializes the total score as zero. The for loop starts and it will allow the judge to input the score as '*'. As there are 4 judges, for each category first four judges will vote. If the judge fails to input '*' or the count exceeds the limit of five, ValueError will raise with an error statement. It will allow the user to input the score again. Even if the judge enters an invalid variable with '*', it will omit the invalid character while counting the '*'. After scoring done by each judge, the rating.count ('*') will be added total_score. After the completion of the inner loop, it will print the total_score of that category's project, append the category, project ID, and total_score to scores, initialize place_info as None, and dictionary totalScore to store details of total_score, place_info, and project details. Then it iterates again to another category. After the final iteration of the outer loop, it initializes the first_place, second_place, and third_place as None.

Furthermore, it will iterate each score in scores, and using decision making it will compare each 3 scores with each other and also it will handle the ties using random selection. It will choose the places and then it will assign places in the place_info of the totalScore dictionary. Then it will start to print overall ranking as first_place, second_place, and third_place respectively, and a tuple place_info is also created. At last, it will return the dictionary totalScore for VAP and continue to start the VAP function.

IV. OUTPUT

Here the checking for the count of (‘*’) is implemented, and the tie between both teams is also handled. And then it will lead to VAP ().

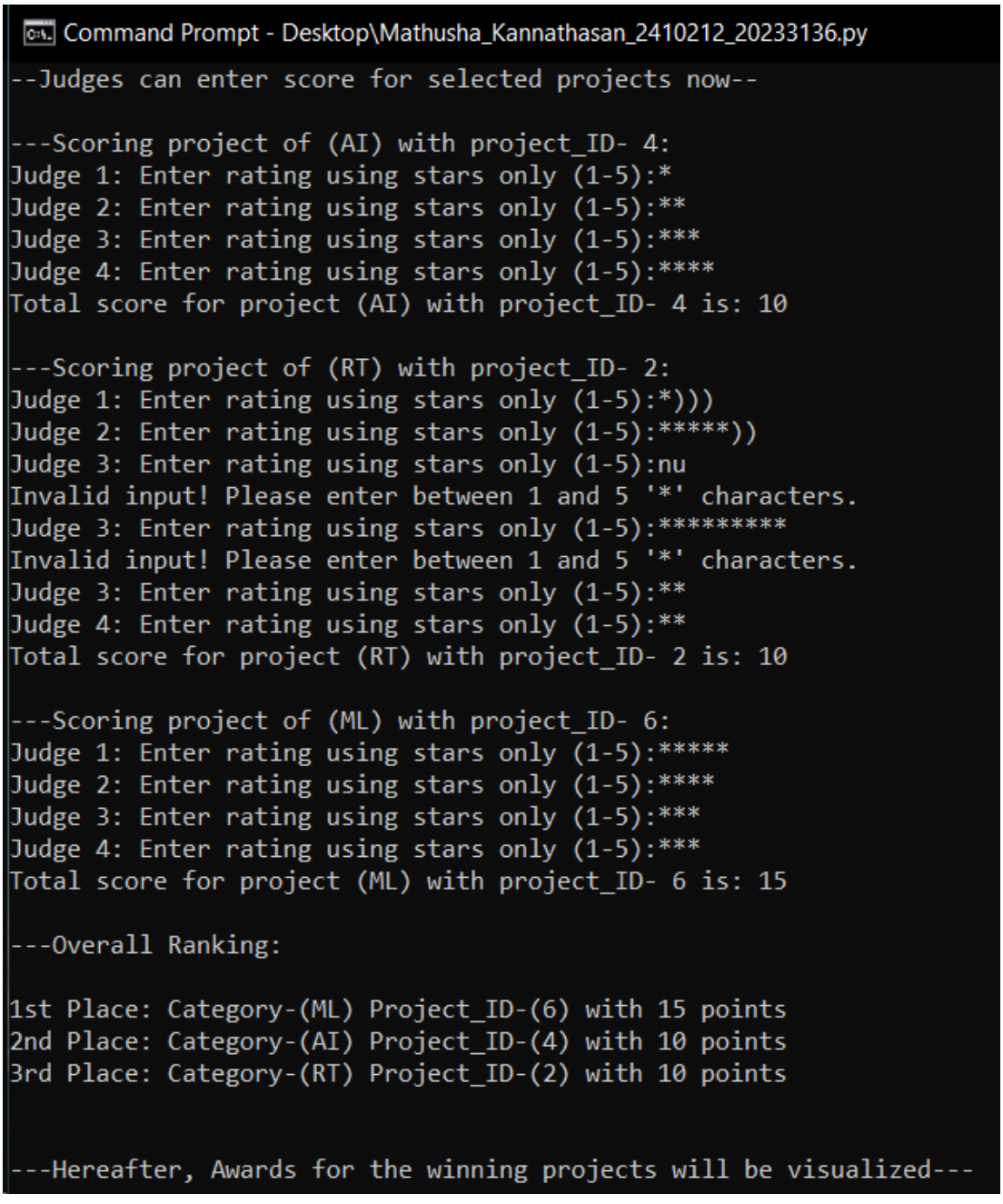


Figure 28 - AWP.output

11) FUNCTION FOR VISUALIZING AWARD-WINNING PROJECTS: VAP (TOTALSCORE)

I. FLOW_CHART

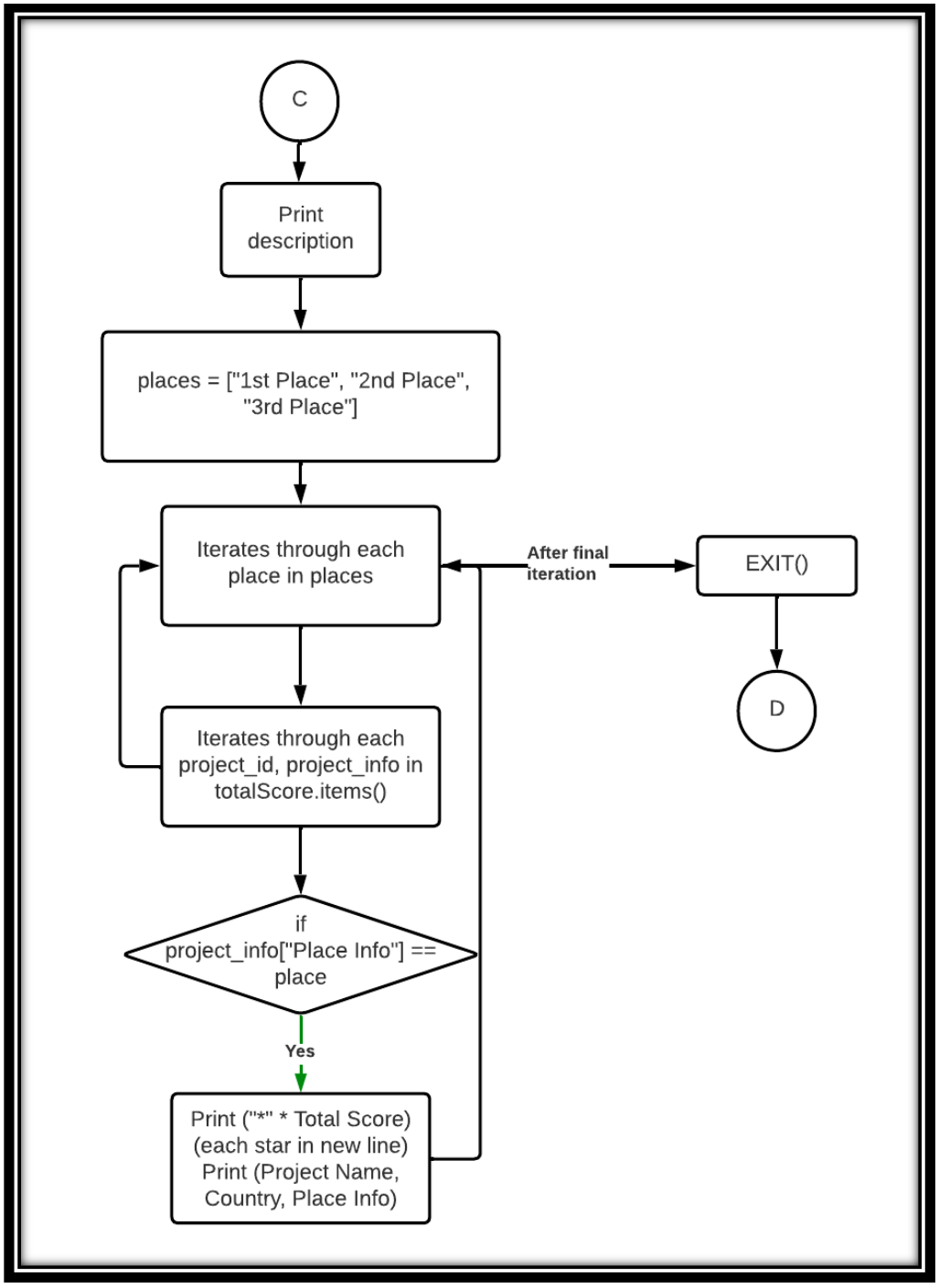


Figure 29 - VAP.flowchart

II. PYTHON_CODE

```
##############################################################################
# Visualizing Award-Winning Projects (VAP)
# Here the awards will be visualized using "*"

def VAP(totalScore):
    print()
    time.sleep(1)
    print("---Hereafter, Awards for the winning projects will be visualized---")
    print()
    time.sleep(2)

    # Define a list to store the place information
    places = ["1st Place", "2nd Place", "3rd Place"]

    # Iterate through places
    for place in places:
        # Iterate through the totalScore dictionary to find projects for the current place
        for project_id, project_info in totalScore.items():
            if project_info["Place Info"] == place:
                print("\n".join(["  *" for _ in range(project_info["Total Score"])]))
                print(project_info["Project Details"]["Project Name"])
                print(project_info["Project Details"]["Country"])
                print(project_info["Place Info"])
                print()
                time.sleep(2)
    time.sleep(1)
##############################################################################
```

III. DESCRIPTION

The function VAP is the continuation of AWP. It is used to visualize the 1st three winners using '*'. Like total_score they gain from the four judges will be awarded as '*'. It starts with a description of visualizing and initializes a list "places". Then it will iterate through each place in places as an outer loop and inside that it will start an inner loop to iterate project_id, and project_info in totalScore.items(), if place info matches then it will print the '*' * total_score (like each star will print in a new line), project name, country, and place info. After completing the inner loop for 1st place. Then it will start to iterate through 2nd place.

IV. OUTPUT

Here the totalScore given by the judges is visualized using '*' and then it will directly lead to EXIT ().

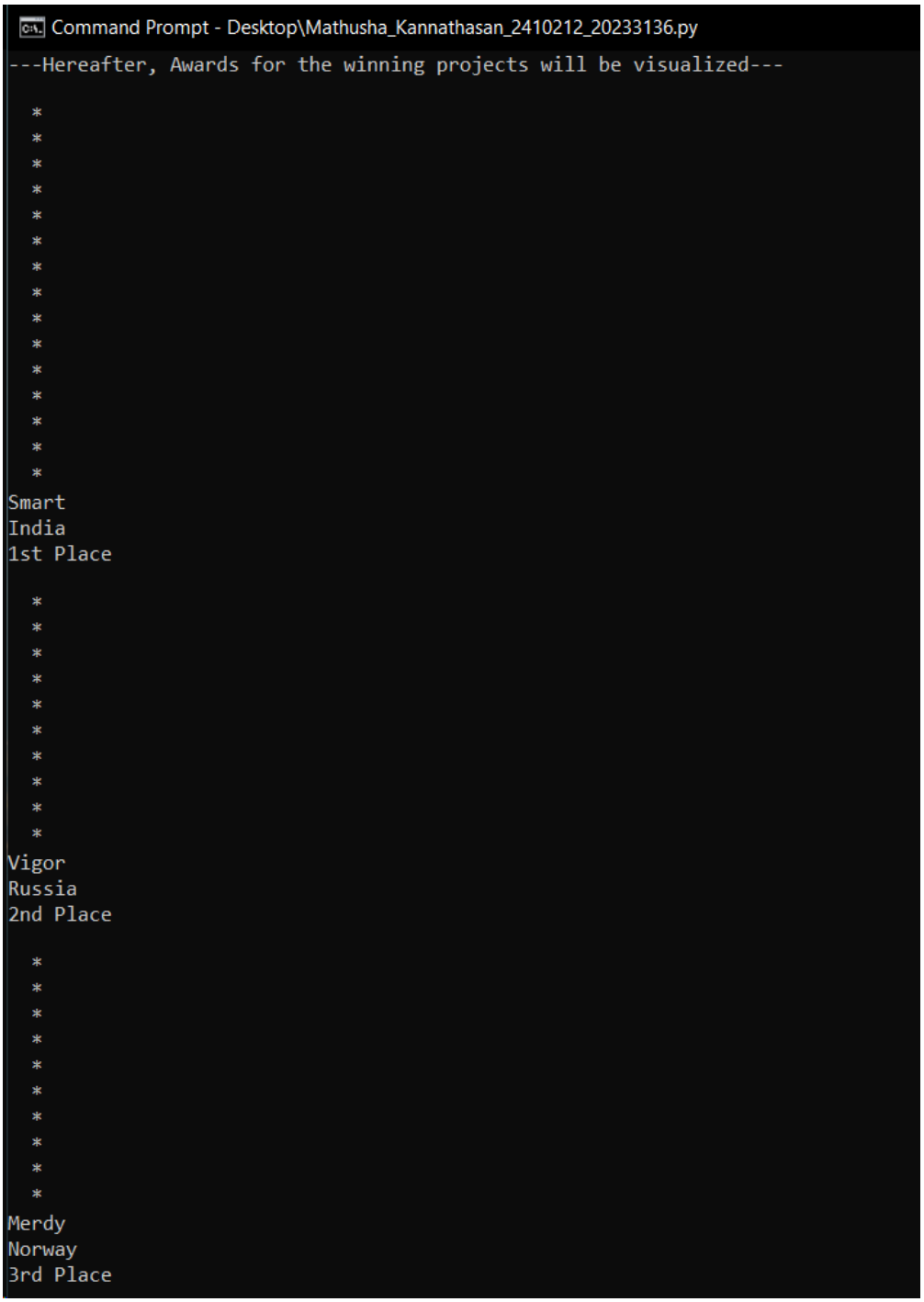


Figure 30 - VAP.output

12) FUNCTION FOR EXITING THE PROGRAM: EXIT ( )

I. FLOW_CHART

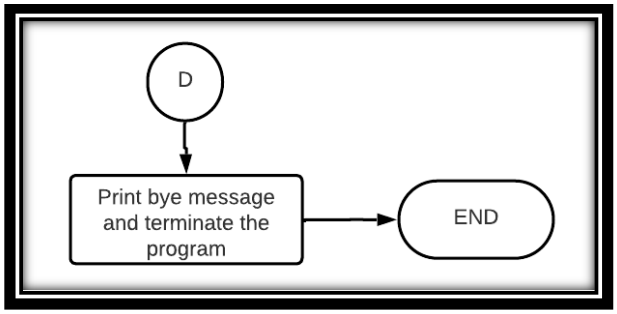


Figure 31 - EXIT.flowchart

II. PYTHON_CODE

```
##########################################################################################
Function to terminate the program(EXIT)

def EXIT():
    time.sleep(1)
    print("--Exiting the program...")
    print("Thank you for participating in TechExpo. Have a great day!")
    exit()  # Exit the program
##########################################################################################
```

III. DESCRIPTION

The function EXIT is the continuation of VAP. It will display the exit message and terminate the program.

IV. OUTPUT

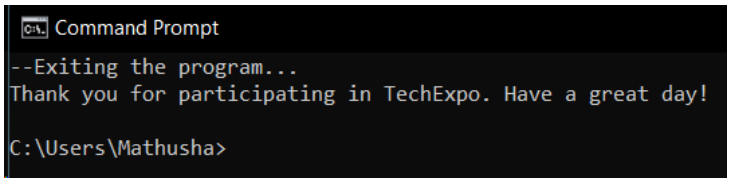


Figure 32 - EXIT.output

13) REFERENCES

Foundation, T. P. (2024, April 15). *The Python Software Foundation*. Retrieved April 2024, from Python_functions: https://docs.python.org/3/library/functions.html

GeeksforGeeks. (2022, Jan 27). *GeeksforGeeks*. Retrieved April 2024, from random.choices() method in Python: https://www.geeksforgeeks.org/random-choices-method-in-python/

GeeksforGeeks. (2024, March 14). *GeeksforGeeks*. Retrieved April 2024, from Python Program for Bubble Sort: https://www.geeksforgeeks.org/python-program-for-bubble-sort/

Gray, D. (2023, Jun 20). *YouTube*. Retrieved from Python Tutorials for Beginners: https://www.youtube.com/playlist?list=PL0Zuz27SZ-6MQri81d012LwP5jvFZ_scc

W3Schools. (2024). *W3Schools*. Retrieved April 2024, from python/python_file_handling: https://www.w3schools.com/python/python_file_handling.asp

W3Schools. (2024). *W3Schools*. Retrieved April 2024, from Python Random choices() Method: https://www.w3schools.com/python/ref_random_choices.asp#:~:text=Definition%20and%20Usage,any%20other%20kind%20of%20sequence.