ROBERT GORDON UNIVERSITY ABERDEEN

INFORMATICS INSTITUTE OF TECHNOLOGY

BSc (Hons) Artificial Intelligence and Data Science

Module: CM1601

Individual Coursework Report

Module Leader: Ms. Sachinthani Perera

RGU Student ID : 2330949

IIT Student ID : 20231603

Student Name : Mindiya De Zoysa

Content

Table of Figures

Section AHD

Function description

Code;

```
import random

max_horses = 20
horse_count = 0

#Event Name
print("Rapid Run.\n")
#Writing the main topics as a function so that it would be easy for the user to enter the command
def topic():
   print("AHD - Adding horse details.\t"
        "UHD - Updating horse details.\t"
        "DHD - Deleting horse details.\t"
        "VHD - View the Registered horses.\n"
        "SHD - Saving the horse details.\t"
        "SDD - Selecting four horses randomly.\t"
        "WHD - Display the winning horses.\t"
        "VWH - Visualize winning horses.\n"
        "ESC - Exit the program.\n"
        "Please enter the command:")

#Writing a function for all the main topic and then putting a return for call it again
topic()

def AHD():
   ahd = input().strip().upper()
   return ahd


ahd_input = AHD()
# Print a newline for better formatting
print()

# Check if the input is "AHD"
if (ahd_input == "AHD"):
   # Collect 20 horse ID from the user
   values = set()

   for i in range(20):
      while True:
         user_input = input(f"Enter Horse ID {i + 1}: ")
```

```
        # Check if the input can be converted to an integer and has exactly 3 digits
        if (user_input.isdigit() and len(user_input) == 3):
            value = int(user_input)

            # Check if the number is not repeated
            if value not in values:
                values.add(value)
                break
            else:
                print("Error: Number is repeated. Enter a different number.")
        else:
            print("Error: Enter a 3-digit number.")

    # Now, contains 20 unique 3-digit numbers.
    print("Unique Horse IDs:", values)

    # Shuffle the list of values
    shuffled_values = list(values)
    random.shuffle(shuffled_values)

    # Group the values into a, b, c, and d
    group_a = shuffled_values[:5]
    group_b = shuffled_values[5:10]
    group_c = shuffled_values[10:15]
    group_d = shuffled_values[15:]

    # Display the groups
    print("Group A:", group_a)
    print("Group B:", group_b)
    print("Group C:", group_c)
    print("Group D:", group_d)

    # Inform the user that they are in the horse adding details page
    print("You are in the horse adding details page.")
    # Print a newline for better formatting
    print()
    # Create dictionaries to store horse details for each group
    horse_details_a = {}
    horse_details_b = {}
    horse_details_c = {}
    horse_details_d = {}

    while (horse_count < max_horses):
        horse_id = input(f"Please enter the horse ID of the {horse_count + 1} horse (Should be a three-digit
number): ")
```

```
if (horse_id.isdigit() and len(horse_id) == 3):
   horse_id = int(horse_id)
else:
   print("Invalid input. Please enter a three-digit number.")
   continue

# Check if the horse ID is already used
if (
      horse_id in horse_details_a
      or horse_id in horse_details_b
      or horse_id in horse_details_c
      or horse_id in horse_details_d
):
   print("This horse ID is already in use. Please enter a different one.")
   continue

horse_name = input("Please enter the horse name: ")
while True:
      try:
         horse_age = int(input("Please enter the age: "))
         if (1 <= horse_age <= 30):
            break
         else:
            print("Invalid age. Please enter an age between 1 and 30.")
      except ValueError:
         print("Invalid input. Please enter a valid age as a number.")
horse_breed = input("Please enter the horse breed: ")
jockey_name = input("Please enter the jockey name: ")
race_record = input("Please enter the race record: ")

horse_count += 1
# Save changes to a file
save_to_file()
# Determine the group and store the details in the respective dictionary
if horse_id in group_a:
   horse_details_a[horse_id] = {
      "Horse Name": horse_name,
      "Horse Age": horse_age,
      "Horse Breed": horse_breed,
      "Jockey Name": jockey_name,
      "Race Record": race_record
   }
elif horse_id in group_b:
   horse_details_b[horse_id] = {
      "Horse Name": horse_name,
      "Horse Age": horse_age,
      "Horse Breed": horse_breed,
```

```
            "Jockey Name": jockey_name,
            "Race Record": race_record
        }
    elif horse_id in group_c:
        horse_details_c[horse_id] = {
            "Horse Name": horse_name,
            "Horse Age": horse_age,
            "Horse Breed": horse_breed,
            "Jockey Name": jockey_name,
            "Race Record": race_record
        }
    elif horse_id in group_d:
        horse_details_d[horse_id] = {
            "Horse Name": horse_name,
            "Horse Age": horse_age,
            "Horse Breed": horse_breed,
            "Jockey Name": jockey_name,
            "Race Record": race_record
        }

    print(f"Horse {horse_count} details added.")

  print(f"Maximum number of horses ({max_horses}) reached. Cannot add more horses.")
else:
  # Inform the user that they have entered a wrong command
  print("You have entered a wrong command.")
# Print a newline for better formatting
print()

# Print the grouped horse details
print("Group A Horse Details:")
for horse_id, details in horse_details_a.items():
  print(f"Horse ID: {horse_id}, Details: {details}")

print("\nGroup B Horse Details:")
for horse_id, details in horse_details_b.items():
  print(f"Horse ID: {horse_id}, Details: {details}")

print("\nGroup C Horse Details:")
for horse_id, details in horse_details_c.items():
  print(f"Horse ID: {horse_id}, Details: {details}")

print("\nGroup D Horse Details:")
for horse_id, details in horse_details_d.items():
  print(f"Horse ID: {horse_id}, Details: {details}")
```

Assumptions;

The assumptions I took was to group the horses to there group by the using by doing a random shuffle so that the users won't be able to them in to the groups what they want. Then the other assumption was to display the saved details which is the same thing done in VHD section but before that I did it here.

Screenshot of the output

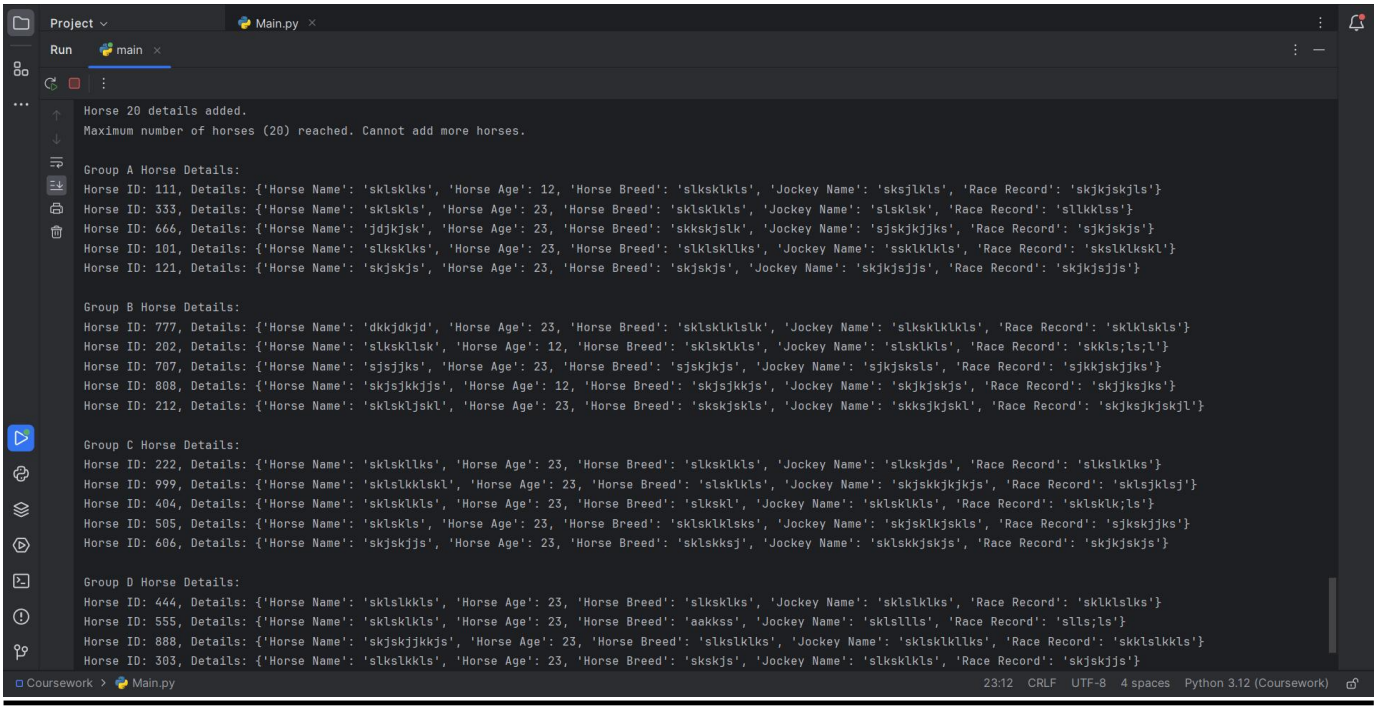

Figure 1 AHD output

Flow Chart

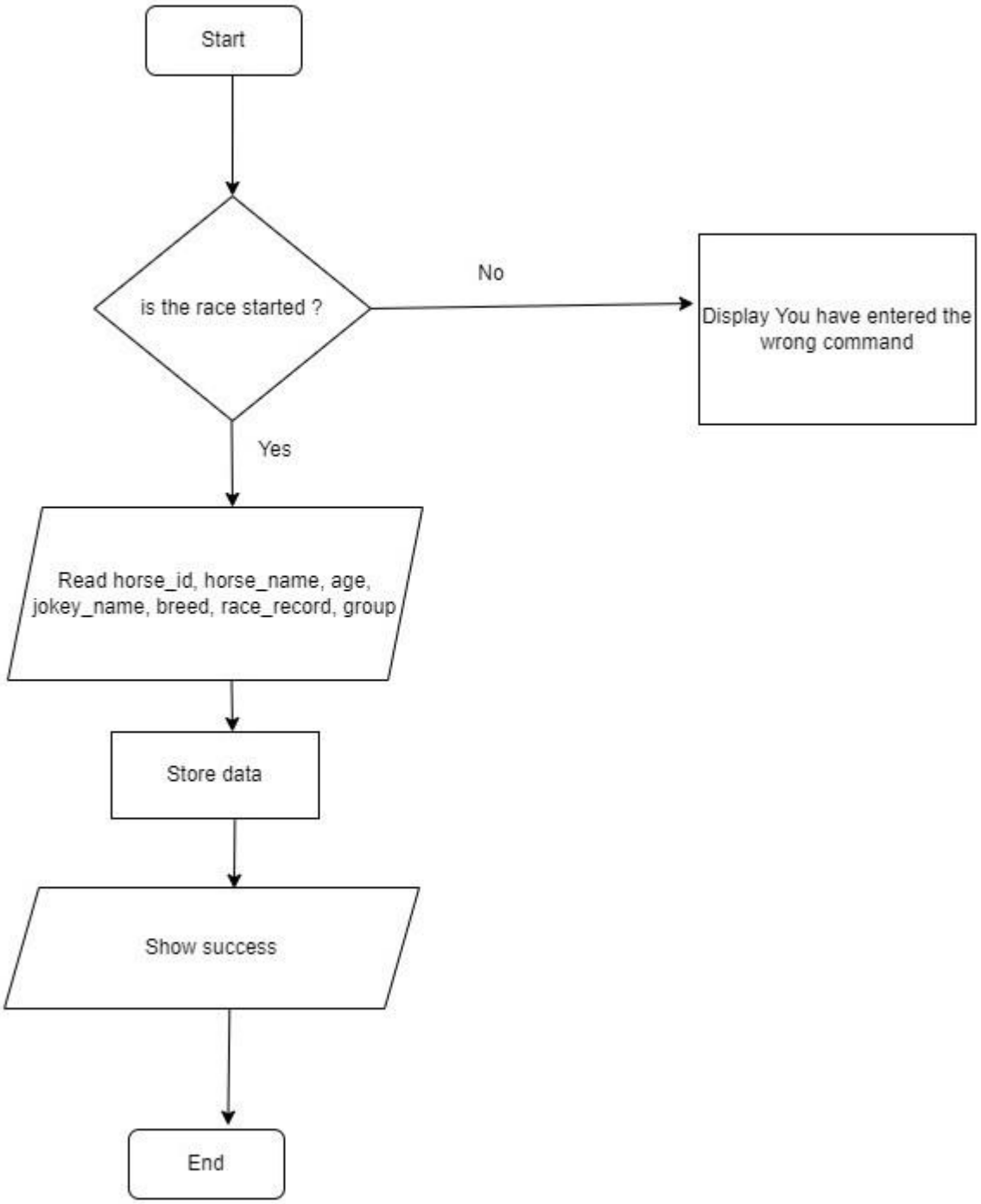

Figure 2 AHD flowchart

Section UHD

Function description

Code;

```
#Event Name
print("Rapid Run.\n")
#Writing the main topics as a function so that it would be easy for the user to enter the command
def topic():
   print("AHD - Adding horse details.\t"
        "UHD - Updating horse details.\t"
        "DHD - Deleting horse details.\t"
        "VHD - View the Registered horses.\n"
        "SHD - Saving the horse details.\t"
        "SDD - Selecting four horses randomly.\t"
        "WHD - Display the winning horses.\t"
        "VWH - Visualize winning horses.\n"
        "ESC - Exit the program.\n"
        "Please enter the command:")

#Writing a function for all the main topic and then putting a return for call it again
def UHD():
   uhd = input().strip().upper()
   return uhd


topic()
uhd_input = UHD()
# Print a newline for better formatting
print()

# Check if the input is "UHD"
if uhd_input == "UHD":
   # Inform the user that they are in the horse update details page
   print("You are in the horse update details page.")

   # Ask the user if they want to update any user details
   uhd_user_q = input("Do you want to update any horse details? (Please use 'Yes' or 'No'):
").strip().lower()

   if (uhd_user_q == "yes"):
      while True:
         # Get the Horse ID to update from the user
         uhd_horse_id = input("Enter the Horse ID you want to update: ")
```

```
# Convert input to integer assuming Horse IDs are integers
uhd_horse_id = int(uhd_horse_id)

# Check if the entered Horse ID exists in any of the groups
if uhd_horse_id in horse_details_a:
    uhd_horse_group = 'A'
elif uhd_horse_id in horse_details_b:
    uhd_horse_group = 'B'
elif uhd_horse_id in horse_details_c:
    uhd_horse_group = 'C'
elif uhd_horse_id in horse_details_d:
    uhd_horse_group = 'D'
else:
    # If the horse is not found, inform the user
    print(f"Horse ID {uhd_horse_id} not found. Please enter a valid Horse ID.")
    continue

# Prompt user for updated details
updated_name = input("Enter updated horse name: ")
updated_age = int(input("Enter updated horse age: "))
updated_breed = input("Enter updated horse breed: ")
updated_jockey = input("Enter updated jockey name: ")
updated_record = input("Enter updated race record: ")

# Update the details based on the group
if uhd_horse_group == 'A':
    horse_details_a[uhd_horse_id] = {
        "Horse Name": updated_name,
        "Horse Age": updated_age,
        "Horse Breed": updated_breed,
        "Jockey Name": updated_jockey,
        "Race Record": updated_record
    }
elif uhd_horse_group == 'B':
    horse_details_b[uhd_horse_id] = {
        "Horse Name": updated_name,
        "Horse Age": updated_age,
        "Horse Breed": updated_breed,
        "Jockey Name": updated_jockey,
        "Race Record": updated_record
    }
elif uhd_horse_group == 'C':
    horse_details_c[uhd_horse_id] = {
        "Horse Name": updated_name,
        "Horse Age": updated_age,
        "Horse Breed": updated_breed,
        "Jockey Name": updated_jockey,
```

```
            "Race Record": updated_record
        }
    elif uhd_horse_group == 'D':
        horse_details_d[uhd_horse_id] = {
            "Horse Name": updated_name,
            "Horse Age": updated_age,
            "Horse Breed": updated_breed,
            "Jockey Name": updated_jockey,
            "Race Record": updated_record
        }

    # If the horse is updated, inform the user
    print(f"Horse ID {uhd_horse_id} details updated successfully.")

    # Ask if the user wants to update more horse details
    uhd_repeat = input("Do you want to update more horse details? (Please use 'Yes' or 'No'):
").strip().lower()
    if (uhd_repeat != "yes"):
        # Save changes to a file and break the loop
        save_to_file()
        break

  else:
    # Inform the user that no horse updates have been done
    print("No horse updates have been done.")

else:
  # Inform the user that they have entered a wrong command
  print("You have entered a wrong command.")
# Print a newline for better formatting
print()
```

Assumptions;

No assumptions in this section.

Screenshot of the output

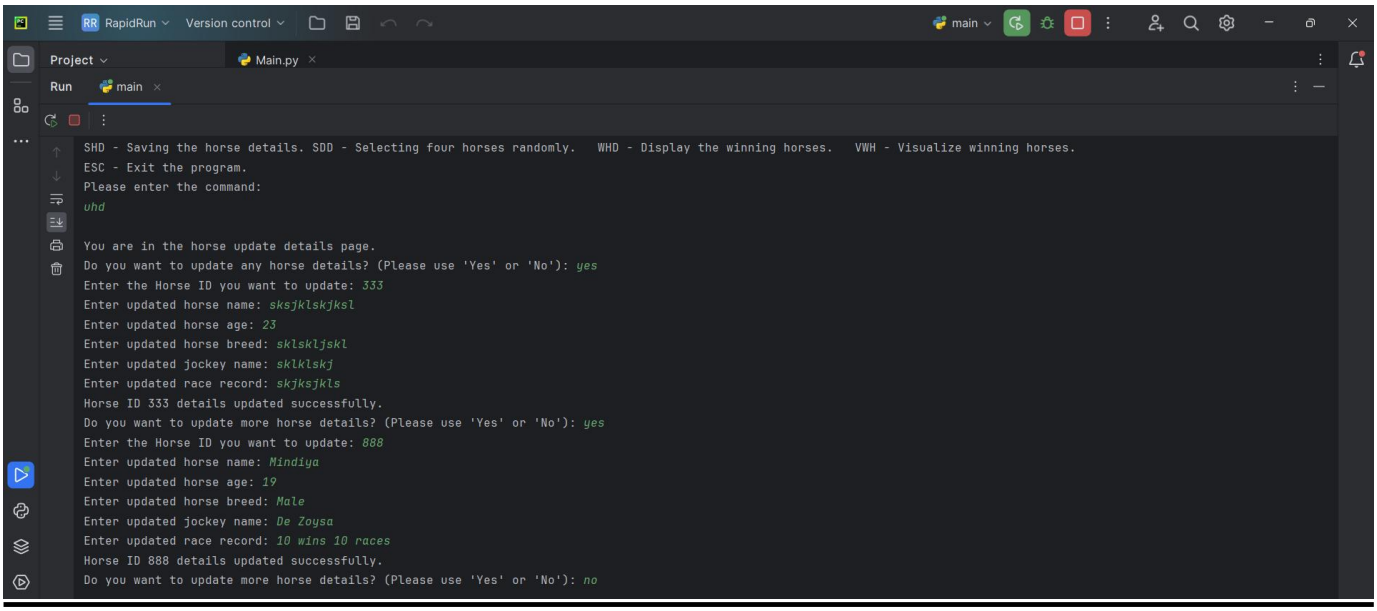

Figure 3 UHD output

Flow Chart

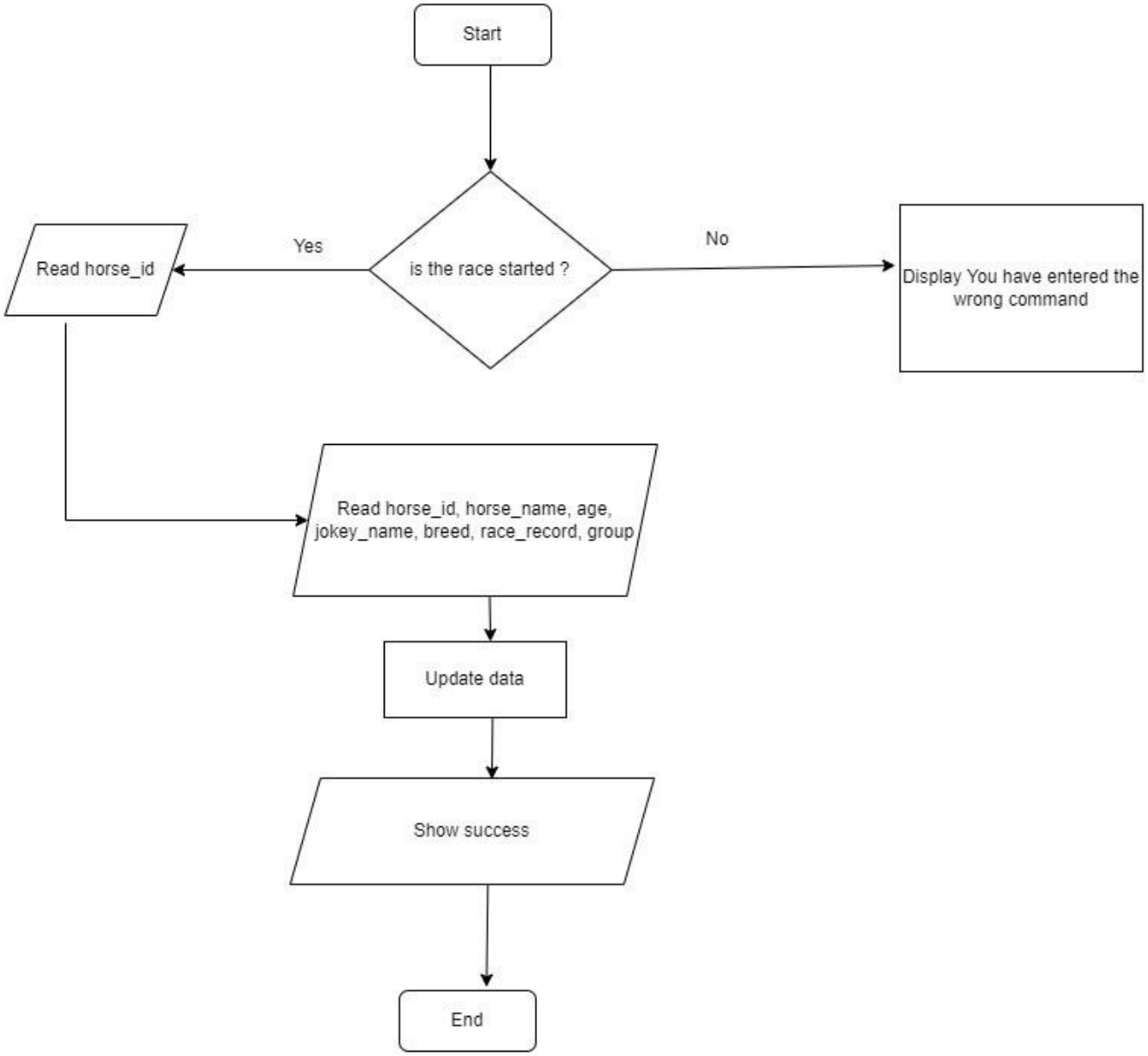

Figure 4 UHD flowchart

Section DHD

Function description

Code;

```
#Event Name
print("Rapid Run.\n")
#Writing the main topics as a function so that it would be easy for the user to enter the command
def topic():
   print("AHD - Adding horse details.\t"
        "UHD - Updating horse details.\t"
        "DHD - Deleting horse details.\t"
        "VHD - View the Registered horses.\n"
        "SHD - Saving the horse details.\t"
        "SDD - Selecting four horses randomly.\t"
        "WHD - Display the winning horses.\t"
        "VWH - Visualize winning horses.\n"
        "ESC - Exit the program.\n"
        "Please enter the command:")

#Writing a function for all the main topic and then putting a return for call it again
def DHD():
   dhd = input().strip().upper()
   return dhd

topic()
dhd_input = DHD()
# Print a newline for better formatting
print()

# Check if the input is "DHD"
if (dhd_input == "DHD"):
   # Inform the user that they are in the horse delete details page
   print("You are in the horse delete details page.")

   # Ask the user if they want to delete any user details
   dhd_user_q = input("Do you want to delete any user details (Please use 'Yes' or 'No'): ").strip().lower()
   # Check if the user wants to delete user details
   if (dhd_user_q == "yes"):
      # Get the Horse ID to delete from the user
      horse_id_to_delete = int(input("Enter the Horse ID you want to delete: "))

      # Lists to store group information and horse details
      group_lists = [group_a, group_b, group_c, group_d]
```

```
        horse_details_lists = [horse_details_a, horse_details_b, horse_details_c, horse_details_d]

        # Flag to check if the horse with the specified ID is found
        found = False

        # Iterate through each group and check if the Horse ID exists
        for i, group_list in enumerate(group_lists):
            if horse_id_to_delete in group_list:
                # If found, delete the horse details
                found = True
                horse_details_lists[i].pop(horse_id_to_delete, None)
                group_list.remove(horse_id_to_delete)
                print(f"Horse with ID {horse_id_to_delete} deleted.")
                break

        # If the horse is not found, inform the user
        if not found:
            print(f"No horse found with ID {horse_id_to_delete}.")
        # Ask the user if they want to delete more horse details
        while True:
            # Ask if the user wants to update more horse details
            dhd_repeat = input("Do you want to delete any more user details (Please use 'Yes' or 'No'):
").strip().lower()
            if (dhd_repeat == "yes"):
                # Save changes to a file and continue the loop
                save_to_file()
                continue
            else:
                # Break out of the loop if the user doesn't want to delete more details
                break

    else:
        # Inform the user that no horse deletes have been done
        print("No horse deletes haven't been done.")

else:
    # Inform the user that they have entered a wrong command
    print("You have entered a wrong command.")
# Print a newline for better formatting
print()
```

Assumptions;

No assumptions in this section.

Screenshot of the output

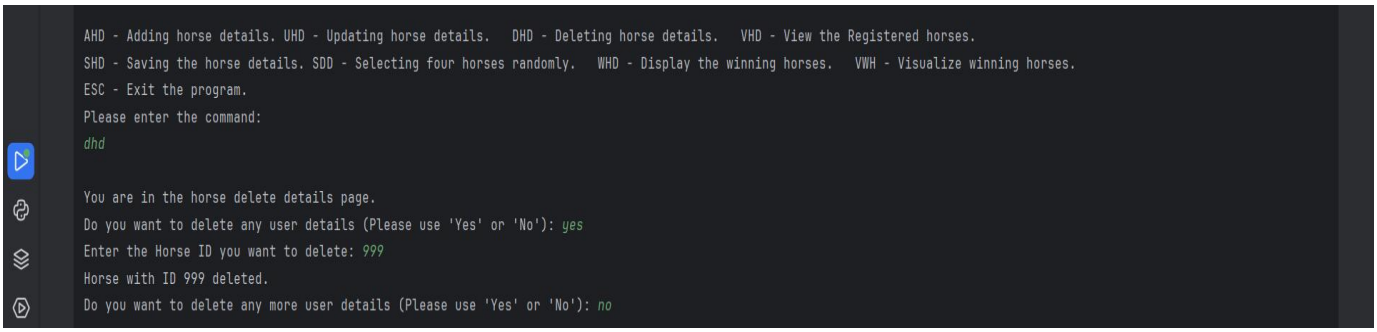

Figure 5 DHD output

Flow Chart

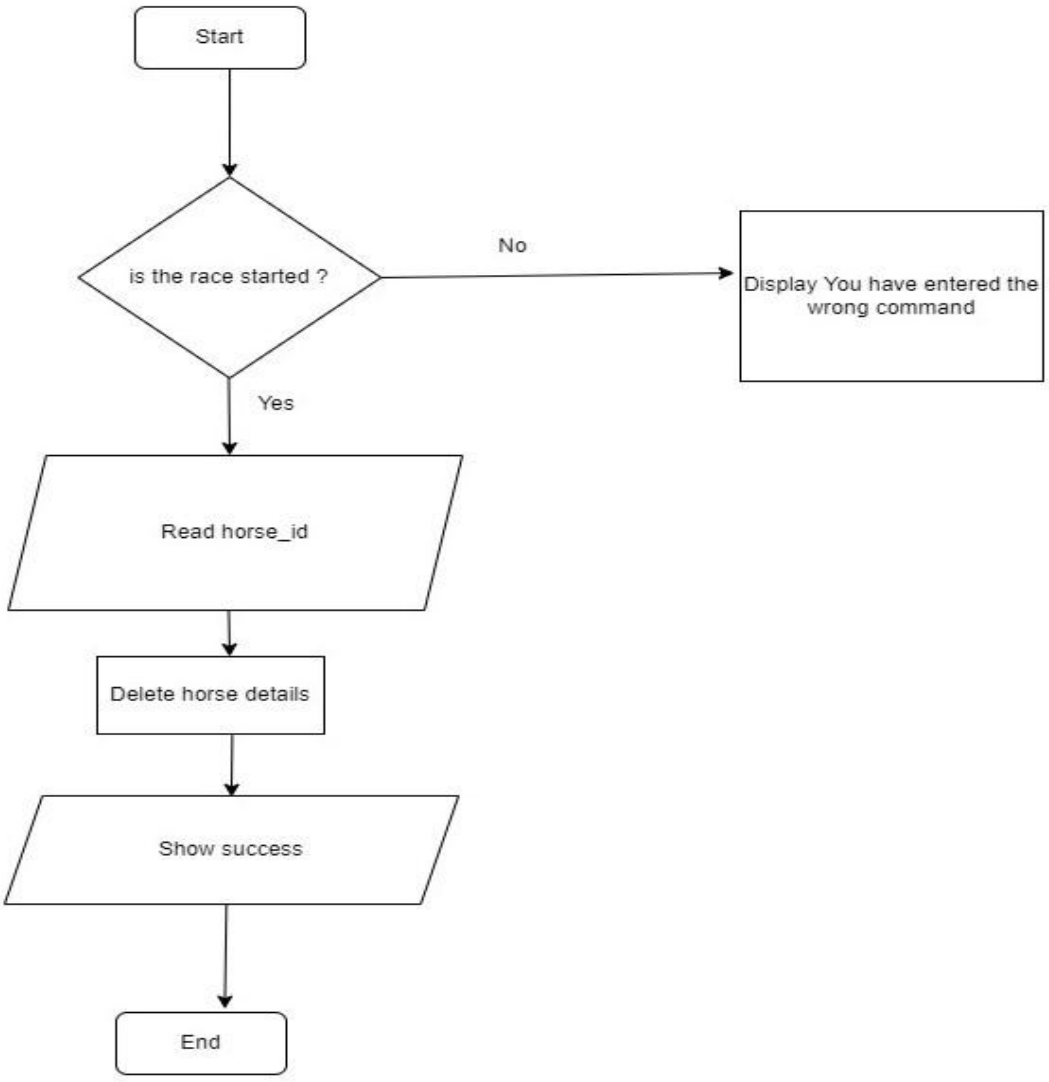

Figure 6 DHD flowchart

Section VHD

Function description

Code;

```
#Event Name
print("Rapid Run.\n")
#Writing the main topics as a function so that it would be easy for the user to enter the command
def topic():
   print("AHD - Adding horse details.\t"
        "UHD - Updating horse details.\t"
        "DHD - Deleting horse details.\t"
        "VHD - View the Registered horses.\n"
        "SHD - Saving the horse details.\t"
        "SDD - Selecting four horses randomly.\t"
        "WHD - Display the winning horses.\t"
        "VWH - Visualize winning horses.\n"
        "ESC - Exit the program.\n"
        "Please enter the command:")

#Writing a function for all the main topic and then putting a return for call it again
def VHD():
   vhd = input().strip().upper()
   return vhd

topic()
vhd_input = VHD()
# Print a newline for better formatting
print()

if (vhd_input == "VHD"):
   # Inform the user that they are in view the registered horses' details page
   print("You are in view the registered horses' details page.")
   # Sort horse details based on horse_id
   sorted_horse_details_a = dict(sorted(horse_details_a.items()))
   sorted_horse_details_b = dict(sorted(horse_details_b.items()))
   sorted_horse_details_c = dict(sorted(horse_details_c.items()))
   sorted_horse_details_d = dict(sorted(horse_details_d.items()))

   print("\nGroup A Horse Details (Sorted by Horse ID):")
   for horse_id, details in sorted_horse_details_a.items():
      print(f"Horse ID: {horse_id}, Details: {details}")

   print("\nGroup B Horse Details (Sorted by Horse ID):")
   for horse_id, details in sorted_horse_details_b.items():
```

```
        print(f"Horse ID: {horse_id}, Details: {details}")

    print("\nGroup C Horse Details (Sorted by Horse ID):")
    for horse_id, details in sorted_horse_details_c.items():
        print(f"Horse ID: {horse_id}, Details: {details}")

    print("\nGroup D Horse Details (Sorted by Horse ID):")
    for horse_id, details in sorted_horse_details_d.items():
        print(f"Horse ID: {horse_id}, Details: {details}")

else:
    # Inform the user that they have entered a wrong command
    print("You have entered a wrong command.")
# Print a newline for better formatting
print()
```

Assumptions;

No assumptions in this section.

Screenshot of the output

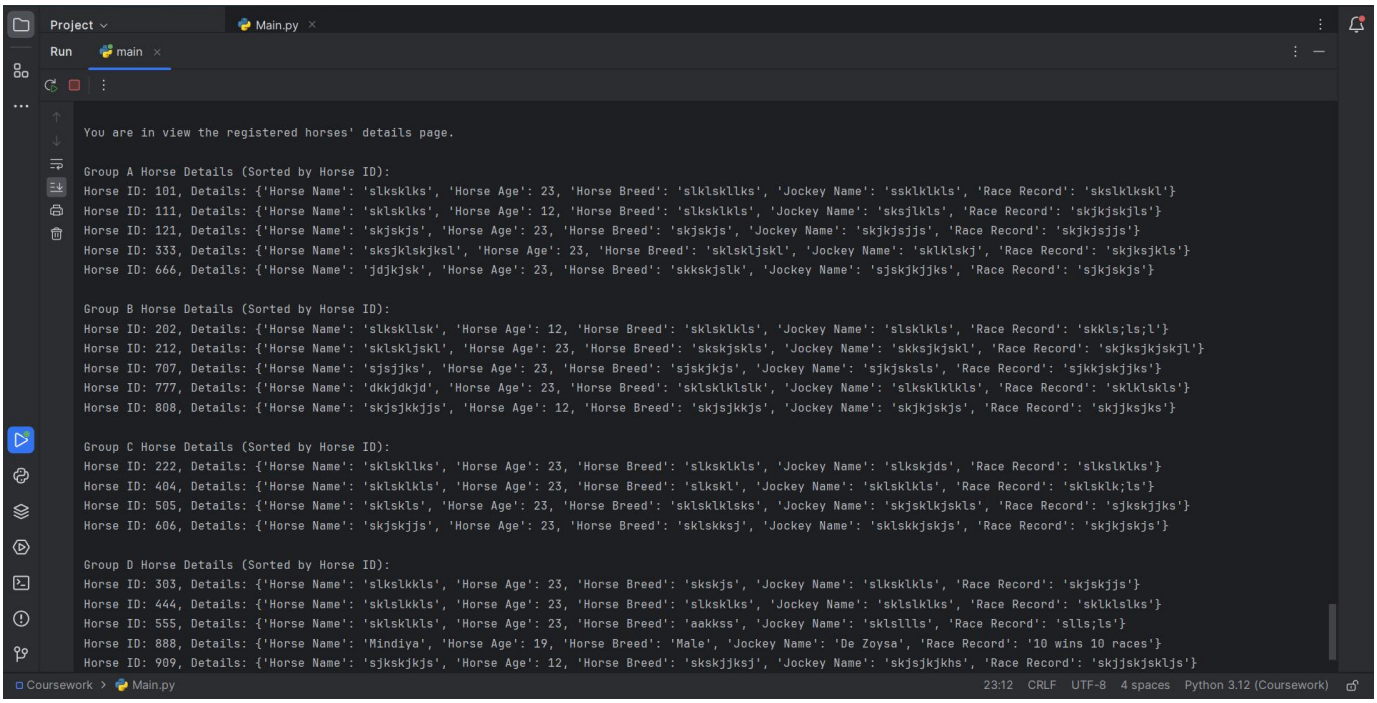

Figure 7 VHD output

Flow Chart

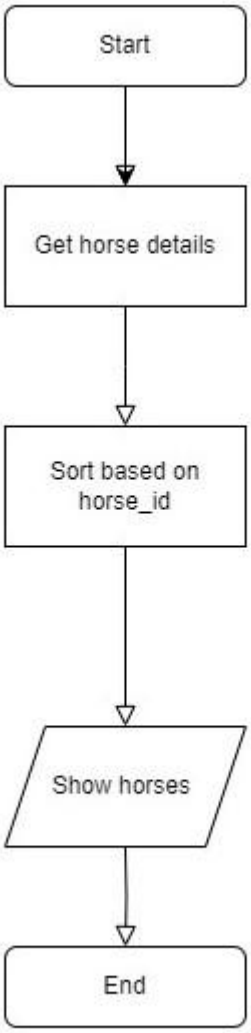

Figure 8 VHD flowchart

Section SHD

Function description

Code;

```
def save_to_file():
    # Combine all horse details into a dictionary
    all_horse_details = {
        'Group A': horse_details_a,
        'Group B': horse_details_b,
        'Group C': horse_details_c,
        'Group D': horse_details_d
    }

    # Save the details to a text file
    with open("horse_details.txt", "w") as file:
        for group, details in all_horse_details.items():
            file.write(f"{group}:\n")
            for key, value in details.items():
                file.write(f"  {key}: {value}\n")
            file.write("\n")

#Event Name
print("Rapid Run.\n")
#Writing the main topics as a function so that it would be easy for the user to enter the command
def topic():
    print("AHD - Adding horse details.\t"
          "UHD - Updating horse details.\t"
          "DHD - Deleting horse details.\t"
          "VHD - View the Registered horses.\n"
          "SHD - Saving the horse details.\t"
          "SDD - Selecting four horses randomly.\t"
          "WHD - Display the winning horses.\t"
          "VWH - Visualize winning horses.\n"
          "ESC - Exit the program.\n"
          "Please enter the command:")

#Writing a function for all the main topic and then putting a return for call it again
def SHD():
    shd = input().strip().upper()
    return shd

topic()
shd_input = SHD()
# Print a newline for better formatting
```

```
print()

if (shd_input == "SHD"):
    # Inform the user that they are in the save the horse details to the text file page
    print("You are in the save the horse details to the text file page.")
    print("The details you have add, update and delete are saved in a separate text file.")

    # Print a newline for better formatting
    print()
else:
    # Inform the user that they have entered a wrong command
    print("You have entered a wrong command.")
# Print a newline for better formatting
print()
```

Assumptions;

No assumptions in this section.

Screenshot of the output

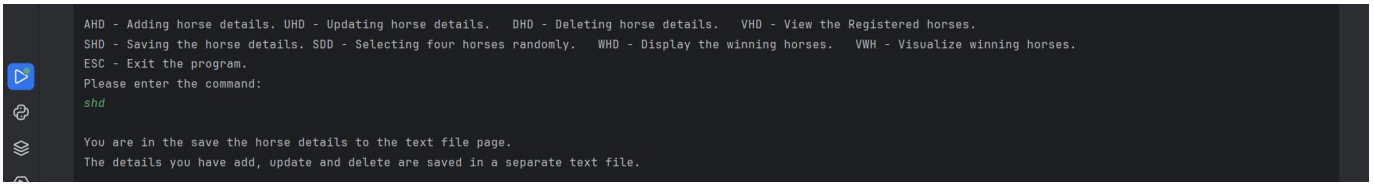

Figure 9 SHD output

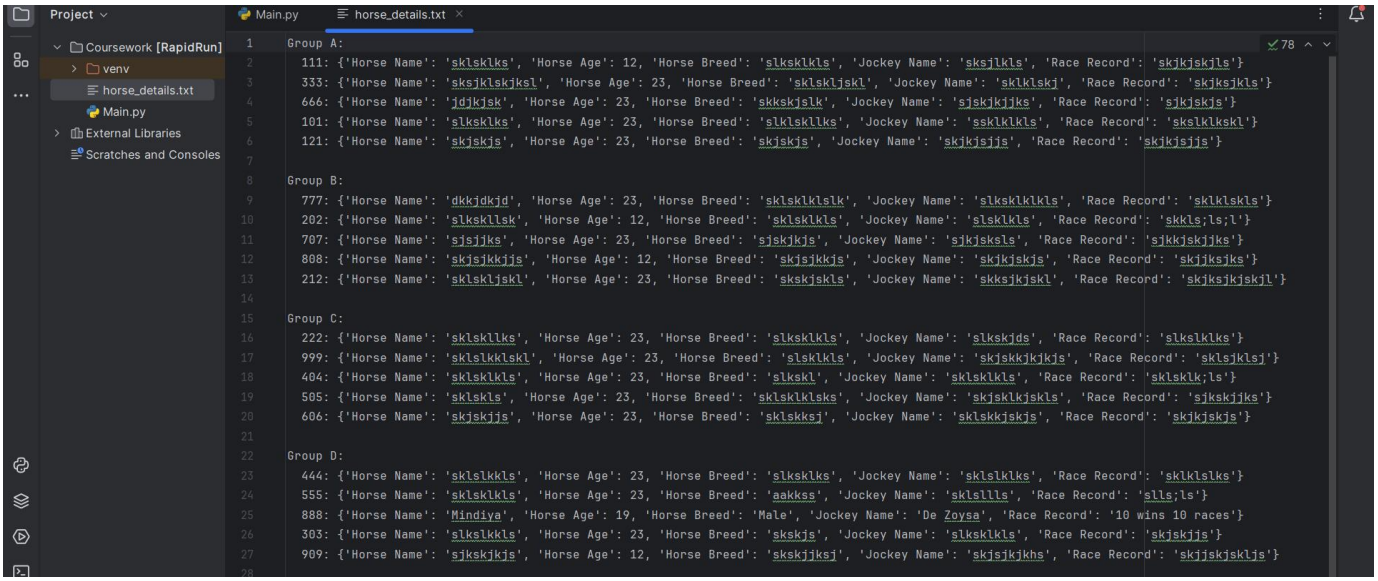

Figure 11 SHD Text File

Flow Chart

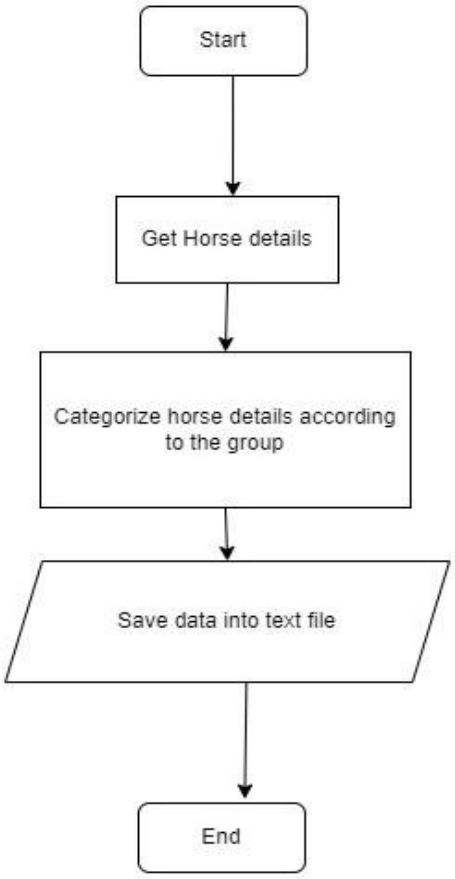

Figure 11 SHD flowchart

Section SDD

Function description

Code;

```
import random

#Event Name
print("Rapid Run.\n")
#Writing the main topics as a function so that it would be easy for the user to enter the command
def topic():
   print("AHD - Adding horse details.\t"
        "UHD - Updating horse details.\t"
        "DHD - Deleting horse details.\t"
        "VHD - View the Registered horses.\n"
```

```
        "SHD - Saving the horse details.\t"
        "SDD - Selecting four horses randomly.\t"
        "WHD - Display the winning horses.\t"
        "VWH - Visualize winning horses.\n"
        "ESC - Exit the program.\n"
        "Please enter the command:")

#Writing a function for all the main topic and then putting a return for call it again
def SDD():
    sdd = input().strip().upper()
    return sdd

def random_draw(horse_details_group):
    #Simulate a random draw and select a horse from the given group.
    group_horses = list(horse_details_group.keys())
    selected_horse_id = random.choice(group_horses)
    return selected_horse_id, horse_details_group[selected_horse_id]

def display_selected_horses(selected_horses):
    #Display the details of the randomly selected horses.
    print("\nRandomly Selected Horses for the Final Round:")
    for group, (horse_id, details) in selected_horses.items():
        print(f"Group {group} - Horse ID: {horse_id}, Details: {details}")

topic()
sdd_input = SDD()
# Print a newline for better formatting
print()

if (sdd_input == "SDD"):
    # Inform the user that they are in the selecting four horses randomly page
    print("You are in the selecting four horses randomly page.")
    selected_horses = {
        'A': random_draw(horse_details_a),
        'B': random_draw(horse_details_b),
        'C': random_draw(horse_details_c),
        'D': random_draw(horse_details_d)
    }

    display_selected_horses(selected_horses)
else:
    # Inform the user that they have entered a wrong command
    print("You have entered a wrong command.")
# Print a newline for better formatting
print()
```

Assumptions;

No assumptions in this section.

Screenshot of the output

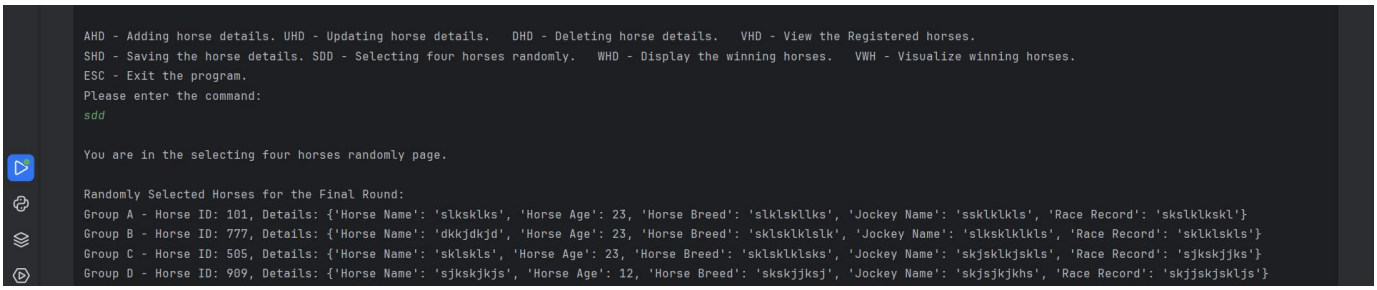

Figure 12 SDD output

Flow Chart

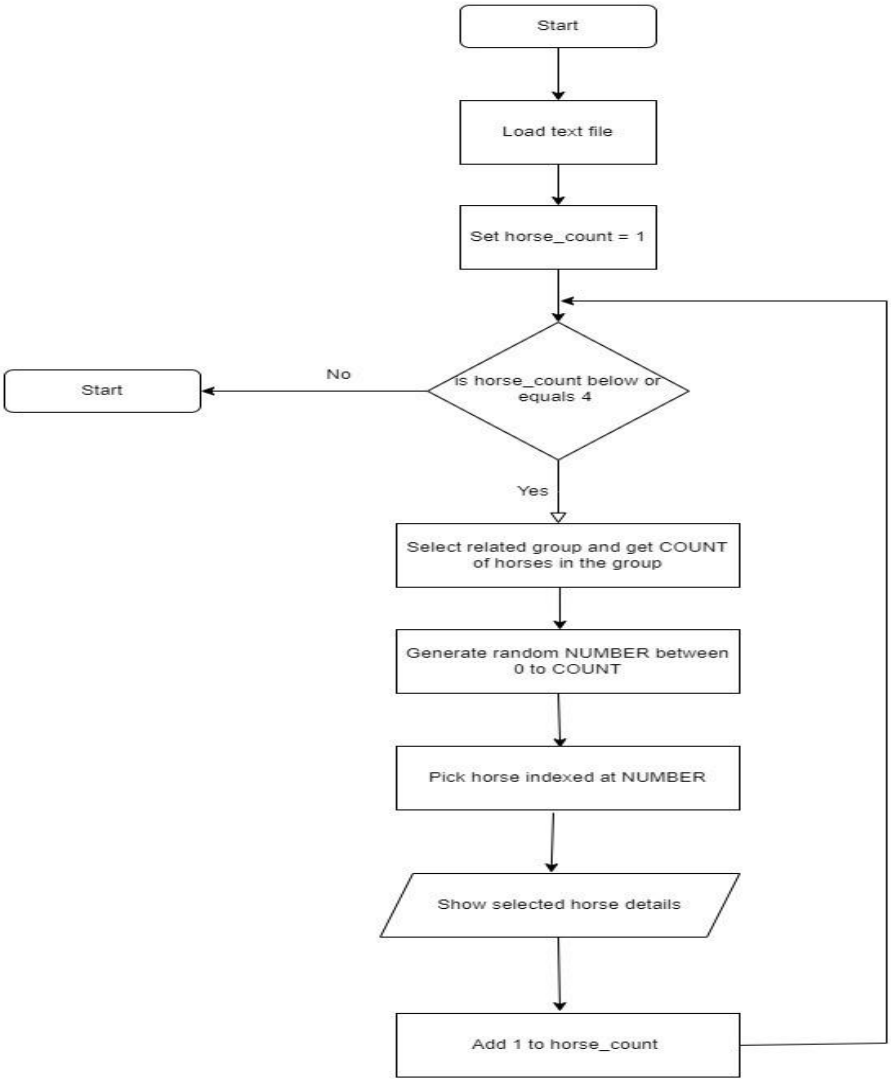

Figure 13 SDD flowchart

Section WHD

Function description

Code;

```
import random

#Event Name
print("Rapid Run.\n")
#Writing the main topics as a function so that it would be easy for the user to enter the command
def topic():
   print("AHD - Adding horse details.\t"
        "UHD - Updating horse details.\t"
        "DHD - Deleting horse details.\t"
        "VHD - View the Registered horses.\n"
        "SHD - Saving the horse details.\t"
        "SDD - Selecting four horses randomly.\t"
        "WHD - Display the winning horses.\t"
        "VWH - Visualize winning horses.\n"
        "ESC - Exit the program.\n"
        "Please enter the command:")

#Writing a function for all the main topic and then putting a return for call it again
def WHD():
   whd = input().strip().upper()
   return whd

def assign_random_times(selected_horses):
   # Assign random times between 0 to 90 seconds for each selected horse
   for group, (horse_id, details) in selected_horses.items():
      selected_horses[group] = (horse_id, details, random.randint(0, 90))

def display_final_results(final_results):
   # Display the final results with 1st, 2nd, and 3rd places
   print("\nFinal Results:")
   for place, (group, (horse_id, details, time_taken)) in enumerate(final_results, start=1):
      print(f"{place} Place - Group {group} - Horse ID: {horse_id}, Details: {details}, Time Taken:
{time_taken} seconds")

def get_final_results(selected_horses):
   # Sort the horses based on the time taken and get the final results
   sorted_horses = sorted(selected_horses.items(), key=lambda x: x[1][2])
   return sorted_horses[:3]

topic()
```

```
whd_input = WHD()
# Print a newline for better formatting
print()

if (whd_input == "WHD"):
    # Inform the user that they are in the display winning horses page
    print("You are in the display winning horses page.")
    # Assign random times to the selected horses
    assign_random_times(selected_horses)

    # Display the randomly assigned times for visualization
    print("\nRandomly Assigned Times for Visualization:")
    for group, (horse_id, details, time_taken) in selected_horses.items():
        print(f"Group {group} - Horse ID: {horse_id}, Time Assigned: {time_taken} seconds")

    # Get the final results based on the time taken
    final_results = get_final_results(selected_horses)

    # Display the final results with 1st, 2nd, and 3rd places
    display_final_results(final_results)

else:
    # Inform the user that they have entered a wrong command
    print("You have entered a wrong command.")
# Print a newline for better formatting
print()
```

Assumptions;

No assumptions in this section.

Screenshot of the output

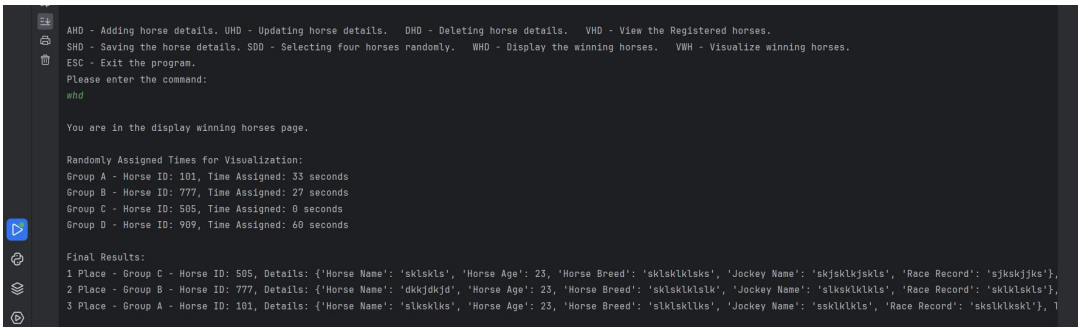

Figure 14 WHD output

Flow Chart

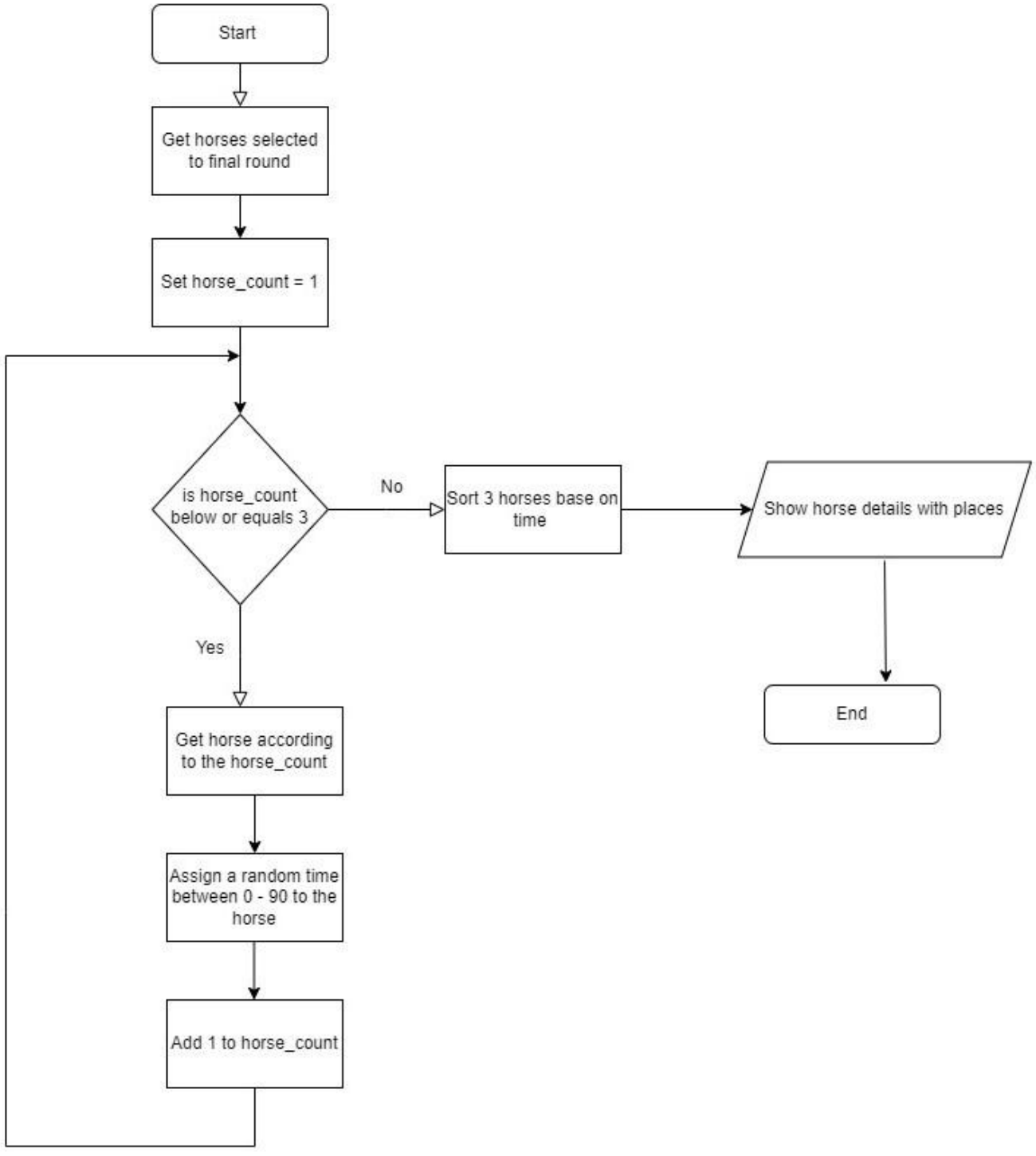

Figure 15 WHD flowchart

Section VWH

Function description

Code;

```
#Event Name
print("Rapid Run.\n")
#Writing the main topics as a function so that it would be easy for the user to enter the command
def topic():
   print("AHD - Adding horse details.\t"
        "UHD - Updating horse details.\t"
        "DHD - Deleting horse details.\t"
        "VHD - View the Registered horses.\n"
        "SHD - Saving the horse details.\t"
        "SDD - Selecting four horses randomly.\t"
        "WHD - Display the winning horses.\t"
        "VWH - Visualize winning horses.\n"
        "ESC - Exit the program.\n"
        "Please enter the command:")

#Writing a function for all the main topic and then putting a return for call it again
def VWH():
   vwh = input().strip().upper()
   return vwh

def visualize_time(time_taken):
   #Visualize the time spent by a horse using '*'. Each '*' represents 10 seconds.
   num_stars = time_taken // 10
   return '*' * num_stars

def visualize_winning_horses(final_results):
   #Visualize the time spent by each winning horse.
   print("\nVisualizing Winning Horses:")
   for place, (group, (horse_id, details, time_taken)) in enumerate(final_results, start=1):
      time_visualization = visualize_time(time_taken)
      print(f"Horse {place}: {time_visualization.ljust(20)}{horse_id} {time_taken}s ({place} Place)")

topic()
vwh_input = VWH()
# Print a newline for better formatting
print()

if (vwh_input == "VWH"):
   # Inform the user that they are in the Visualize Winning horses page
   print("You are in the Visualize Winning horses page.")
```

```
    # Visualize the time spent by each winning horse
    visualize_winning_horses(final_results)

else:
    # Inform the user that they have entered a wrong command
    print("You have entered a wrong command.")
# Print a newline for better formatting
print()
```

Assumptions;

No assumptions in this section.

Screenshot of the output

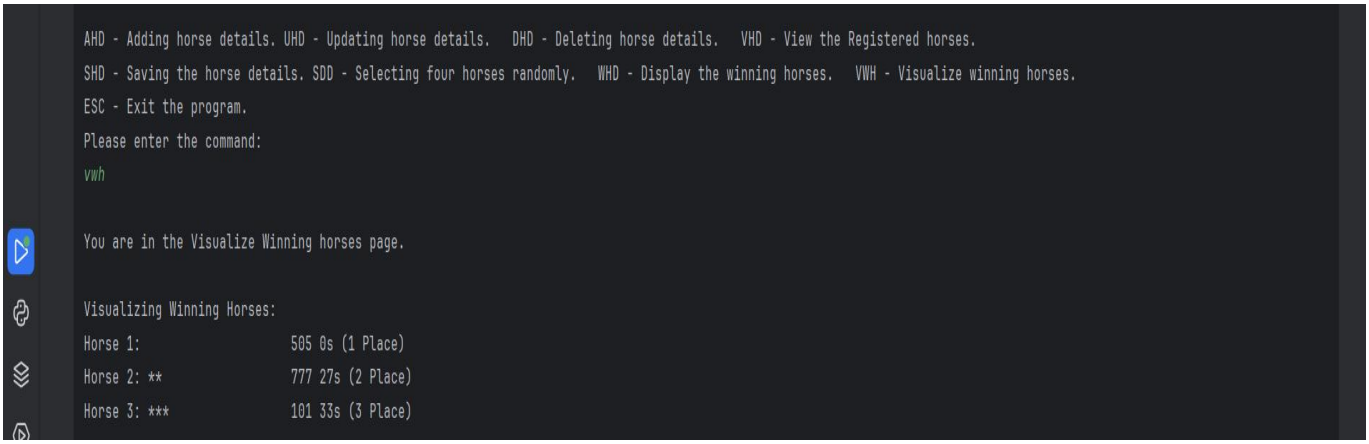

Figure 16 VWH output

Flow Chart

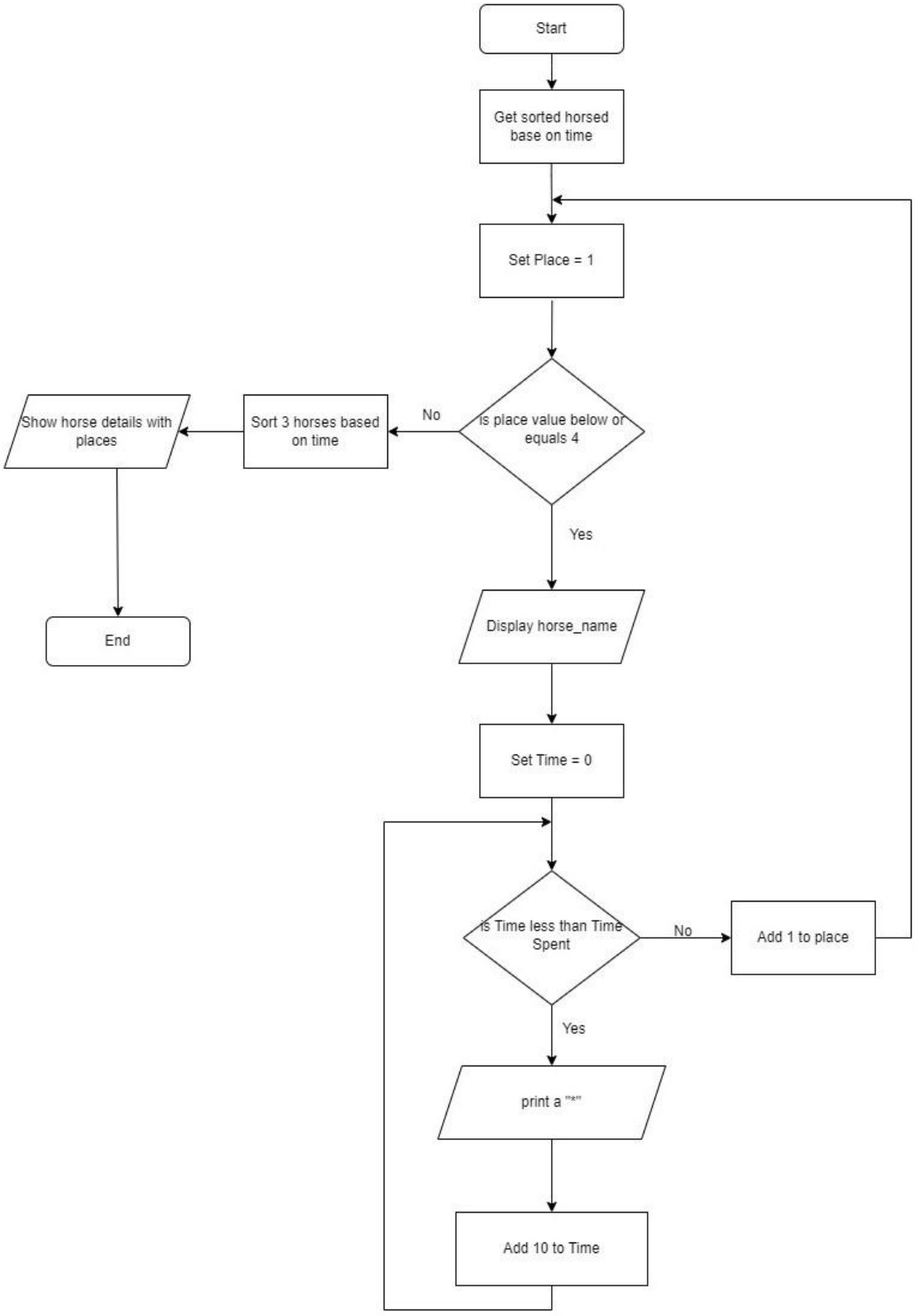

Figure 17 VWH flowchart

Section ESC

Function description

Code;

```
#Event Name
print("Rapid Run.\n")
#Writing the main topics as a function so that it would be easy for the user to enter the command
def topic():
   print("AHD - Adding horse details.\t"
        "UHD - Updating horse details.\t"
        "DHD - Deleting horse details.\t"
        "VHD - View the Registered horses.\n"
        "SHD - Saving the horse details.\t"
        "SDD - Selecting four horses randomly.\t"
        "WHD - Display the winning horses.\t"
        "VWH - Visualize winning horses.\n"
        "ESC - Exit the program.\n"
        "Please enter the command:")

#Writing a function for all the main topic and then putting a return for call it again
def ESC():
   esc = input().strip().upper()
   return esc

topic()
esc_input = ESC()

while True:
   if (esc_input == "ESC"):
      break
```

Assumptions;

No assumptions in this section.

Screenshot of the output

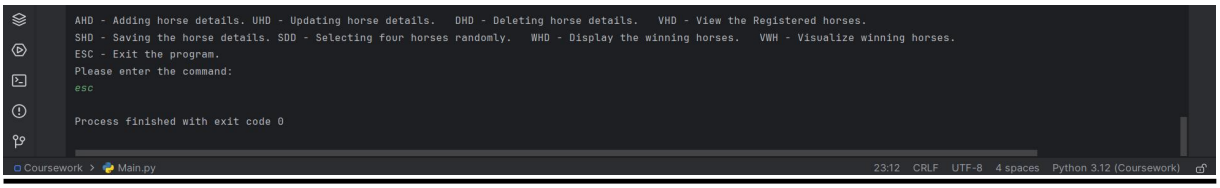

Figure 18 ESC output

References

EDUCBA. (2020). Python Validation | Types and Examples of Python Validation. [online] Available at: https://www.educba.com/python-validation/.

GeeksforGeeks. (2017). File Handling in Python. [online] Available at: https://www.geeksforgeeks.org/file-handling-python/?ref=lbp.

W3Schools (n.d.). Python - Global Variables. [online] www.w3schools.com. Available at: https://www.w3schools.com/python/python_variables_global.asp.

W3schools.com. (2019). Python File Open. [online] Available at: https://www.w3schools.com/python/python_file_open.asp.

HackerRank. (n.d.). Solve Python Code Challenges. [online] Available at: https://www.hackerrank.com/domains/python