ROBERT GORDON UNIVERSITY ABERDEEN

INFORMATICS INSTITUTE OF TECHNOLOGY

BSc (Hons) Artificial Intelligence and Data Science

Module: CM1601

Programming Fundamentals

Individual Coursework Report

Module Leader: Ms. Sachinthani Perera

RGU Student ID : 2409099

IIT Student ID : 20232838

Student Name : L.P.V.I. Warnpura

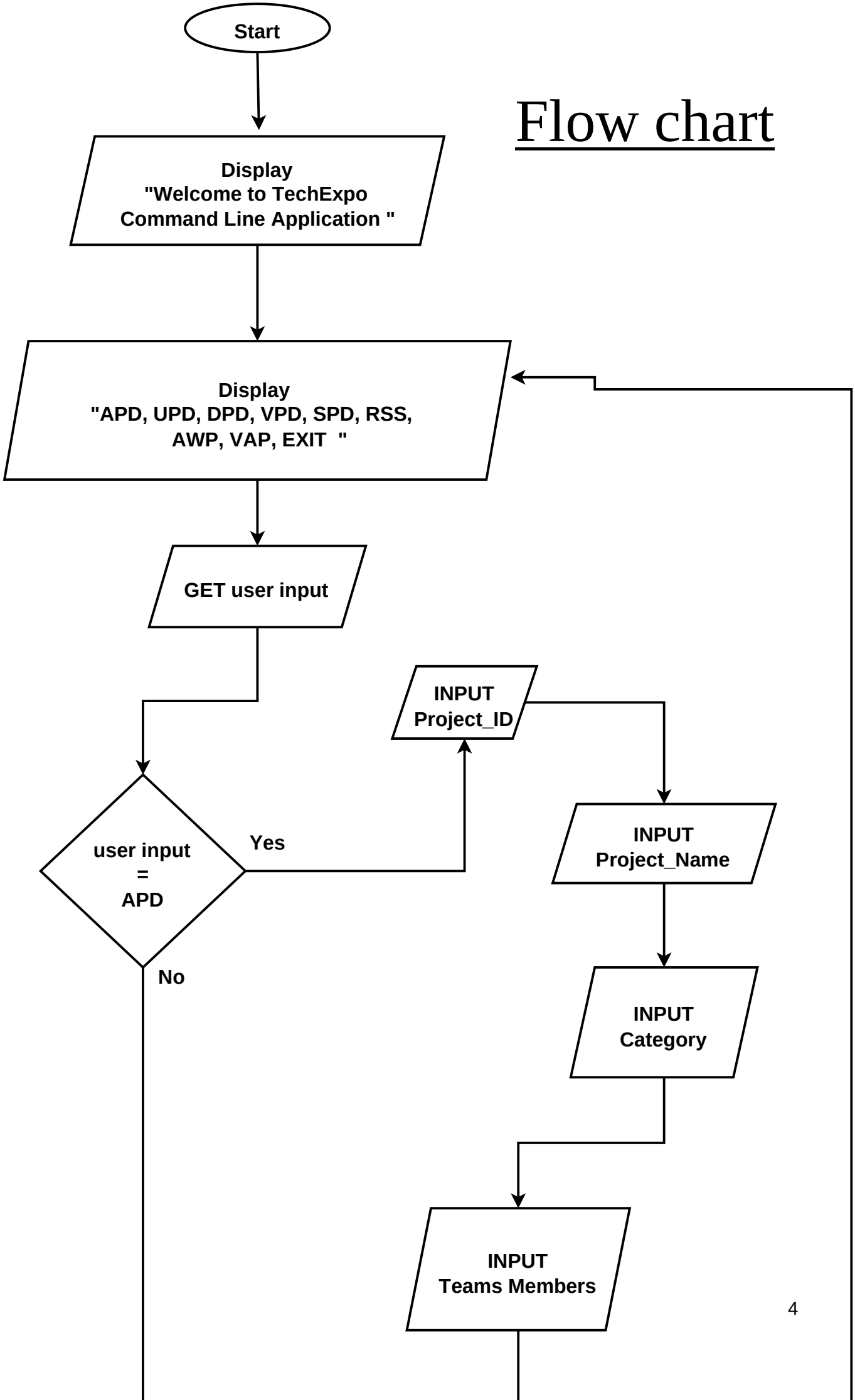
Acknowledgement

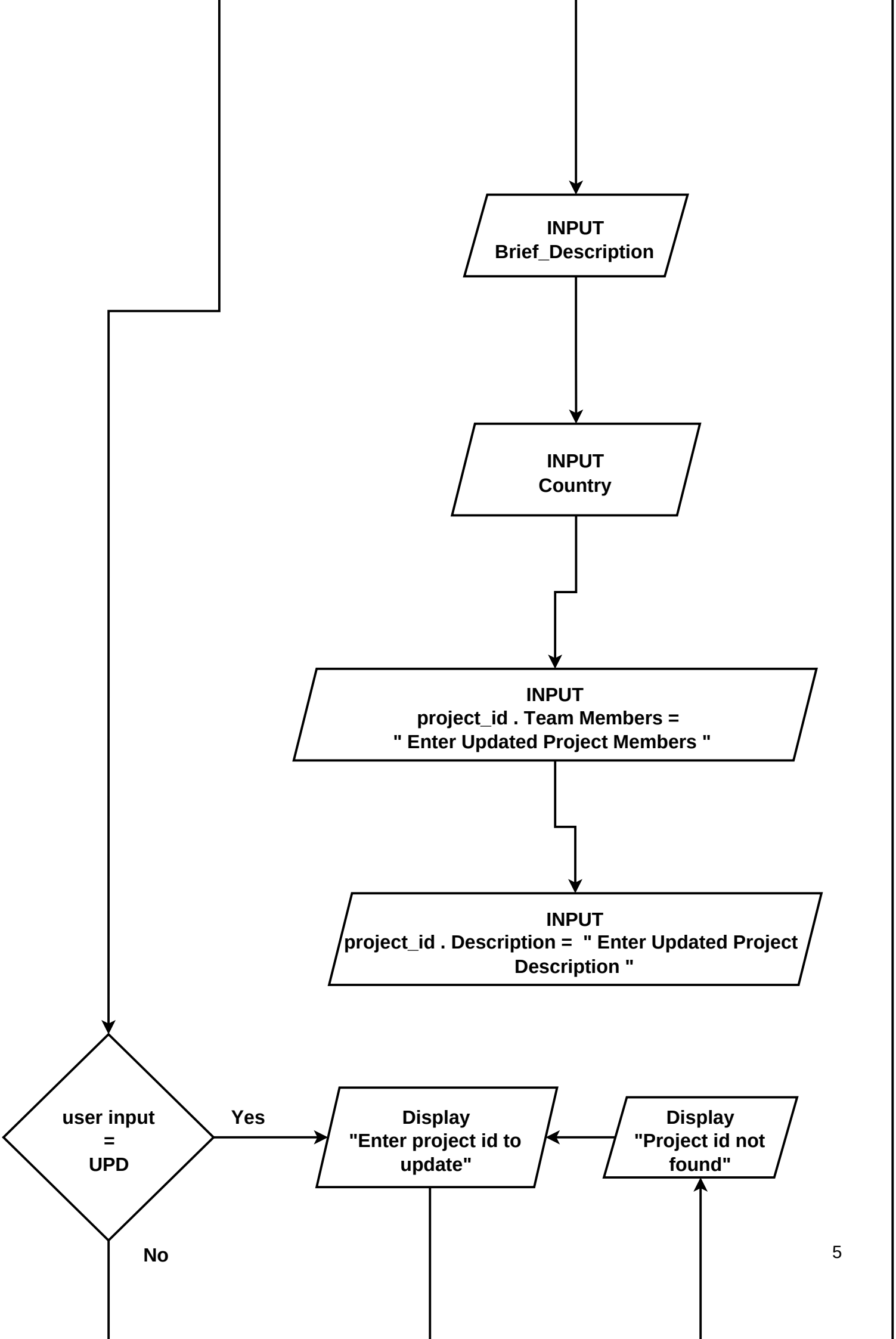
I would like to thank all those who supported me in creating and completing this report. I would also like to spread my heartfelt gratitude to our module leader,
Ms. Sachinthani Perera for the constant guidance and support he gave me throughout this project. Without Ms. Sachinthani Perera and the other respective lecturers, I would not have been able to complete my coursework and report.

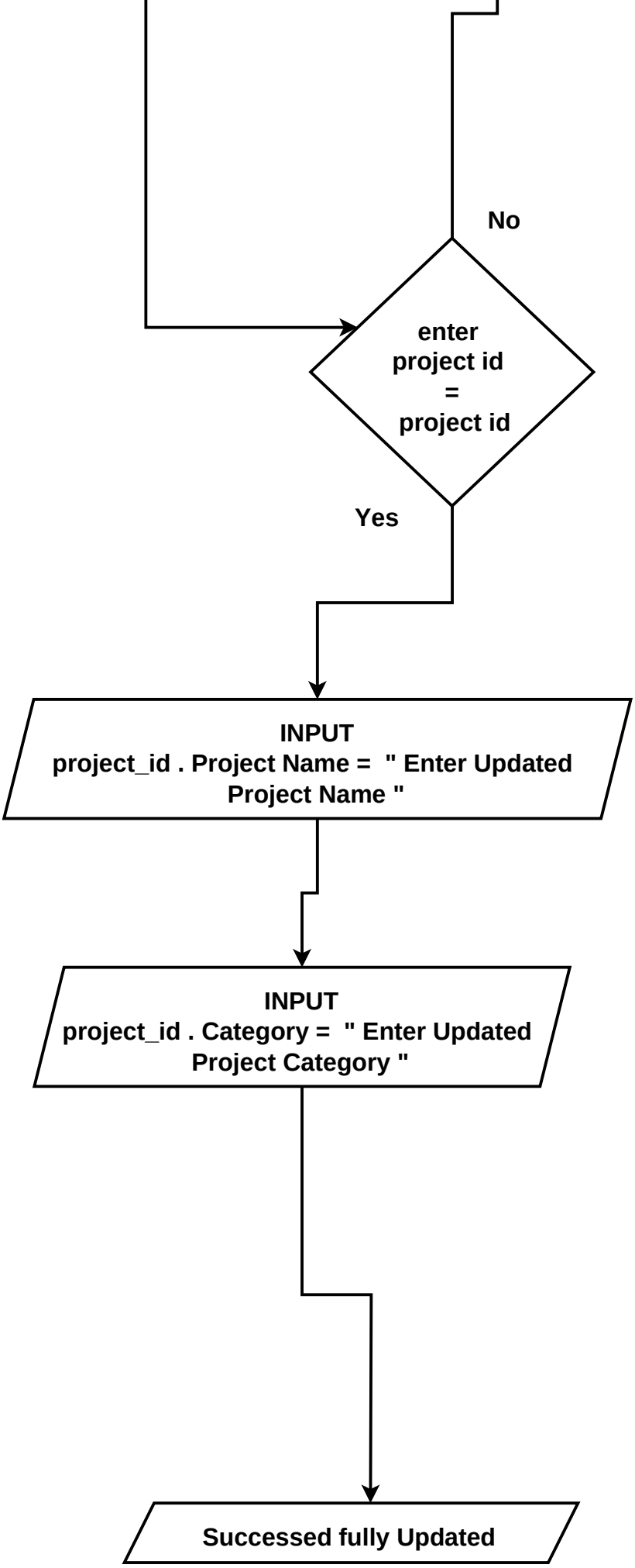
Therefore, I would like to have a form of acknowledgment towards them in this report. I also want to express our gratitude to all the other IIT lecturers who helped us and guided us.
Finally,
I want to thank our parents and friends for helping us to complete this report.

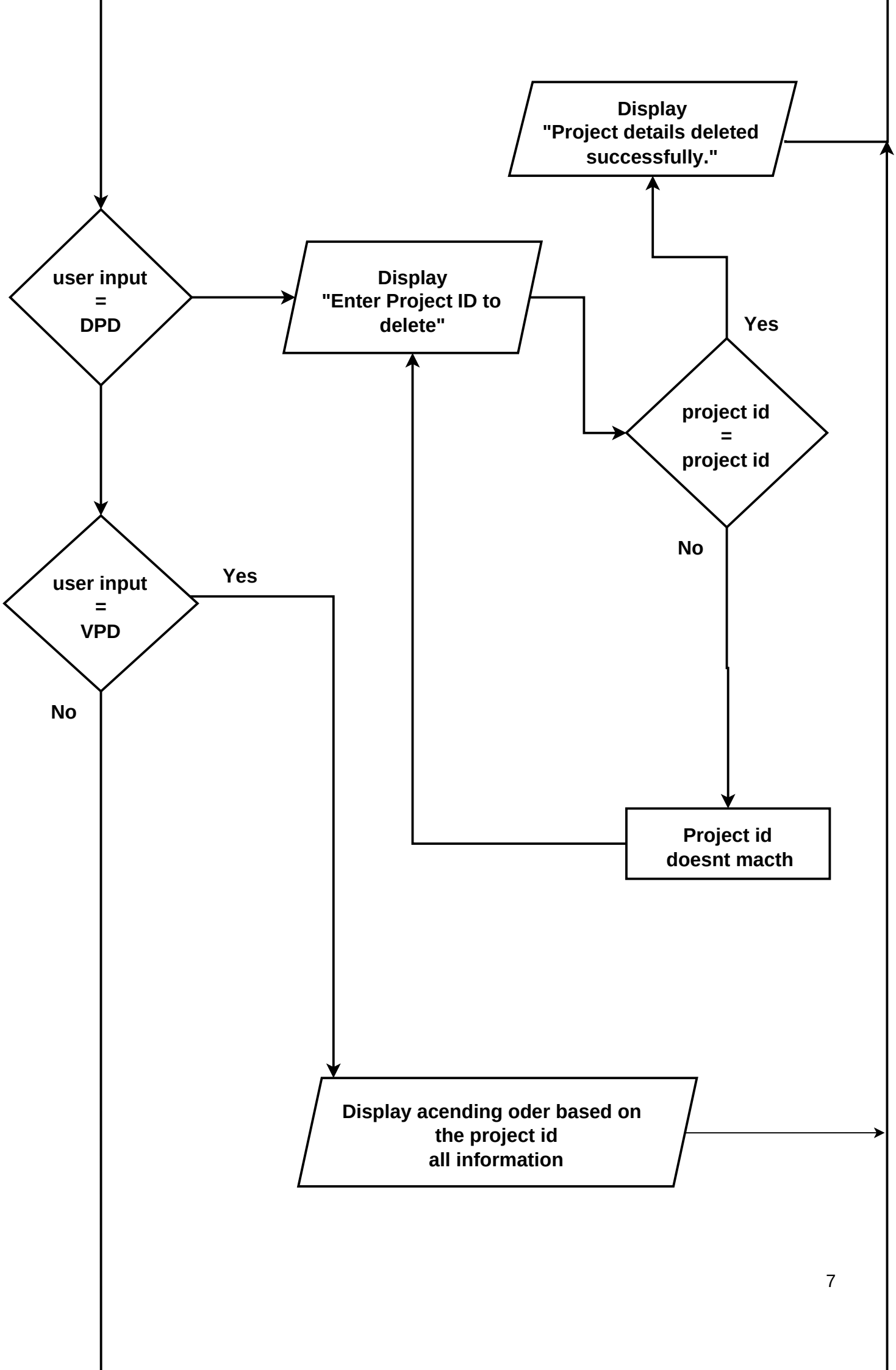
Content table PageNo.

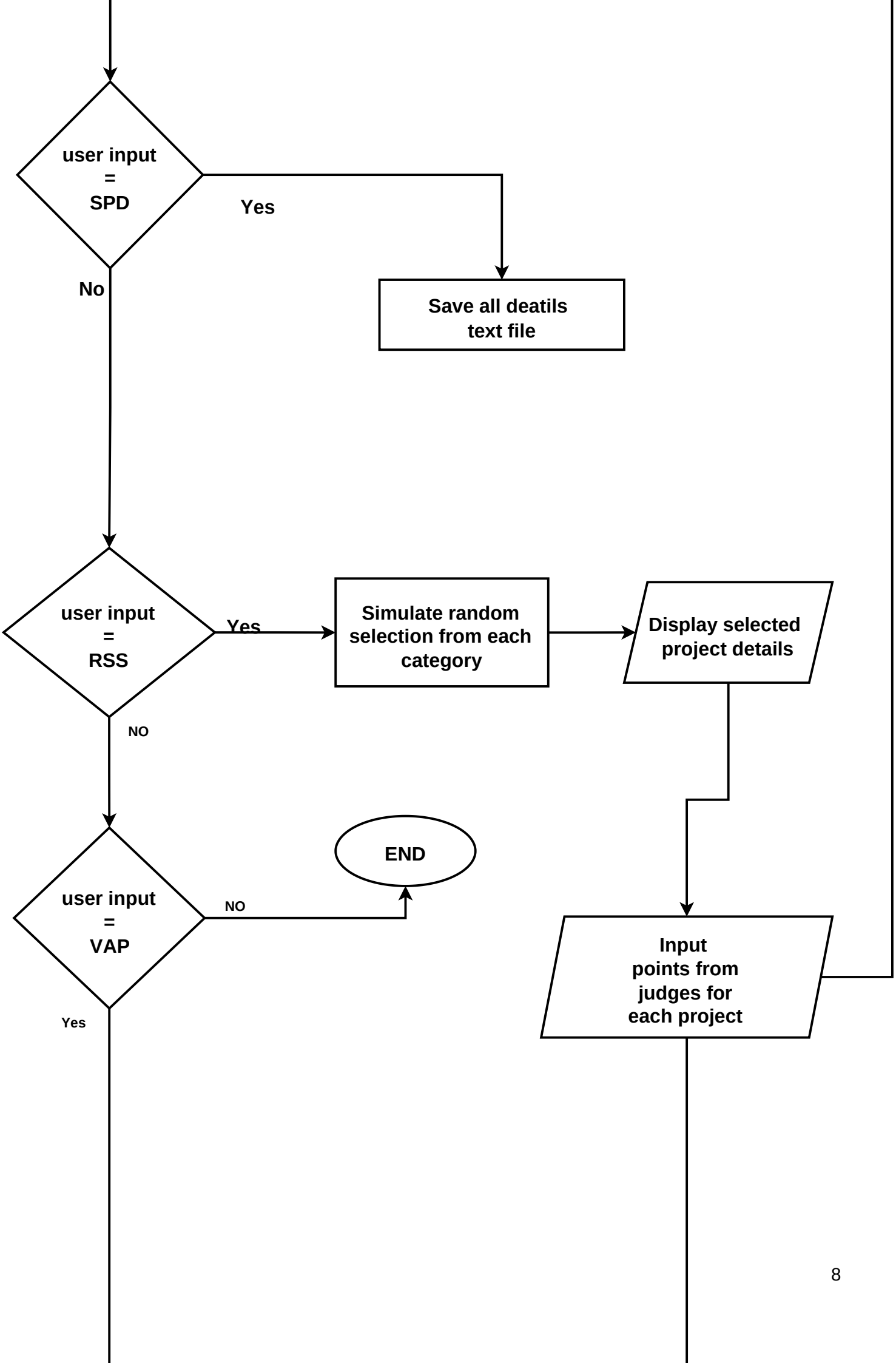
Flow chart

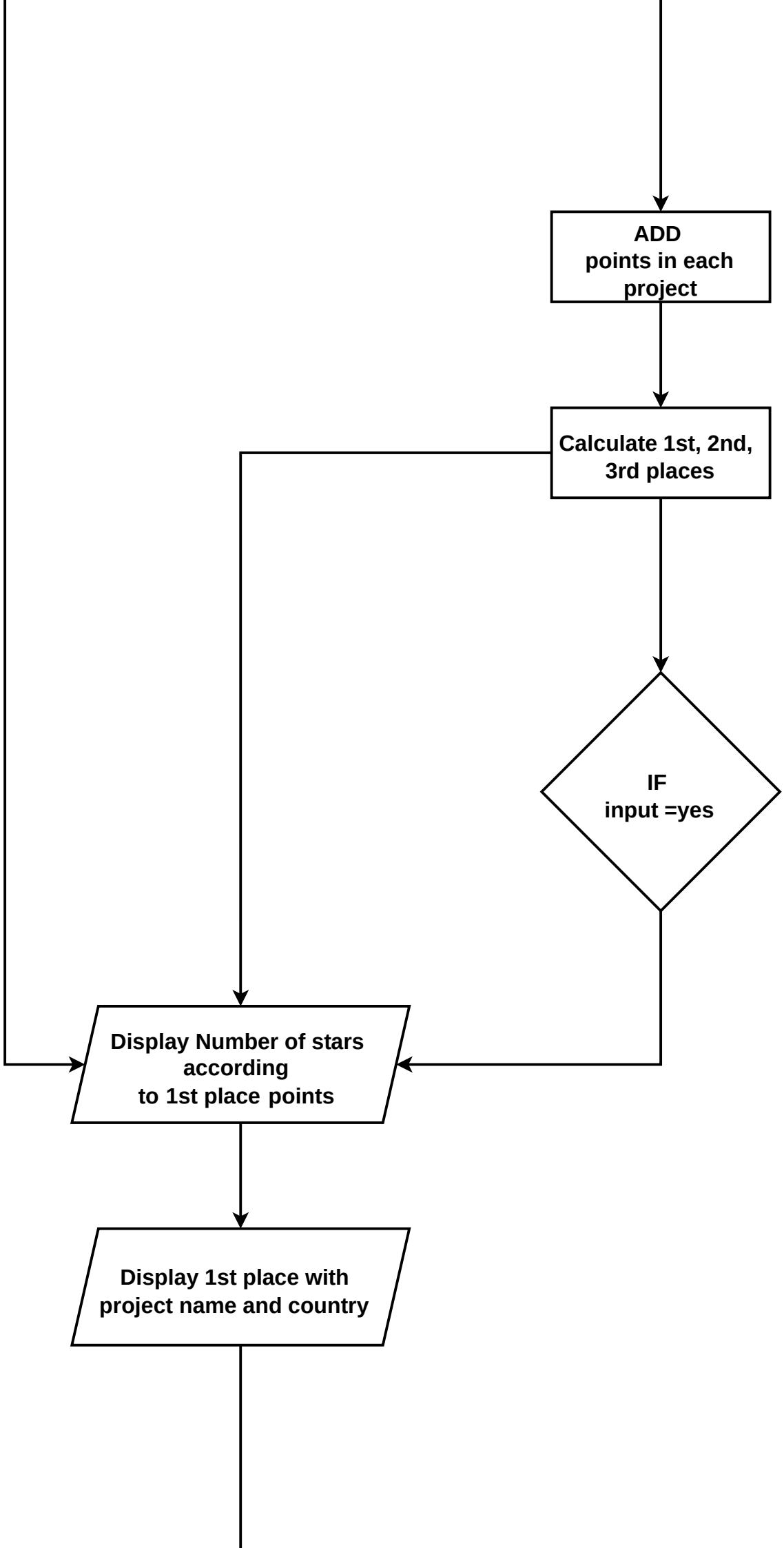

Start
Display
"Welcome to TechExpo
Command Line Application "
Display
"APD, UPD, DPD, VPD, SPD, RSS,
AWP, VAP, EXIT "
GET user input
user input
=
APD
Yes
No
INPUT
Project_ID
INPUT
Project_Name
INPUT
Category
INPUT
Teams Members

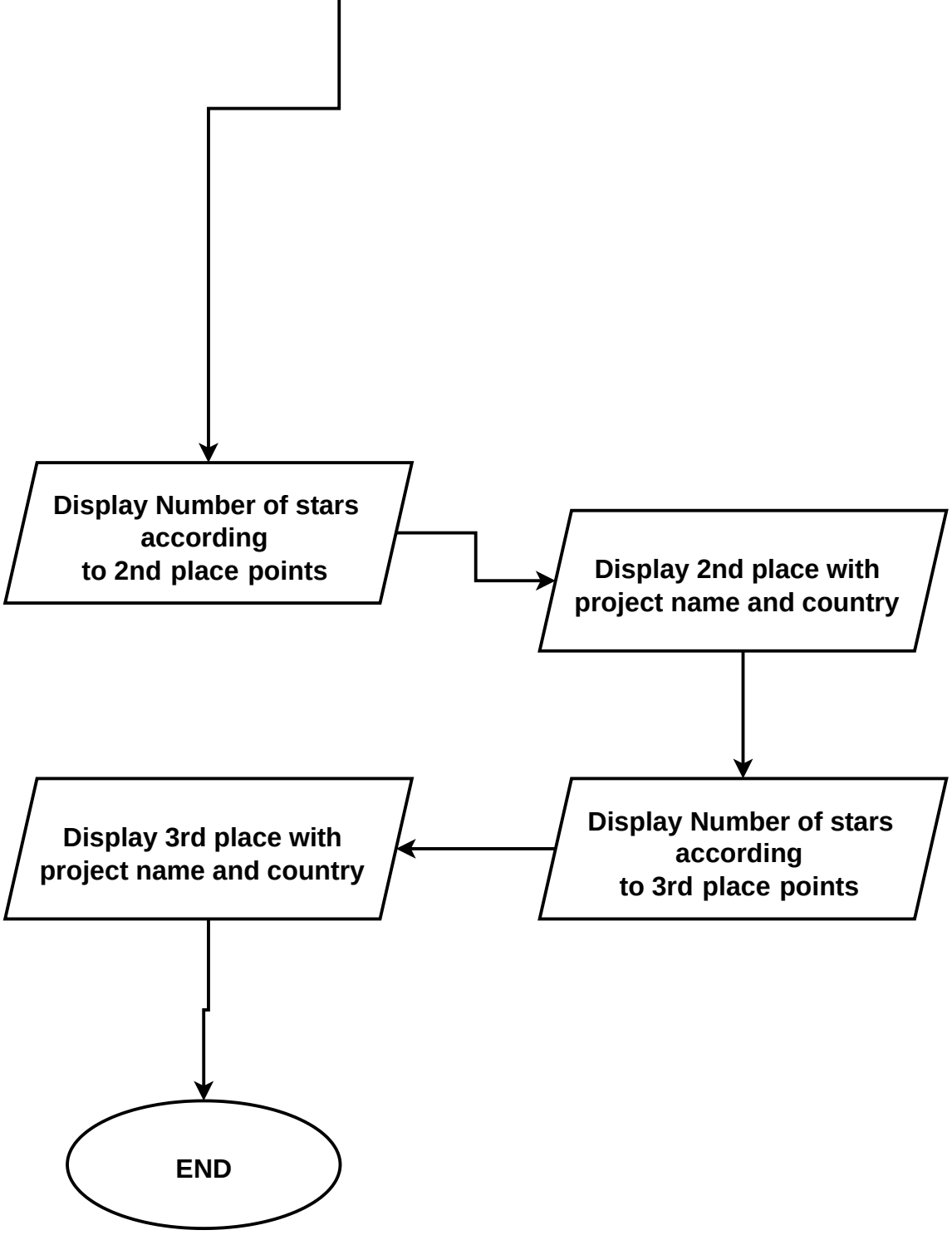

INPUT
Brief_Description
INPUT
Country
INPUT
project_id . Team Members =
" Enter Updated Project Members "
INPUT
project_id . Description = " Enter Updated Project Description "
user input
=
UPD
Yes
No
Display
"Enter project id to update"
Display
"Project id not found"


No
enter
project id
=
project id
Yes
INPUT
project_id . Project Name = " Enter Updated
Project Name "
INPUT
project_id . Category = " Enter Updated
Project Category "
Successed fully Updated


Display
"Project details deleted
successfully."
user input
=
DPD
Display
"Enter Project ID to
delete"
Yes
project id
=
project id
No
user input
=
VPD
Yes
No
Project id
doesnt macth
Display acending oder based on
the project id
all information


user input
=
SPD
Yes
No
Save all deatils
text file
user input
=
RSS
Yes
Simulate random
selection from each
category
Display selected
project details
NO
END
user input
=
VAP
NO
Yes
Input
points from
judges for
each project


ADD
points in each
project
Calculate 1st, 2nd,
3rd places
IF
input =yes
Display Number of stars
according
to 1st place points
Display 1st place with
project name and country


Display Number of stars according to 2nd place points
Display 2nd place with project name and country
Display Number of stars according to 3rd place points
Display 3rd place with project name and country
END

function ()

- Saved location

```
# Define global variables
projects = {}  # Dictionary to store project details
project_details_saved = False  # Variable to track if project details have been saved
```

- projects: This that is a dictionary to which the project details have to be stored is a variable. The magic of Python is that dictionaries are key-value pairs, which let you to assign a value with a unique key. It appears that the goal here is to name the title key and keep data as the values such as project description, deadlines, team members in tabular format.

- project_details_saved: This flag is a variable of boolean class used for ascertaining the saving of project details. The is given as an initial value as this means it is not sure (yet) that some details have been saved. This flag is crucial in determining whether the users need to keep changes they have made to the project details saved or if they should be prompted to keep them before they exit the program or the session.

- display menu functional

```
# Function to display menu
def display_menu():
    print("1. Adding Project Details (APD)   " )
    print()
    print("2. Updating Project Details (UPD)   " )
    print()
    print("3. Deleting Project Details (DPD)   ")
    print()
    print("4. Viewing Project Details (VPD)   ")
    print()
    print("5. Saving Project Details to Text File (SPD)   ")
    print()
    print("6. Random Spotlight Selection & Award given (RSS and AWP)   ")
    print()
    print("7. Visualizing Award-Winning Projects (VAP)   ")
    print()
    print("8. Exiting the Program (EXIT)   ")
    print()
```

- This function is -> display_menu() in other words - where users can see a menu of options available to them. Students can select the corresponding number and get a short description for each of the variants. Let's break down each option:Let's break down each option:

- Adding Project Details (APD): Here, the user probably gives an option wherein the user can add new projects as key to projects dictionary (please note: the keys can be strings only).

- Updating Project Details (UPD): Such adjustments may give opportunity to users to edit existing information, e.g. they can be able to change

deadline or team members.

- Deleting Project Details (DPD): Possibly, this checkbox helps to delete the project from a projects dictionary.

- Viewing Project Details (VPD): This procedure will spew out all details about the project and other data that has been saved in the projects word list.

- Saving Project Details to Text File (SPD): This choice may make the data with arguments for tasks into a text document and save this text file on the disk replacing the dictionary.

- Random Spotlight Selection & Award given (RSS and AWP): This possibility appears to be, randomly bringing the projects out from the projects dictionary and rewarding it with recognizing award or any prize.

- Visualizing Award-Winning Projects (VAP): This choice could be achieved by showing or displaying data on the basis of the award winning projects in the dictionary project which is already stored.

- Exiting the Program (EXIT): Thanks to this mode the program gets a chance to be ended.

- Briefly, this menu ensures a useful way for users to operate with project information and take crucial management steps concerning projects.

- Add project function

```
# Function to add project details
def add_project():
    global project_details_saved
    if project_details_saved:
        print("Project details have already been saved. You cannot add new projects.")
        print()
        return

    # Input project details
    while True:
        project_id = input("Enter Project ID (numeric only & must be a 3-digit number): ")
        if project_id.isdigit():  # Check if input consists only of digits
            project_id = int(project_id)  # Convert to integer
            if 0 < project_id <= 999:
                break
            else:
                print("Project ID must be a 3-digit number.")
                print()
        else:
            print("Invalid input. Please enter a numeric value.")
            print()

    project_name = input("Enter Project Name: ")
    category = input("Enter Category: ")
    team_members = input("Enter Team Members (separated by comma): ").split(",")
    description = input("Enter Brief Description: ")
    country = input("Enter Country: ")

    # Store project details
    # Convert project IDs to strings when adding projects
    projects[str(project_id)] = {
        "Project Name": project_name,
        "Category": category,
        "Team Members": team_members,
        "Description": description,
        "Country": country
        }
```

- This add_project() function does not just assist users to provide the necessary information to declare a new project but also places it in the projects dictionary. undefined

- It sets True for the project_details_saved flag. If so, narrative which says that the data has been already saved and saving the new information will be done is provided.

- Project ID: It is an assurance that the number input is a 3-digit number.

- Project Name, Category, Team Members, Description, and Country: It requests for the entry of such variables.

- Following the complete collecting of project details, it stores them using the project ID as a key and a dictionary of project details as the value into the projects dictionary.

- This feature allows users to reorder the projects depending on their priorities and it provides a roadmap for users to follow.

Add sample data

```
# Function to add sample data
def add_sample_data():
    global projects
    sample_data = [
        {
            "Project ID": "001",
            "Project Name": "Social Media App",
            "Category": "Software Development",
            "Team Members": ["John Doe", "Jane Smith", "Alice Johnson"],
            "Description": "Developing a social media application for sharing photos and
messages.",
            "Country": "USA"
        },
            {
            "Project ID": "002",
            "Project Name": "Social Media App",
            "Category": "Software Development",
            "Team Members": ["John Doe", "Jane Smith", "Alice Johnson"],
            "Description": "Developing a social media application for sharing photos and
messages.",
            "Country": "USA"
        },
        {
            "Project ID": "003",
            "Project Name": "Data Science - www ",
            "Category": "Artificial Intelligence",
            "Team Members": ["David Brown", "Emily Wilson", "Michael Lee"],
            "Description": "Building an online platform for buying and selling various products.",
            "Country": "Canada"
        },
        {
            "Project ID": "004",
            "Project Name": "Software Development - AAA",
            "Category": "Software Development",
            "Team Members": ["John Doe", "Jane Smith", "Alice Johnson"],
            "Description": "Developing a social media application for sharing photos and
messages.",
            "Country": "USA"
        },
        {
            "Project ID": "005",
            "Project Name": "E-commerce Webs",
            "Category": "Data Science",
            "Team Members": ["David Brown", "Emily Wilson", "Michael Lee"],
            "Description": "Building an online platform for buying and selling various products.",
            "Country": "Canada"
        },
        {
            "Project ID": "006",
            "Project Name": "Software Development",
            "Category": "Artificial Intelligence",
            "Team Members": ["Sophia Martinez", "Daniel Taylor", "Olivia Clark"],
            "Description": "Creating an addictive mobile game with challenging levels.",
            "Country": "UK"
        },
```

```
{
        "Project ID": "007",
        "Project Name": "AI Chatbot",
        "Category": "Artificial Intelligence",
        "Team Members": ["Liam Anderson", "Ella White", "James Wilson"],
        "Description": "Designing an intelligent chatbot capable of answering user queries.",
        "Country": "Australia"
    },
    {
        "Project ID": "008",
        "Project Name": "Robotics Project",
        "Category": "Data Science",
        "Team Members": ["Ava Garcia", "Noah Brown", "Sophie Moore"],
        "Description": "Building a robot capable of performing various tasks autonomously.",
        "Country": "Germany"
    },
    {
        "Project ID": "009",
        "Project Name": "Data Analysis Tool",
        "Category": "Data Science",
        "Team Members": ["Mia Johnson", "William Taylor", "Ethan Martinez"],
        "Description": "Developing a tool for analyzing large datasets and generating insights.",
        "Country": "France"
    },
    {
        "Project ID": "010",
        "Project Name": "Software Development",
        "Category": "Artificial Intelligence",
        "Team Members": ["Isabella Wilson", "Mason Garcia", "Abigail Brown"],
        "Description": "Researching and implementing renewable energy solutions.",
        "Country": "Brazil"
    },
    {
        "Project ID": "011",
        "Project Name": "Healthcare App",
        "Category": "Artificial Intelligence",
        "Team Members": ["Evelyn Clark", "Logan Anderson", "Chloe Moore"],
        "Description": "Developing a mobile application for monitoring health metrics.",
        "Country": "Japan"
    }

    # Add more sample data entries here
]
```

Call sample data function

```
    # Add sample data to projects dictionary
    for data in sample_data:
        projects[data["Project ID"]] = {
            "Project Name": data["Project Name"],
            "Category": data["Category"],
            "Team Members": data["Team Members"],
            "Description": data["Description"],
            "Country": data["Country"]
        }

# Call the function to add sample data
add_sample_data()
```

- putting the sample data to the projects dictionary by confining the for loop over the sample_data list which allows for each project's details to be assigned to the projects dictionary in the form of a project ID key.

- To be more specific, the add_sample_data() function you've called allows you to add the sample project data that you've just defined. The process is now complete, and the project dictionary new version contains different projects with their details.

Update project function

```
# Function to update project details

def update_project():
    global project_details_saved
    if project_details_saved:
        print("Project details have already been saved. You cannot update project details.")
        return
    project_id = input("Enter Project ID to update: ")

    if project_id in projects:
        #Update project details
        projects[project_id]["Project Name"] = input("Enter Updated Project Name: ")
        projects[project_id]["Category"] = input("Enter Updated Category: ")
        projects[project_id]["Team Members"] = input("Enter Updated Team Members (separated by
comma): ").split(",")
        projects[project_id]["Description"] = input("Enter Updated Brief Description: ")
        projects[project_id]["Country"] = input("Enter Updated Country: ")
    else:
        print("Project ID not found.")
```

- This update_project() functions enables users to change the row for the specific project assignment stored in the projects dictionaryHere's how it works:

- It if evaluates if the project_details_saved flag is set to True. If so it shows that the headline already has been saved with the data and checks not repeat the work of the saving of data.

- It illicits the user to type inidividual project ID of the project you wish to renew.

- If the current project ID that is entered matches with one in the 'projects' dictionary, it will prompt the user to input the updated project details, such as project name, category, team members, description, and country.

- It revisits the project dictionary and adds the project details from the user in the dictionary.

- In this case, if the entered project ID is not present in the projects dictionary, it prints the message that desired project ID not found.

- In this context the feature gives users the opportunity to modify some project variables.

Delete project function

```
# Function to delete project details
def delete_project():
    global project_details_saved
    if project_details_saved:
        print("Project details have already been saved. You cannot delete project details.")
        return
    project_id = input("Enter Project ID to delete: ")
    if project_id in projects:
        del projects[project_id]
        print("Project details deleted successfully.")
    else:
        print("Project ID not found.")
```

- This function delete_project, lets users delete a certain project and all details related to it. The details are stored in a dictionary, called projects. Here's how it works:

- It tests if the project_details_saved flag is None/False. Yes, it flows whether project details are saved or not and prevents the user from loading it if project details are saved already.

- It mates the user to introduce project ID into which project he/she wants to delete.

- If the mandatory project ID matches accordingly the project heading of the projects dictionary and the project details associated with that project ID is deleted with the help of del keyword.

- If the played in project ID is not placed in the dictionary as projects, the system notifies the user that the project ID was not located.

- With feature, users have a safe house for them to remove the un-needed project details from the kit.

View project details Function

```
# Function to view project details
def view_projects():
    global projects, project_details_saved

    if not projects:  # Check if there are no projects saved
        print("No projects have been saved yet")
        print()
        return

    if project_details_saved:  # Check if project details have not been saved
        print("Project details have already been saved. You cannot view project details.")
        print()
        return

    # Display project details sorted by Project ID
    for project_id in sorted(projects.keys()):
        print("Project ID:", project_id)  # No need for int() conversion if project_id should be a string
        for key, value in projects[project_id].items():
            print(key + ":", value)
        print()
```

- This method_view_projects() creates to display the details of all projects in projects dictionary. Here's a breakdown of how it works:Here's a breakdown of how it works:

- Sophisticated as it may seem, it turns out that all it has to do is to inquire of the projects dictionary if there are any projects saved. Consequently, if you use the prompt it will print a statement that no projects were saved yet.

- It checks if the flag project_details_saved has the value True on If so, the next click will lead the user to the page where she can see what projects she has already saved on the trip details.

- If neither condition is met or does not satisfy any of the actions, it then displays the project details sorted beneath the project ID. It works its way approach the value of the projects pAirE "dict" that is represented by ID of our project and then it writes down the data of each project.

- Projects details are given flowing in a well structured way with one new line for each item such as the project ID, project name, category, members, description and country.

- Then color printing of the various projects names is done and to mark the difference and the completion of one project a blank line is printed.

- This function provides a convenient way for the users of the program to see all the details of the projects which is stored in the projects dictionary.Thus, assuming the storage of all projects details in the projects dictionary, the view_projects() function accomplishes to show the details of all projects. Here's a breakdown of how it works:Here's a breakdown of how it works:

Save project details as a text file function

```
# Function to save project details to text file
def save_to_text_file():
    global project_details_saved
    try:
        if project_details_saved:
            print("Project details have already been saved.")
            print()
            return
    except IOError as err:
        print("Error saving project details:", err)



    # Write project details to a text file
    with open("project_details.txt", "w") as file:
        for project_id, details in projects.items():
            file.write(f"Project ID: {project_id}\n")
            for key, value in details.items():
                file.write(f"{key}: {value}\n")
            file.write("\n")
```

- A function, save_to_text_file() is provided to save the dictionaries which present a project's information found in projects dictionary inside a text file "project_details.txt". Let's go through how it works:Let's go through how it works:

- It secondly knows if the variable is project_details_saved not. And the last one, it prints that project details already have been saved and quit processing of this function.

- With the procedure part inside of the try block, it tries to write the project details to the text file. An error in the try block will pop up during this process which will then catch the except block and an error message will be displayed.

- It goes repeatedly over the projects dictionary items writing down the contents of different projects to an opened text file. Each of these projects begins with its unique ID tag, followed by every detail (key-value pair) displayed in as a new line.

- The process goes on writing, each project details will be saved in the last line of a paragraph before the paragraph shifts to another project details

- Finally, the set project_details_saved to True will be a confession that the project details have been saved successfully.

- Such tool allows you to not only store project information, but also have it available at any moment of time for revising.

Random spotlight section

```
# Function to simulate random spotlight selection
def random_spotlight_selection():
    # Initialize a dictionary to store randomly selected projects for each category
    selected_projects = {}
    categories = set([project['Category'] for project in projects.values()])

    # Iterate through each category
    for category in categories:
        # Get projects belonging to the current category
        category_projects = [project_id for project_id, project in projects.items() if project["Category"] ==
category]

        # Check if there are projects in the current category
        print(f"Category: {category}")
        for project_id in category_projects:
            project_details = projects[project_id]
            print()
            print()
            print("Selected Project Details:")
            for key, value in project_details.items():
                print(key + ":", value)
            print("Project id is : " + project_id)
            print()
            give_judge_pointss(project_id, project_details)  # call * marks funtion
            #give_judge_points(project_id, project_details)    # call numeric funtion
    yes_or_no = input("If you wish final marks, NOW it's ready. Do you want to continue (yes/no): ")
    if yes_or_no == 'yes':
        visualize_projects()
    else:
        print()
```

- It does this using the given random_spotlight_selection() function that is similar to a project selection process based on randomness.Here's how it works:

- It sets up a dictionary called selected_projects where by some projects from different categories are randomly selected.

- It builds up a specific set of borders by figuring out the category of the project in the dictionary of projects.

- It runs over each category, chooses projects randomly, and then displays one among that category every time it is displayed.

- What it does is that it prints the information about the projects and then it calls a function named the give_judge_points() that assigns the marks or points to the project. However, it seems like there are two function calls commented out, indicating two different approaches for assigning points or marks to projects: one with an asterisk symbol, while the other uses a numerical figure.

- Then the results are displayed specifying how many marks each was allocated and gives the option of seeing the total marks. When the user goes on with it, the function visualize_projects() becomes active.

- Basically, this service gives the possibility to run random selection from different sections, their details as well as assigning scores or marks.

Marks give function ****

```
# Give judge  using ************
def give_judge_pointss(project_id, project_details):
    points = []
    for i in range(4):
        while True:
            judge_points = input(f"Judge {i+1} points (out of 5) for project {project_details['Project
Name']}: ")
            if judge_points in ['*', '**', '***', '****', '*****']:
                break
            else:
                print("Invalid input! Please enter '*' to '*****'.")
        points.append(len(judge_points))

    # Calculate total points for the current project
    total_points = sum(points)

    # Update project details with total points
    projects[project_id]["Total Points"] = total_points
```

- write a function give_judge_points() statement hints the judges to use asterisks to give points from ('* to '*****) Here's a breakdown of how it works:Here's a breakdown of how it works:

- It allows the judge to interact with the project by providing it with their points using the asterisk as a symbol of scores from 1 to 5.

- It verify whether the judge is telling a fact (i.e., '*' to '*****). However, it asks the judge to clarify invalid arguments. Conclusively it requires the judge to input the valid arguments.

- It totals up the overall points for the project depending on the total size of strings of asterisks entered by each evaluator.
- And it loads the project details from the projects dictionary and also updates the total points assigned by the judges.

- This function is a quick way for the judges to assign the points to the project by using an asterisk to count the points distribution dots.

Marks given like numeric function

```
# Give judge points in numerical
def give_judge_points(project_id, project_details):
    points = []
    for i in range(4):
        while True:
            try:
                judge_points = int(input(f"Judge {i+1} points (out of 5) for project
{project_details['Project Name']}: "))
                if 1 <= judge_points <= 5:
                    break
                else:
                    print("Invalid input! Please enter a number between 1 and 5.")
            except ValueError:
                print("Invalid input! Please enter a valid integer.")
        points.append(judge_points)
        print()
```

- This give_judge_points() function enables judges to assign a maximum of 5 points to a project by choosing numbers from the range 1 to 5.Here's how it works:

- It directs each one of the judges to give points for the project featuring numbers.

- It verifies whether the judge input is valid or not (i.e, an integer between 1 and 5.). In case the answer isn't it, the judge will have to refute the given argument.
- It keeps a list of the valid arbitrary digits entered by each judge.

- It prints a line with no contents after each judge has assigned scores to the task.

- It offers justification for the move of judges to award the project points from within the proposed range.

Visualization award winning function

```
# Function to visualize award-winning projects
def visualize_projects():
    sorted_projects = sorted(projects.items(), key=lambda x: x[1].get("Total Points", 0), reverse=True)

    # Display details of 1st, 2nd, and 3rd place projects
    for i in range(3):
        # Determine the number of stars based on total points
        stars = "*" * sorted_projects[i][1].get("Total Points", 0)
        for _ in range(len(stars)):
            print("*")
        print(f"{i+1}st Place:")
        print("Project Name:", sorted_projects[i][1]["Project Name"])
        print("Country:", sorted_projects[i][1]["Country"])
        print("Total Points:", sorted_projects[i][1].get("Total Points", 0))
        print("Project ID:", sorted_projects[i][0])  # Access project ID from the sorted projects
    print("Exiting the program...")
    exit()
```

- The projects that are highlighted by visualize_projects() have total points, and this is to depict the information of these projects' details.Here's how it works:

- It separates the projects list utilizing the projects dictionary to order it in respect to the total points in descending order. The total point constitutes the yardstick, the highest totalled point projects will be displayed first.

- It goes through every project in this list of the sorted list and point out the details of first, second, and third place ones from this list. The server is designed to manage the display of winning projects by showing their

names, country, total points, and project ID.

- It prints each star for every reached point accumulated. This makes the total points received easily seen.

- Moreover, the output is the following, " Exiting the program..." and the program exits.

Welcome page function

```
# Welcome page for your application
def display_welcome():
    welcome_text = """


$$$$$$$$\\                $$\\            $$$$$$$$\\
\\__$$  __|               $$ |           $$  _____|
   $$ | $$$$$$\\   $$$$$$$\\ $$$$$$$\\       $$ |      $$\\   $$\\  $$$$$$\\   $$$$$$\\
   $$ |$$  __$$\\ $$  _____|$$  __$$\\      $$$$$\\    \\$$\\ $$  |$$  __$$\\ $$  __$$\\
   $$ |$$$$$$$$ |$$ /      $$ |  $$ |     $$  __|    \\$$$$  / $$ /  $$ |$$ /  $$ |
   $$ |$$   ____|$$ |      $$ |  $$ |     $$ |       $$  $$<  $$ |  $$ |$$ |  $$ |
   $$ |\\$$$$$$$\\ \\$$$$$$$\\ $$ |  $$ |     $$$$$$$$\\ $$  /\\$$\\ $$$$$$$  |\\$$$$$$  |
   \\__| \\_______| \\_______|\\__|  \\__|     \\________|\\__/  \\__|$$  ____/  \\______/
                                              $$ |
                                              $$ |
                                              \\__|
    Welcome to MY Application!
    This is your TechExpo - Event Mangement Systerm.
    """
    print(welcome_text)
    import time
    time.sleep(4)
```

- Display_welcome() is a function that is used as a starting page of your application. It shows an embellished welcoming message together with a concise explanation of your app in the next scene.Here's how it works:

- It sets the variable welcome_text (a multipoint string containing the welcome message and description of your application) in stylish manner.

- It outputs welcome_text by the print () function. Therefore, the welcome message appears on the screen, which is aimed at the user.

- It imports the 'time' module so that a waiting time of 4 seconds will follow the display of the welcome message which will be done via the function time.sleep(). This is a chance for users to read the welcome message first and then move on to the more subsequent steps.

- This function is the first thing that users see when launching your application so is has to be engaging and interactive. Lastly, they should like what they see on their first impression if you want to keep them interested.

Main program loop

```
# Main program loop
display_welcome()
try:
    while True:
        display_menu()
        choice = input("Enter your choice: ")

        if choice == "1":
            add_project()
        elif choice == "2":
            update_project()
        elif choice == "3":
            delete_project()
        elif choice == "4":
            view_projects()
        elif choice == "5":
            save_to_text_file()
        elif choice == "6":
            random_spotlight_selection()
        elif choice == "7":
            visualize_projects()
        elif choice == "8":
            print("Exiting the program...")
            break
        else:
            print("Invalid choice. Please try again.")
except KeyboardInterrupt:
    print("\nKeyboardInterrupt detected. Exiting the program...")
```

- main program loop helps to process data flows in your application context.Here's how it works:

- The process by using the display_welcome() function, a welcome message is shown first.

- The loop inside the try block never stops working until the user decides to go back to the menu or close the system.

- In the implementation of each loop, this function is using that it displays the menu options using the display_menu() function and prompts the user

to enter their choice.

- Depending on the user's choice, it calls the corresponding function: Depending on the user's choice, it calls the corresponding function:

- Choices (1-5) of implementing are performing the tasks of adding, updating, deleting, viewing, and saving of project data in the database.

- Chance 6 is the starter disorderly example select.

- The seventh choice is the screening of the award-winning projects.

- Nowadays, the economics of politics becomes the main determinant of candidacies, and campaigns are professionally driven.

- If the user enters a different choice from a given, it tells them that and challenges them to try again.

- When the KeyboardInterrupt happens (Ctrl+C), it prints a message to show that the interruption took place, then exits from the program smoothly.

- By the end of this process, the major program loop, serves as a user friendly interface to facilitates communication between the users and the application that can be operated on various project management deals.

The End.