Rapid Race Documentation

Python project with CLI

Stephan Nelishka Dabare

Contents

Table of Figures

1. Program Initialization

Python Code

```
import random
import time

# codes for bold and cyan text.
bold_cyan_text = "\033[1;36m"
reset_format = "\033[0m"

# Title of the program.
print()
print("-" * 156)
print(" " * 65, bold_cyan_text + "~~~~ Rapid Race ~~~~" + reset_format)
print("-" * 156)


# Initializing empty lists to store horse details.
horse_ids = []
horse_names = []
jockey_names = []
horse_ages = []
horse_breeds = []
race_records = []
horse_groups = []


# Create a dictionary to keep track of horse counts for each group(A, B, C, D)
horse_counts = {"A": 0, "B": 0, "C": 0, "D": 0}

#Create a dictionary to insert horses selected for finals.
horses_for_final = {}
```

Output

```
----------------------------------------------------------------------------------------------------------------------------
                                                  ~~~~ Rapid Race ~~~~
----------------------------------------------------------------------------------------------------------------------------
```

Figure 1 Display Race Name

Function Description

This function displays the title of the program like “Rapid Race.” Here, used a stylized way with bold cyan text. It also initializes some empty lists to store details about horses such as horse names, horse IDs, jockey names, horse ages, breed, race details and groups. Furthermore, it creates dictionaries to keep track in each group horse count and those selected for finals.

2. AHD- Adding Horse Details

AHD Function Flowchart

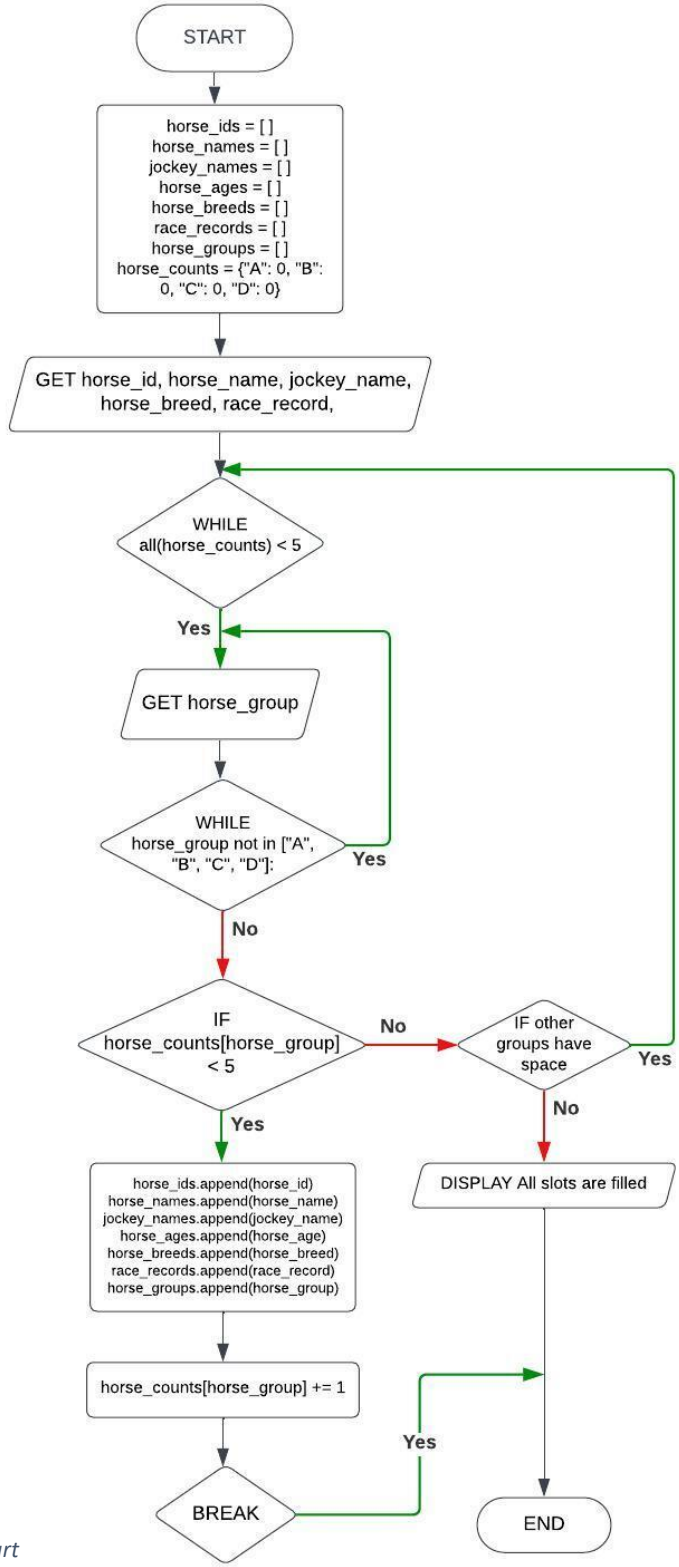

Figure 2 AHD Flowchart

AHD Function Python Code

```python
def add_horse_details():
    # Prompt for horse details.
    while True:
        # Validate horse ID in the correct data type.
        try:
            while True:
                horse_id = int(input("Enter Horse ID: "))
                # Changing the data type of horse_id to string.
                horse_id = str(horse_id)
                # Checking whether horse ID is taken before.
                if horse_id in horse_ids:
                    print("Horse ID is taken. Try another one")
                else:
                    break
            break
        except:
            print("Invalid input")

    # Input horse name.
    horse_name = input("Enter Horse Name: ")

    # Input jockey name.
    jockey_name = input("Enter Jockey Name: ")

    # Validate age in correct data type.
    while True:
        try:
            horse_age = int(input("Enter horse age: "))
            break
        except:
            print("Invalid input")

    # Input horse breed.
    horse_breed = input("Enter Horse Breed: ")

    # Validate wins and races compete in correct data type.
    while True:
        while True:
            try:
                # Input number of wins.
                win = int(input("Number of Wins: "))
                break
            except:
                print("Invalid input")
        while True:
            try:
                # Input number of races compete.
                races = int(input("Number of races compete: "))
                break
            except:
                print("Invalid input")
        # Checking whether win count greater than compete races count.
        if win <= races:
            # Assign variable called race record.
            race_record = f"{win} wins in {races} races"
            break
```

```
        else:
            print("Number of races compete is wrong")
    while True:
        horse_group = input("Enter Horse Group (A, B, C, or D): ").upper()
        # Validate horse group.
        while horse_group not in ["A", "B", "C", "D"]:
            horse_group = input("Invalid group. Enter Horse Group (A, B, C, or D): 
").upper()
        # checking whether one group have only 5 horses.
        if horse_counts[horse_group] < 5:
            print("Horse added to group", horse_group)
            # Add horse details to relevant lists.
            horse_ids.append(horse_id)
            horse_names.append(horse_name)
            jockey_names.append(jockey_name)
            horse_ages.append(horse_age)
            horse_breeds.append(horse_breed)
            race_records.append(race_record)
            horse_groups.append(horse_group)

            # Update horse counts for the respective group.
            horse_counts[horse_group] += 1

            print("\nHorse details added successfully!")
            break
        # Checking whether other groups have enough space to insert horse details.
        elif horse_counts['A'] < 5 or horse_counts['B'] < 5 or horse_counts['C'] < 5 or 
horse_counts['D'] < 5:
            print(f"Cannot add horse to Group {horse_group}. Group is full.")
            print(horse_counts.items())
            print("Try another group")
        else:
            # When all groups are filled display this message.
            print("\nSorry All slots are filled")
            break
```

AHD Function Output

```
Select the option you want: ahd

Is race started? (Yes or No) no
Enter Horse ID: 123
Enter Horse Name: Spirit
Enter Jockey Name: Sam
Enter horse age: 5
Enter Horse Breed: Mustang
Number of Wins: 4
Number of races compete: 12
Enter Horse Group (A, B, C, or D): A
Horse added to group A

Horse details added successfully!
```

Figure 3 Screenshot of AHD output

AHD Function Description

This feature facilitates to add information on a new horse. The user is prompted to provide a various detail, including horse's ID, horse's name, jockey's name, horse's age, breed, wins, races competed, and the horse group (A, B, C, D). It verifies the input by making sure the Horse ID is unique, the age is an integer, and the wins are less than or equal to the race competed.

In this case each horse group can have only 5 horses. If the entered horse group has available slots, the horse details are added to relevant lists. Then the horse count for respective group is updated. If the selected group is full prompted to try another group, and also informed if all the groups are full.

3. DHD- Deleting Horse Details

DHD Function Flowchart

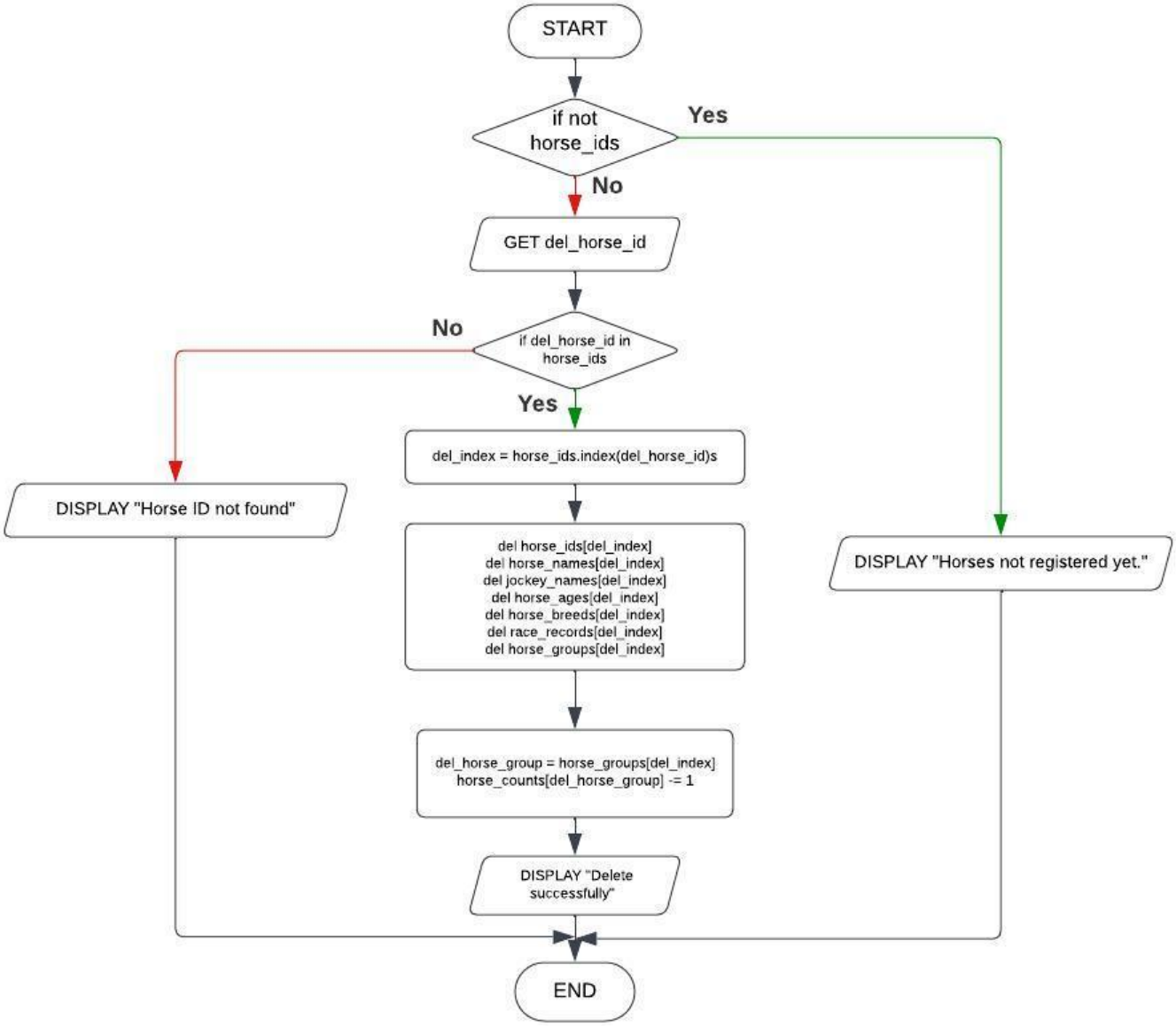

Figure 4 DHD Flowchart

DHD Function Python Code

```
def del_horse_details():
    # Check if any horses are registered yet
    if not horse_ids:
        print("\nNo Horses registered yet.")
        return

    # Get user input which horse ID need to be deleted.
    del_horse_id = input("Enter the Horse ID you want to delete: ")

    # Check if the entered Horse ID exists
    if del_horse_id in horse_ids:
        # Get user confirm to delete horse details.
        del_confirm = input(f"\nAre you sure to delete all the data of Horse with Horse
ID {del_horse_id} (Yes or No) ").lower()
        if del_confirm == "yes":
            # Get respective index according to the given horse ID.
            del_index = horse_ids.index(del_horse_id)

            # Remove horse details from lists
            del horse_ids[del_index]
            del horse_names[del_index]
            del jockey_names[del_index]
            del horse_ages[del_index]
            del horse_breeds[del_index]
            del race_records[del_index]
            # Deleting horse group assign to a variable
            del_horse_group = horse_groups[del_index]
            del horse_groups[del_index]

            # Update horse counts for the respective group
            horse_counts[del_horse_group] -= 1
            print(f"\nHorse with Horse ID {del_horse_id} deleted successfully!")
        elif del_confirm == "no":
            print(f"\nDeleting data of Horse with Horse ID {del_horse_id} is cancelled")
        else:
            # If user enter an ID which not in horse ids display this message.
            print("\nInvalid Input, Try again")
    else:
        print(f"\nHorse with Horse ID {del_horse_id} not found.\nTry again with a valid
Horse ID")
```

DHD Function Outputs

```
Select the option you want: dhd

Is race started? (Yes or No) no

No Horses registered yet.
```

Figure 5 Output when horses not registered yet.

```
Select the option you want: DHD

Is race started? (Yes or No) no
Enter the Horse ID you want to delete: 123

Are you sure to delete all the data of Horse with Horse ID 123 (Yes or No) yes

Horse with Horse ID 123 deleted successfully!
```

Figure 6 DHD output

DHD Function Description

Based on the given Horse ID, this function permits to delete the horse details related to that horse ID. First of all, it checks any horses are registered. If not, it notifies the user that there are no horses to remove. If horses are registered, the user is required to input the horse ID for the horse they wish to remove.

After that, the code checks to see if the entered horse ID exists in the horse_ids list. If it does, the user is prompted to confirm the deletion. If confirmed, the function deletes all the data related to the specified horse. Additionally, it modifies the horse count for the respective group. If user decided to not to delete, the function stops the deletion process. If the entered horse ID is not found, notifies the user, and tells them to try again.

4. UHD- Update Horse Details

UHD Function Flowchart

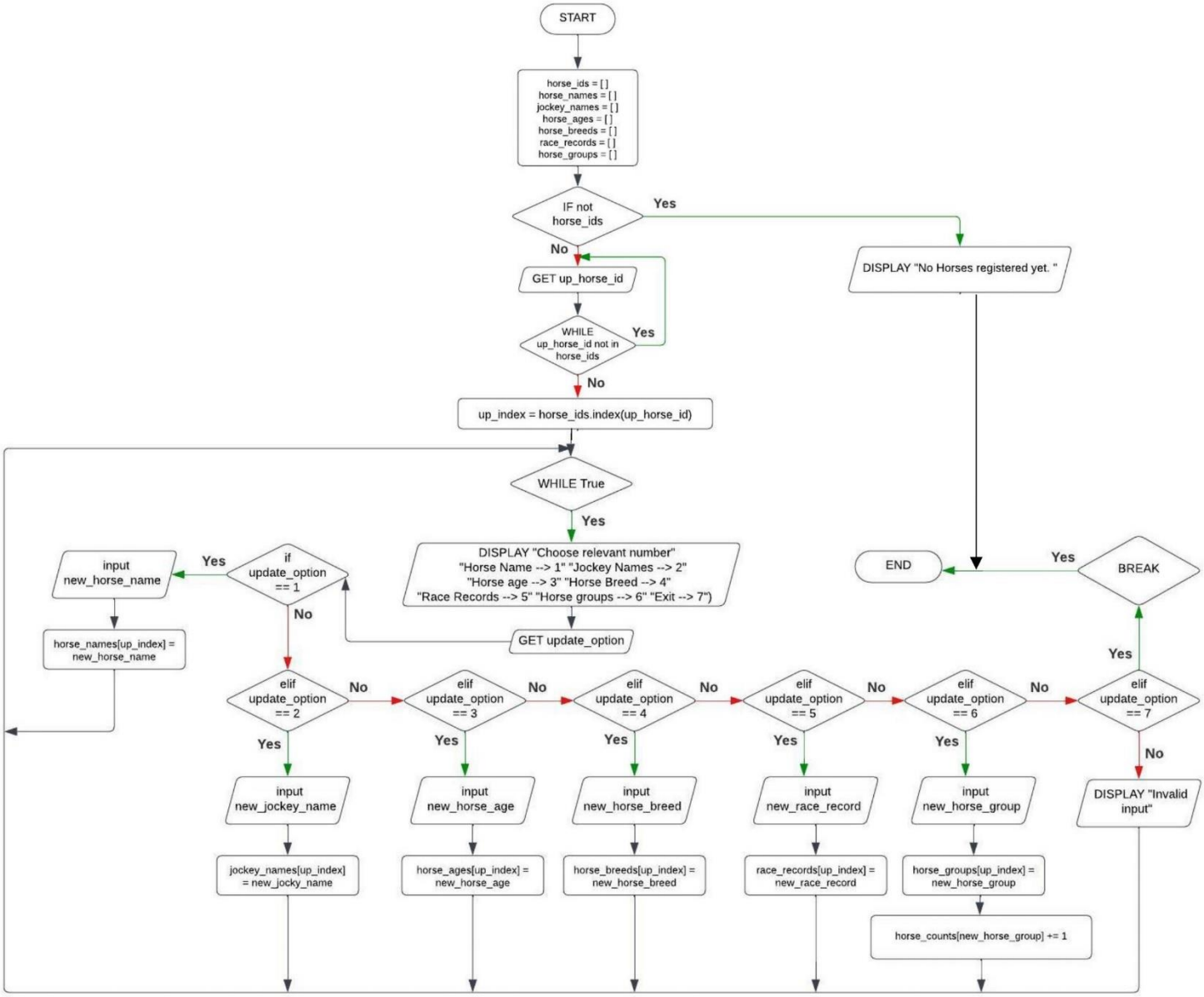

Figure 7 UHD Flowchart

UHD Function Python Code

```python
def update_horse_details():
    # Check if any horses are registered yet.
    if not horse_ids:
        print("\nNo Horses registered yet.")
        return
    # Get update horse ID as user input.
    up_horse_id = input("\nEnter the Horse ID you want to update: ")

    # Check if the entered Horse ID exists in the list.
    if up_horse_id in horse_ids:
        # Find the respective index for the given horse ID.
        up_index = horse_ids.index(up_horse_id)
        # Display a command line menu for user to choose an option.
        while True:
            print("\nChoose relevant number")
            print("Horse Name --> 1")
            print("Jockey Names --> 2")
            print("Horse age --> 3")
            print("Horse Breed --> 4")
            print("Race Records --> 5")
            print("Horse groups --> 6")
            print("Exit --> 7")

            while True:
                try:
                    # Get user input according to the menu.
                    update_option = int(input("\nSelect the option you want: "))
                    break
                except:
                    print("Invalid Input. Try again")
            # Check if is the user input equals to 1.
            if update_option == 1:
                # Get user input and assign new horse name to a variable.
                new_horse_name = input("Enter Horse name: ")
                # Update the new horse name with the respective index.
                horse_names[up_index] = new_horse_name
                print("\nHorse name changed successfully")
            # Check if is the user input equals to 2.
            elif update_option == 2:
                # Get user input and assign new jockey name to a variable.
                new_jockey_name = input("Enter Jockey name: ")
                # Update the new jockey name with the respective index.
                jockey_names[up_index] = new_jockey_name
                print("\nJockey name changed successfully")
            # Check if is the user input equals to 3.
            elif update_option == 3:
                while True:
                    # Validate new horse age in correct data type.
                    try:
                        # Get user input and assign new horse age to a variable.
                        new_horse_age = int(input("Enter Horse age: "))
                        # Update the new horse age with the respective index.
                        horse_ages[up_index] = new_horse_age
                        print("\nHorse age changed successfully")
                        break
```

```
                except:

        print("\nInvalid Input. Try again")
            # Check if is the user input equals to 4.
            elif update_option == 4:
                # Get user input and assign new horse breed to a variable.
                new_horse_breed = input("Enter Horse breed: ")
                # Update the new horse breed with the respective index.
                horse_breeds[up_index] = new_horse_breed
                print("\nHorse breed changed successfully")
            # Check if is the user input equals to 5.
            elif update_option == 5:
                # Validate wins and races compete in correct data type.
                while True:
                    try:
                        # Get user input and assign new wins count to a variable.
                        new_win = int(input("Number of Wins: "))
                        break
                    except:
                        print("Invalid input")
                while True:
                    try:
                        # Get user input and assign new races compete count to a
variable.
                        new_races = int(input("Number of races compete: "))
                        break
                    except:
                        print("Invalid input")
                # Assign new race records to a variable.
                new_race_record = f"{new_win} wins in {new_races} races"
                # Update the new race records with the respective index.
                race_records[up_index] = new_race_record
                print("\nRace record changed successfully")
            # Check if is the user input equals to 6.
            elif update_option == 6:
                # Get user input and assign new horse group to a variable.
                new_horse_group = input("Enter Horse Group (A, B, C, or D): ").upper()
                # Validate horse group.
                while new_horse_group not in ["A", "B", "C", "D"]:
                    new_horse_group = input("Invalid group. Enter Horse Group (A, B, C,
or D): ").upper()
                # checking whether one group have only 5 horses.
                if horse_counts[new_horse_group] < 5:
                    # Update horse counts for the old group.
                    horse_counts[horse_groups[up_index]] -= 1
                    print("Horse added to group", new_horse_group)
                    # Add horse to the relevant group.
                    horse_groups[up_index] = new_horse_group
                    print("\nHorse group changed successfully")
                    # Update horse counts for the respective group.
                    horse_counts[new_horse_group] += 1
                # Checking whether other groups have enough space to insert horse
details.
                elif horse_counts['A'] < 5 or horse_counts['B'] < 5 or horse_counts['C']
< 5 or horse_counts['D'] < 5:
                    print(f"Cannot add horse to Group {new_horse_group}. Group is full.")
                    # Show user which group have enough space to add horse
                    print(horse_counts.items())
                    print("Try another group")
                else:
```

```
                # When all slots are filled display this message.
                print("You can't change the group. All slots are filled.")
        # Check if is the user input equals to 7.
        elif update_option == 7:
            break
        else:
            print("Invalid Input. Try again")
else:
    print("Invalid Horse ID. Please enter valid Horse ID.")
```

UHD Function Outputs

```
Select the option you want: uhd

Is race started? (Yes or No) no

No Horses registered yet.
```

Figure 8 UHD output when horses are not registered yet.

```
Select the option you want: UHD

Is race started? (Yes or No) no

Enter the Horse ID you want to update: 123

Choose relevant number
Horse Name --> 1
Jockey Names --> 2
Horse age --> 3
Horse Breed --> 4
Race Records --> 5
Horse groups --> 6
Exit --> 7

Select the option you want: 1
Enter Horse name: Brando

Horse name changed successfully
```

Figure 9 UHD output Change Horse name

```
Select the option you want: 4
Enter Horse breed: Aribian

Horse breed changed successfully
```

Figure 16 Update horse breed

```
Select the option you want: 6
Enter Horse Group (A, B, C, or D): B
Horse added to group B

Horse group changed successfully
```

Figure 11 Update horse group

```
dict_items([('A', 1), ('B', 0), ('C', 0), ('D', 0)])
```

Figure 12 Before updating horse group, Horse counts

```
dict_items([('A', 0), ('B', 1), ('C', 0), ('D', 0)])
```

Figure 13 After updating horse group, Horse count

```
Horse Details Table:
-------------------------------------------------------------------------------------------------------------
| Horse ID  | Horse Name          | Jockey Name         | Horse Age | Horse Breed     | Race Record         | Horse Group |
-------------------------------------------------------------------------------------------------------------
| 123       | Spirit              | Sam                 | 4         | Mustang         | 4 wins in 12 races  | A           |
-------------------------------------------------------------------------------------------------------------
```

Figure 15 Before updating horse details

```
Horse Details Table:
-------------------------------------------------------------------------------------------------------------
| Horse ID  | Horse Name          | Jockey Name         | Horse Age | Horse Breed     | Race Record         | Horse Group |
-------------------------------------------------------------------------------------------------------------
| 123       | Brando              | Sam                 | 4         | Aribian         | 4 wins in 12 races  | B           |
-------------------------------------------------------------------------------------------------------------
```

Figure 14 After updating horse details

```
Select the option you want: uhd

Is race started? (Yes or No) no

Enter the Horse ID you want to update: 321
Invalid Horse ID. Please enter valid Horse ID.
```

Figure 10 When enter an Invalid horse ID

UHD Function Description

The user can update details of registered horses with this function. After checking whether any horses are registered, it asks the user to input the horse ID they wish to change details.

If the entered horse ID exists in the horse_ids list, the function shows a menu with options to edit details like Horse name, Jockey name, horse age, horse breed, race records, and horse group. By inputting the matching number, the user can select the appropriate choice.

- Option 1: Updates the horse's name.
- Option 2: Updates the jockey's name.
- Option 3: Updates the horse's age.
- Option 4: Updates the horse's breed.
- Option 5: Updates the race records, including the number of wins and races competed.
- Option 6: Updates the horse's group, considering group size constraints.
- Option 7: Exits the update process.

The function checks whether the entered values are in the correct data type and handle invalid inputs correctly. If the user tries to move the horse to a different group, it updates the horse counts based on whether the new group has spaces available. If the entered horse ID not found, the function notifies the user to enter valid Horse ID.

5. VHD-View the registered horses' details table

Sorting Algorithm Flowchart

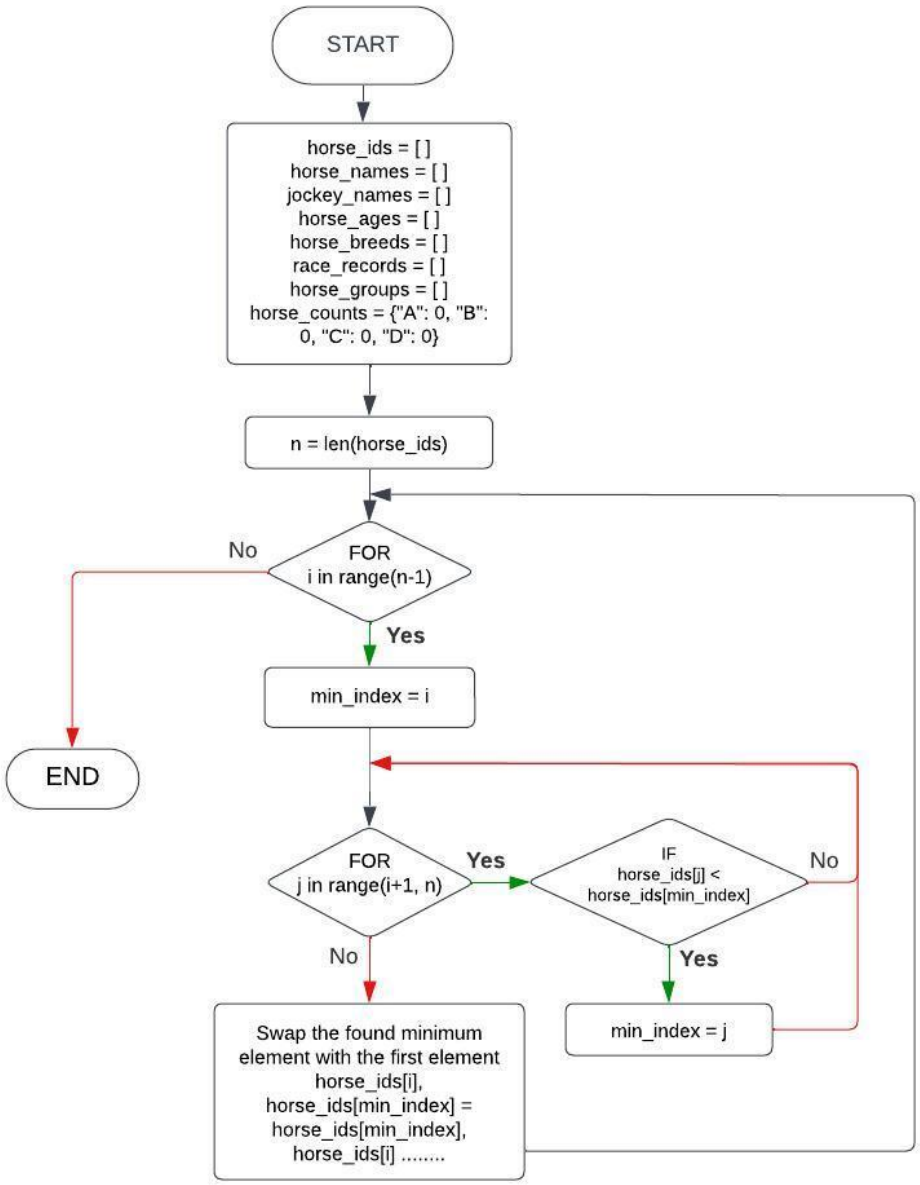

Figure 17 Sorting algorithm Flowchart

VHD Function Flowchart

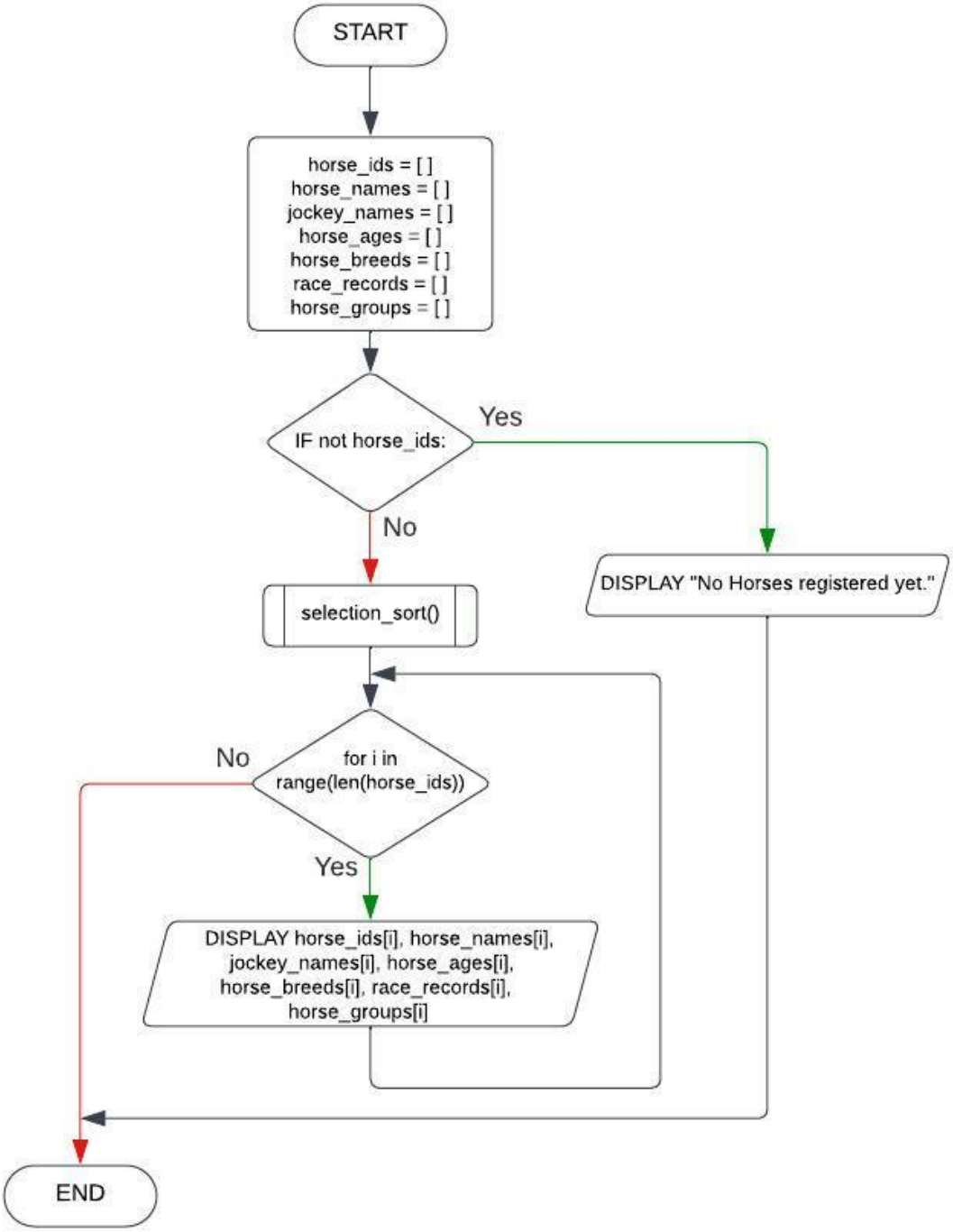

Figure 18 VHD Flowchart

Sorting Algorithm Python Code

```
def selection_sort(horse_ids, horse_names, jockey_names, horse_ages, horse_breeds,
race_records, horse_groups):
    # Sorting algorithm according to ascending order of horse ids.
    n = len(horse_ids)
    for i in range(n - 1):
        # Assign minimum index to i.
        min_index = i
        for j in range(i + 1, n):
            # Convert horse IDs to integers for numerical comparison.
            if int(horse_ids[j]) < int(horse_ids[min_index]):
                # if horse_ids[j]<horse_ids[i] then assign minimum index to j.
                min_index = j

        # Swap the found minimum element with the first element.
        horse_ids[i], horse_ids[min_index] = horse_ids[min_index], horse_ids[i]
        horse_names[i], horse_names[min_index] = horse_names[min_index], horse_names[i]
        jockey_names[i], jockey_names[min_index] = jockey_names[min_index],
jockey_names[i]
        horse_ages[i], horse_ages[min_index] = horse_ages[min_index], horse_ages[i]
        horse_breeds[i], horse_breeds[min_index] = horse_breeds[min_index],
horse_breeds[i]
        race_records[i], race_records[min_index] = race_records[min_index],
race_records[i]
        horse_groups[i], horse_groups[min_index] = horse_groups[min_index],
horse_groups[i]
```

VHD Function Python Code

```
def view_horse_details():
    # Check if any horses are registered.
    if not horse_ids:
        print("\nNo Horses registered yet.")
        return

    # Call sorting algorithm.
    selection_sort(horse_ids, horse_names, jockey_names, horse_ages, horse_breeds,
race_records, horse_groups)

    # Display horse details table.
    print("\nHorse Details Table:")
    print("-" * 126)
    # Display column titles.
    print("| {:<9} | {:<20} | {:<20} | {:<9} | {:<15} | {:<20} | {:<11} |".format("Horse
ID", "Horse Name", "Jockey Name", "Horse Age", "Horse Breed", "Race Record", "Horse
Group"))
    print("-" * 126)

    for i in range(len(horse_ids)):
        # Display horse records according to the given format.
        print("| {:<9} | {:<20} | {:<20} | {:<9} | {:<15} | {:<20} | {:<11}
|".format(horse_ids[i], horse_names[i], jockey_names[i], horse_ages[i], horse_breeds[i],
race_records[i], horse_groups[i]))

    print("-" * 126)
```

VHD Function Output

Horse Details Table:

Horse ID	Horse Name	Jockey Name	Horse Age	Horse Breed	Race Record	Horse Group
123	Spirit	Sam	4	Mustang	4 wins in 12 races	A
124	Thunderbolt	Emily	5	Arabian	3 wins in 10 races	B
125	Midnight Star	Alex	6	Thoroughbred	7 wins in 15 races	A
127	Blaze	Lily	3	Paint	2 wins in 8 races	B
129	Luna	Ethan	5	Friesian	6 wins in 14 races	A
130	Velvet Dream	Noah	6	Morgan	8 wins in 18 races	C
132	Copper Charm	Oliver	4	Morgan	5 wins in 13 races	A
137	Ebony Majesty	Grace	4	Shetland Pony	4 wins in 5 races	C
138	Wind	Aiden	5	Morgan	0 wins in 3 races	C
151	Stormy Knight	Mia	5	Mustang	3 wins in 9 races	B
165	Silver Shadow	Jake	4	Appaloosa	5 wins in 11 races	C
173	Dancer	Liam	5	Thoroughbred	0 wins in 0 races	A
174	Thunder	Kasun	4	Appaloosa	4 wins in 14 races	C
185	Willow Breeze	Sophia	3	Welsh Pony	3 wins in 10 races	D
287	Serenade	Jin	3	Friesian	2 wins in 10 races	D
321	Brand	Jimmy	5	Aribian	2 wins in 10 races	B
345	Maverick	Ava	4	Quarter Horse	4 wins in 12 races	D
387	Diamond Dust	Isabella	4	Shetland Pony	1 wins in 5 races	B
473	Stardust	Zoey	3	Arabian	3 wins in 10 races	D
543	Rustic Rose	Jim	6	Mustang	7 wins in 12 races	D

Figure 19 VHD output

VHD Function Description

Sorting Algorithm: Based on their Horse IDs, the registered horse' details are sorted in ascending order using selection sort algorithm. This iterates through the list of Horse IDs and find the minimum element and swapping it with the first unsorted element in each turn. Until the entire list is sorted, this process is repeated.

After sorting algorithm VHD function displays the registered horses in a table. It notifies the user that there are no horses to display if any horse not registered yet. If horses are registered, it calls the selection_sort function to ensure that the information is arranged in ascending order based on Horse IDs.

Then prints a table with column titles, including Horse ID, Horse Name, Jockey Name, Horse Age, Horse Breed, Race Record, and Horse Group. It iterates through sorted lists and display each horse's details.

6. SHD- Save the horse details to the text file

SHD Function Flowchart

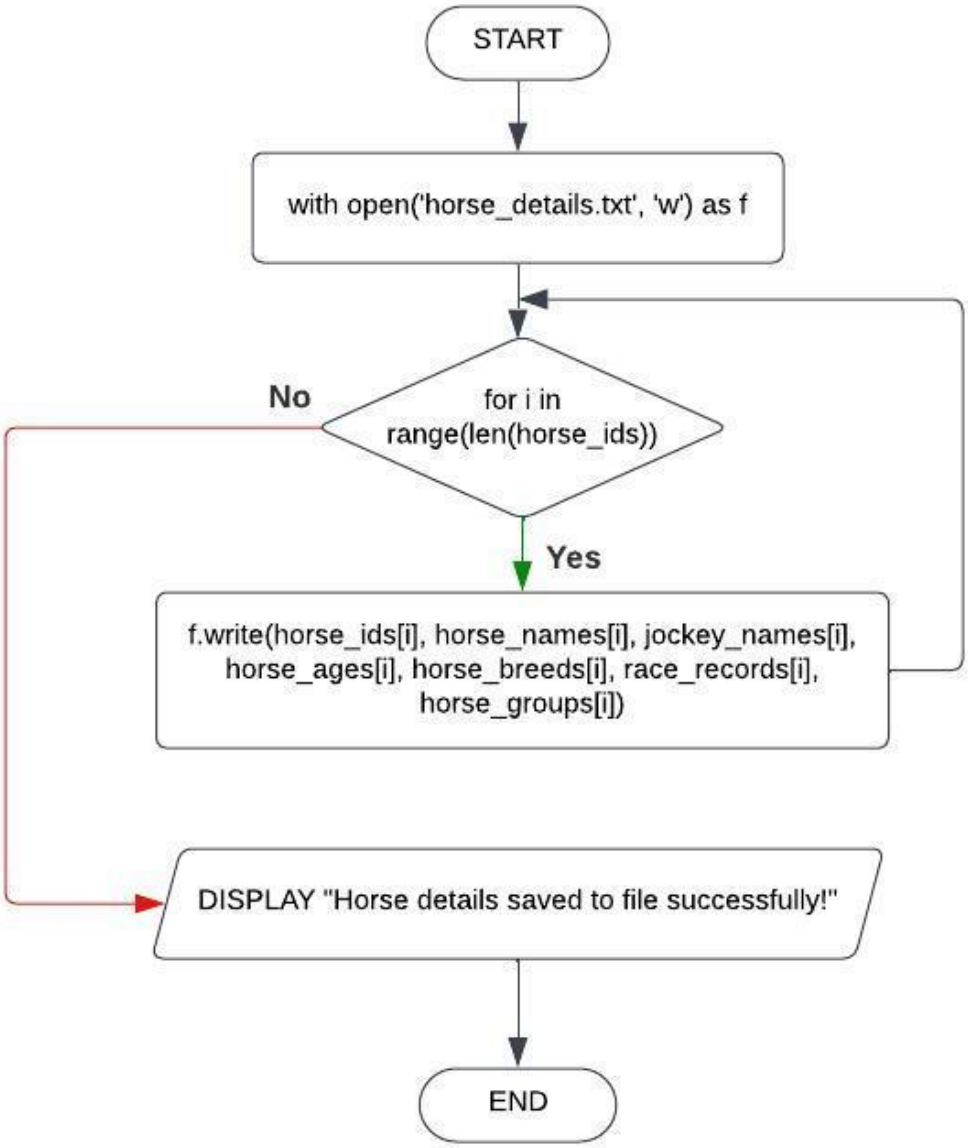

Figure 20 SHD Flowchart

SHD Function Python Code

```
def save_horse_details():
    # Open the file in write mode.
    with open('horse_details.txt', 'w') as f:
        # Display horse details table.
        f.write("\nHorse Details Table:")
        f.write(" ")
        f.write("\n| {:<9} | {:<20} | {:<20} | {:<9} | {:<15} | {:<20} | {:<11}
|".format("Horse ID", "Horse Name", "Jockey Name", "Horse Age", "Horse Breed", "Race
Record", "Horse Group"))
        # Iterate through horse details based on horse IDs and write to the file.
        for i in range(len(horse_ids)):
            f.write("\n| {:<9} | {:<20} | {:<20} | {:<9} | {:<15} | {:<20} | {:<11}
|".format(horse_ids[i], horse_names[i], jockey_names[i], horse_ages[i], horse_breeds[i],
race_records[i], horse_groups[i]))
    print("Horse details saved to file successfully!")
```

SHD Function Output

```
Select the option you want: shd
Horse details saved to file successfully!
```

Figure 21 SHD output

horse_details.txt

File Edit View

```
Horse Details Table:
| Horse ID  | Horse Name          | Jockey Name         | Horse Age | Horse Breed     | Race Record         | Horse Group |
| 123       | Spirit              | Sam                 | 4         | Mustang         | 4 wins in 12 races  | A           |
| 124       | Thunderbolt         | Emily               | 5         | Arabian         | 3 wins in 10 races  | B           |
| 125       | Midnight Star       | Alex                | 6         | Thoroughbred    | 7 wins in 15 races  | A           |
| 127       | Blaze               | Lily                | 3         | Paint           | 2 wins in 8 races   | B           |
| 129       | Luna                | Ethan               | 5         | Friesian        | 6 wins in 14 races  | A           |
| 130       | Velvet Dream        | Noah                | 6         | Morgan          | 8 wins in 18 races  | C           |
| 132       | Copper Charm        | Oliver              | 4         | Morgan          | 5 wins in 13 races  | A           |
| 137       | Ebony Majesty       | Grace               | 4         | Shetland Pony   | 4 wins in 5 races   | C           |
| 138       | Wind                | Aiden               | 5         | Morgan          | 0 wins in 3 races   | C           |
| 151       | Stormy Knight       | Mia                 | 5         | Mustang         | 3 wins in 9 races   | B           |
| 165       | Silver Shadow       | Jake                | 4         | Appaloosa       | 5 wins in 11 races  | C           |
| 173       | Dancer              | Liam                | 5         | Thoroughbred    | 0 wins in 0 races   | A           |
| 174       | Thunder             | Kasun               | 4         | Appaloosa       | 4 wins in 14 races  | C           |
| 185       | Willow Breeze       | Sophia              | 3         | Welsh Pony      | 3 wins in 10 races  | D           |
| 287       | Serenade            | Jin                 | 3         | Friesian        | 2 wins in 10 races  | D           |
| 321       | Brand               | Jimmy               | 5         | Aribian         | 2 wins in 10 races  | B           |
| 345       | Maverick            | Ava                 | 4         | Quarter Horse   | 4 wins in 12 races  | D           |
| 387       | Diamond Dust        | Isabella            | 4         | Shetland Pony   | 1 wins in 5 races   | B           |
| 473       | Stardust            | Zoey                | 3         | Arabian         | 3 wins in 10 races  | D           |
| 543       | Rustic Rose         | Jim                 | 6         | Mustang         | 7 wins in 12 races  | D           |
```

Figure 22 Table created in the text file

SHD Function Description

This function saves the details of registered horses to a text file named “horse_details.txt.” It opens the file in write mode. After that writes the horse details table including Horse ID, Horse Name, Jockey Name, Horse Age, Horse Breed, Race records, and Horse Group.

This function iterates through the lists containing the details of each registered horse. After that writes the information to the file in the specified format. Finally, it prints a confirmation message informing that the horse details have been successfully saved to the file.

7. SDD- Selecting four horses randomly

Read horse details from the file Flowchart

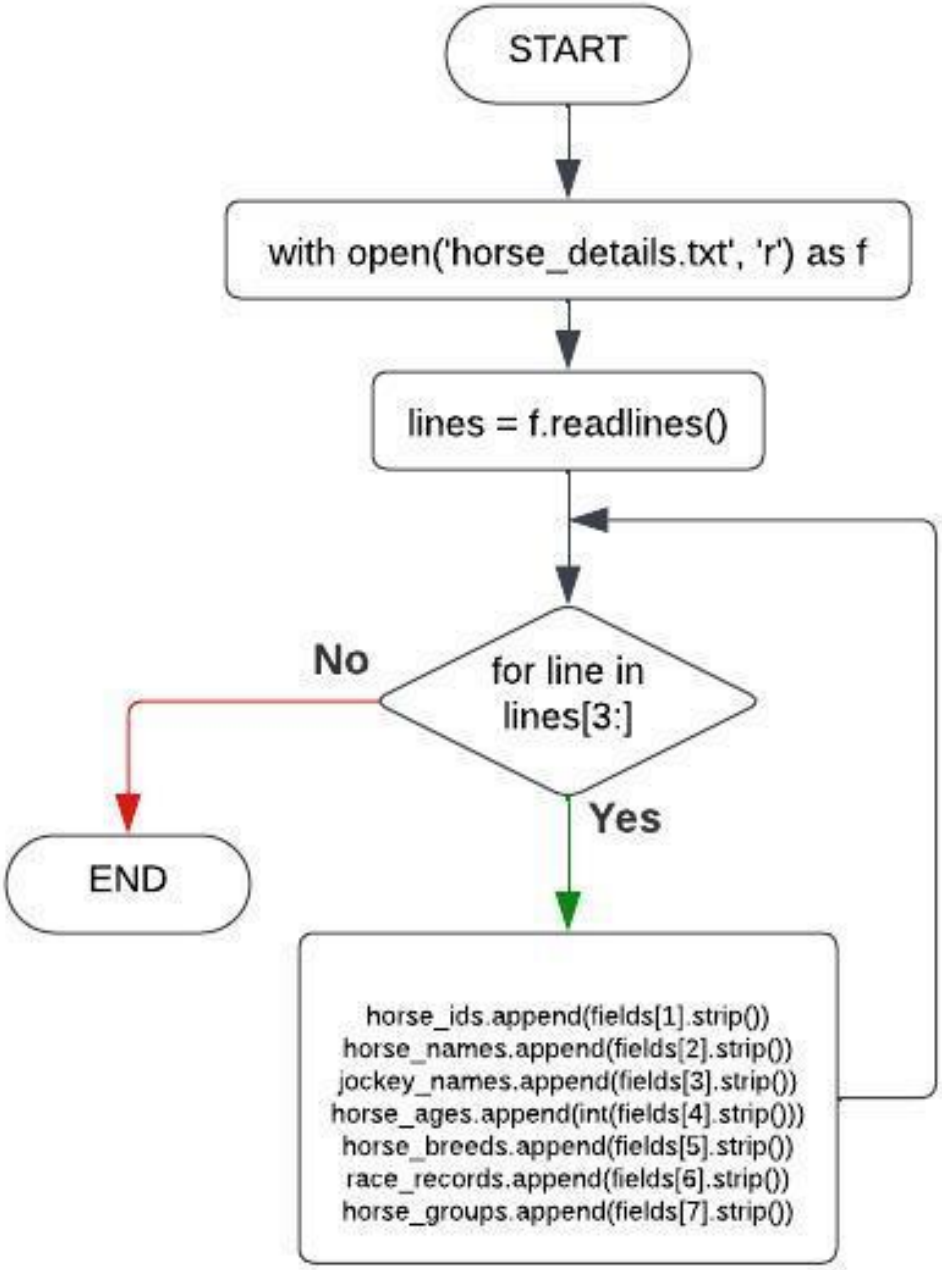

Figure 23 Read horse details from the file flowchart

SDD Function Flowchart

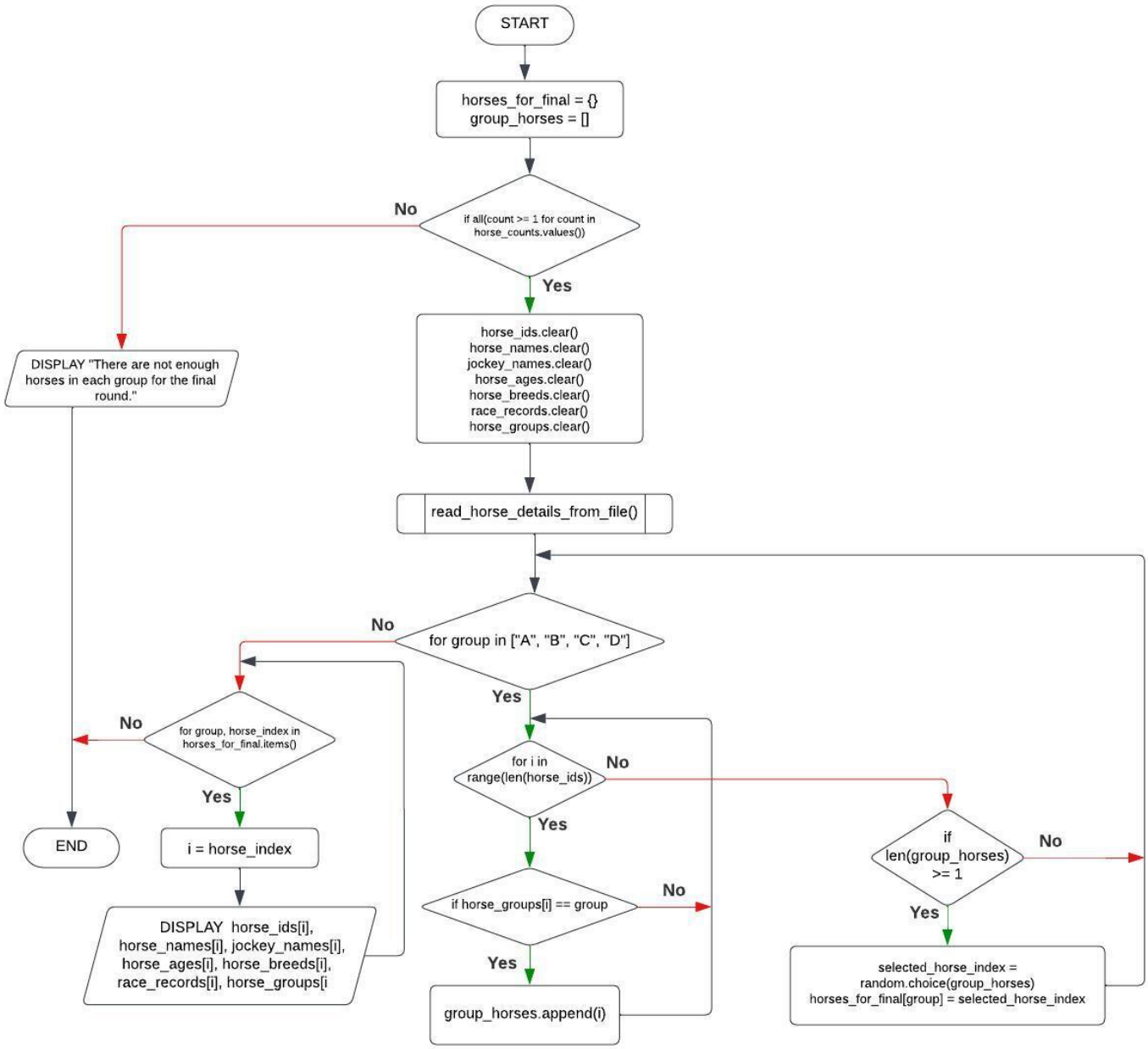

Figure 24 SDD Flowchart

Read horse details from the file Python Code

```
def read_horse_details_from_file():
    # Read horse details from the text file.
    with open('horse_details.txt', 'r') as f:
        # Reads all lines from the file and stores them in the list variable 'lines'.
        lines = f.readlines()

    # Extract horse details from the lines, skipping the header.
    for line in lines[3:]:  # Skip the header lines
        # Removes whitespaces from the string 'line' and then splits it into fields using
the '|'
        fields = line.strip().split('|')
        # Appends data from specific fields of the 'fields' list to their respective
lists.
        horse_ids.append(fields[1].strip())
        horse_names.append(fields[2].strip())
        jockey_names.append(fields[3].strip())
        try:
            horse_ages.append(int(fields[4].strip()))
        except ValueError:
            # Handle the case where the age cannot be converted to an integer
            horse_ages.append(0)
        horse_breeds.append(fields[5].strip())
        race_records.append(fields[6].strip())
        horse_groups.append(fields[7].strip())
```

SDD Function Python Code

```
def select_random_horses(horse_counts, horse_ids, horse_names, jockey_names, horse_ages,
horse_breeds, race_records, horse_groups):
    horses_for_final = {}

    # Check if there are enough horses in each group for the final round, at-least one
horse in each group.
    if all(count >= 1 for count in horse_counts.values()):
        # Clear existing data otherwise data will duplicate.
        horse_ids.clear()
        horse_names.clear()
        jockey_names.clear()
        horse_ages.clear()
        horse_breeds.clear()
        race_records.clear()
        horse_groups.clear()

        # Read horse details from the text file
        read_horse_details_from_file()

        # Select one random horse from each group
        for group in ["A", "B", "C", "D"]:
            group_horses = []

            # Collect horses in the current group
            for i in range(len(horse_ids)):
                # Checks if the horse group of the horse at index 'i' matches the current
group iterated.
```

```
                if horse_groups[i] == group:
                    # Appends the index of the horse to the 'group_horses'
                    group_horses.append(i)

            # Ensure that there is at least 1 horse in the group before sampling.
            if len(group_horses) >= 1:
                selected_horse_index = random.choice(group_horses)
                horses_for_final[group] = selected_horse_index

        # Display the details of the randomly selected horses
        print("\nRandomly Selected Horses for the Final Round:")
        print("-" * 126)
        # Display the table column headings.
        print("| {:<9} | {:<20} | {:<20} | {:<9} | {:<15} | {:<20} | {:<11}
|".format("Horse ID", "Horse Name", "Jockey Name", "Horse Age", "Horse Breed", "Race
Record", "Horse Group"))
        print("-" * 126)

        # Iterates through the items  in the horses_for_final.
        for group, horse_index in horses_for_final.items():
            # Assigns the stored index
            i = horse_index
            # Display horse details at the given index
            print("| {:<9} | {:<20} | {:<20} | {:<9} | {:<15} | {:<20} | {:<11}
|".format(horse_ids[i], horse_names[i], jockey_names[i], horse_ages[i], horse_breeds[i],
race_records[i], horse_groups[i]))

        print("-" * 126)
    else:
        # Display this message when there are not enough horses in each group.
        print("There are not enough horses in each group for the final round.")

    # Returns the populated 'horses_for_final' dictionary.
    # Ready for further processing.
    return horses_for_final
```

SDD Function Output

```
Select the option you want: sdd

Randomly Selected Horses for the Final Round:
-------------------------------------------------------------------------------------------------------------------------
| Horse ID  | Horse Name           | Jockey Name          | Horse Age | Horse Breed     | Race Record          | Horse Group |
-------------------------------------------------------------------------------------------------------------------------
| 125       | Midnight Star        | Alex                 | 6         | Thoroughbred    | 7 wins in 15 races   | A           |
| 124       | Thunderbolt          | Emily                | 5         | Arabian         | 3 wins in 10 races   | B           |
| 138       | Wind                 | Aiden                | 5         | Morgan          | 0 wins in 3 races    | C           |
| 473       | Stardust             | Zoey                 | 3         | Arabian         | 3 wins in 10 races   | D           |
-------------------------------------------------------------------------------------------------------------------------
```

Figure 25 SDD Output

SDD Function Description

Read Horse Details from File: This function reads horse details from the text file. It opens the file in read mode. It then iterates through the lines, starting from the fourth line (skipping the header), and extract fields to populate the lists containing horse details.

This function selects one random horse from each group for the final round. First it checks if there are enough horses in each group. (At least one horse in each group). If the condition met, clear existing data to avoid data duplication. After that reads horse details from the file using the read_horse_details_from_file function.

Then select one random horse from each group. Finally display them in a table. If there are not enough horses in each group for the final round, it prompts a message for the user.

8. WHD- Display Winning horses

Time Sorting Algorithm Flowchart

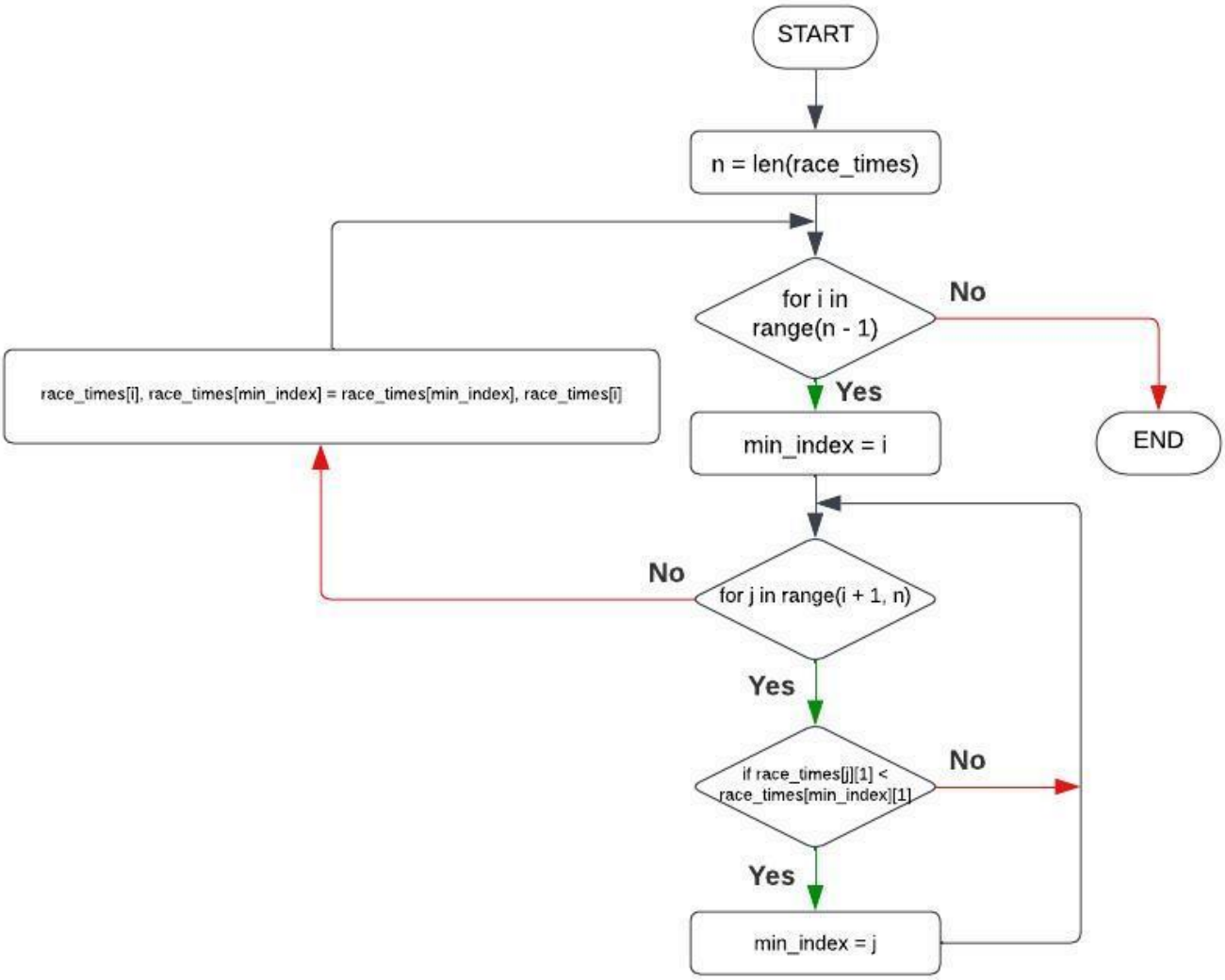

Figure 26 Time sort algorithm flowchart

WHD Flowchart

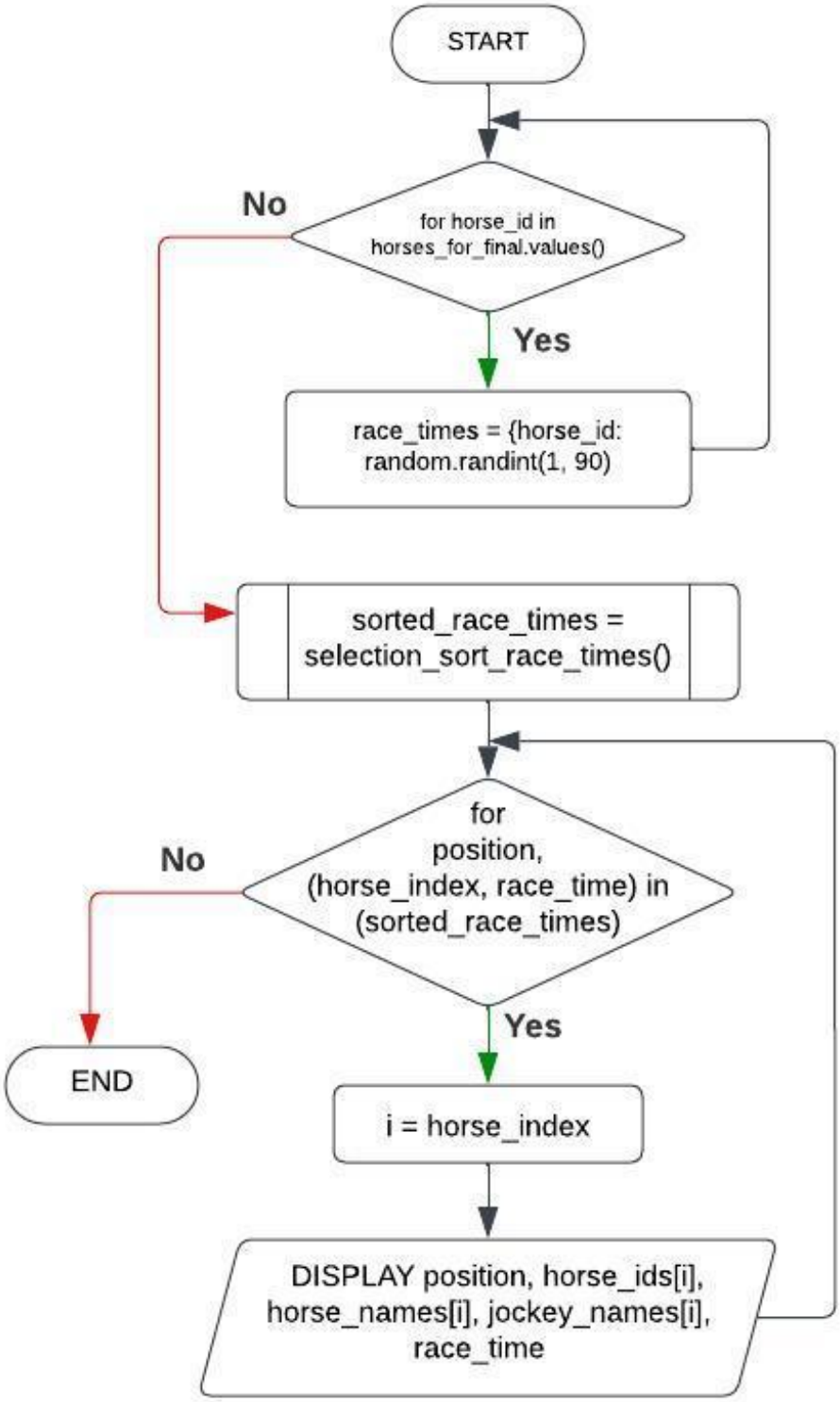

Figure 27 WHD flowchart

Time Sorting Algorithm Python Code

```
def selection_sort_race_times(race_times):
    n = len(race_times)
    # Iterate through the list
    for i in range(n - 1):
        min_index = i
        # Find the minimum element in the unsorted part of the list
        for j in range(i + 1, n):
            if race_times[j][1] < race_times[min_index][1]:
                min_index = j

        # Swap the found minimum element with the first element
        race_times[i], race_times[min_index] = race_times[min_index], race_times[i]

    # Return the sorted list
    return race_times
```

WHD Python Code

```
def display_winning_horses(horses_for_final, horse_ids, horse_names, jockey_names):
    # Simulate the race and assign random times to each horse
    race_times = {horse_id: random.randint(1, 90) for horse_id in
horses_for_final.values()}

    # Sort race_times based on race times
    sorted_race_times = selection_sort_race_times(list(race_times.items()))

    # Display the details of the winning horses
    print("\nWinning Horses:")
    print("-" * 126)
    print("| {:<9} | {:<20} | {:<20} | {:<9} | {:<9} |".format("Position", "Horse ID",
"Horse Name", "Jockey Name", "Race Time"))
    print("-" * 126)

    for position, (horse_index, race_time) in enumerate(sorted_race_times[:4], start=1):
# Display positions from 1 to 4
        i = horse_index
        print("| {:<9} | {:<20} | {:<20} | {:<9} | {:<9}s |".format(position,
horse_ids[i], horse_names[i], jockey_names[i], race_time))

    print("-" * 126)

    return race_times
```

WHD Output

```
Select the option you want: whd

Winning Horses:
-------------------------------------------------------------------------------------------
| Position  | Horse ID              | Horse Name            | Jockey Name | Race Time |
-------------------------------------------------------------------------------------------
| 1         | 138                   | Wind                  | Aiden       | 21        s |
| 2         | 124                   | Thunderbolt           | Emily       | 24        s |
| 3         | 125                   | Midnight Star         | Alex        | 29        s |
| 4         | 473                   | Stardust              | Zoey        | 54        s |
-------------------------------------------------------------------------------------------
```

Figure 28 WHD Output

WHD Description

Selection Sort for Race Times: This function uses algorithm to sort the list ascendingly based on race times. The race times are represented as tuples. Each tuple contains a horse ID and its corresponding race time. The function takes race_times list a parameter.

In each run, the algorithm iterates through the list of race timings, identifying the smallest element based on the race time and swapping it with the first unsorted element. This process goes through until the full list has been sorted based on race times.

WHD function replicates a race by assigning random times to each finalist horse. The times are then sorted using the selection_sort_race_times. After that display the details of the winning horses in a table.

The race times are generated as a dictionary, where each horse ID is mapped to a randomly generated race time between 1 and 90 seconds. The details of the top 4 winning horses are displayed, including their positions. The function returns the dictionary of race times for further processing.

9. VWH- Visualize Winning horses

VWH Flowchart

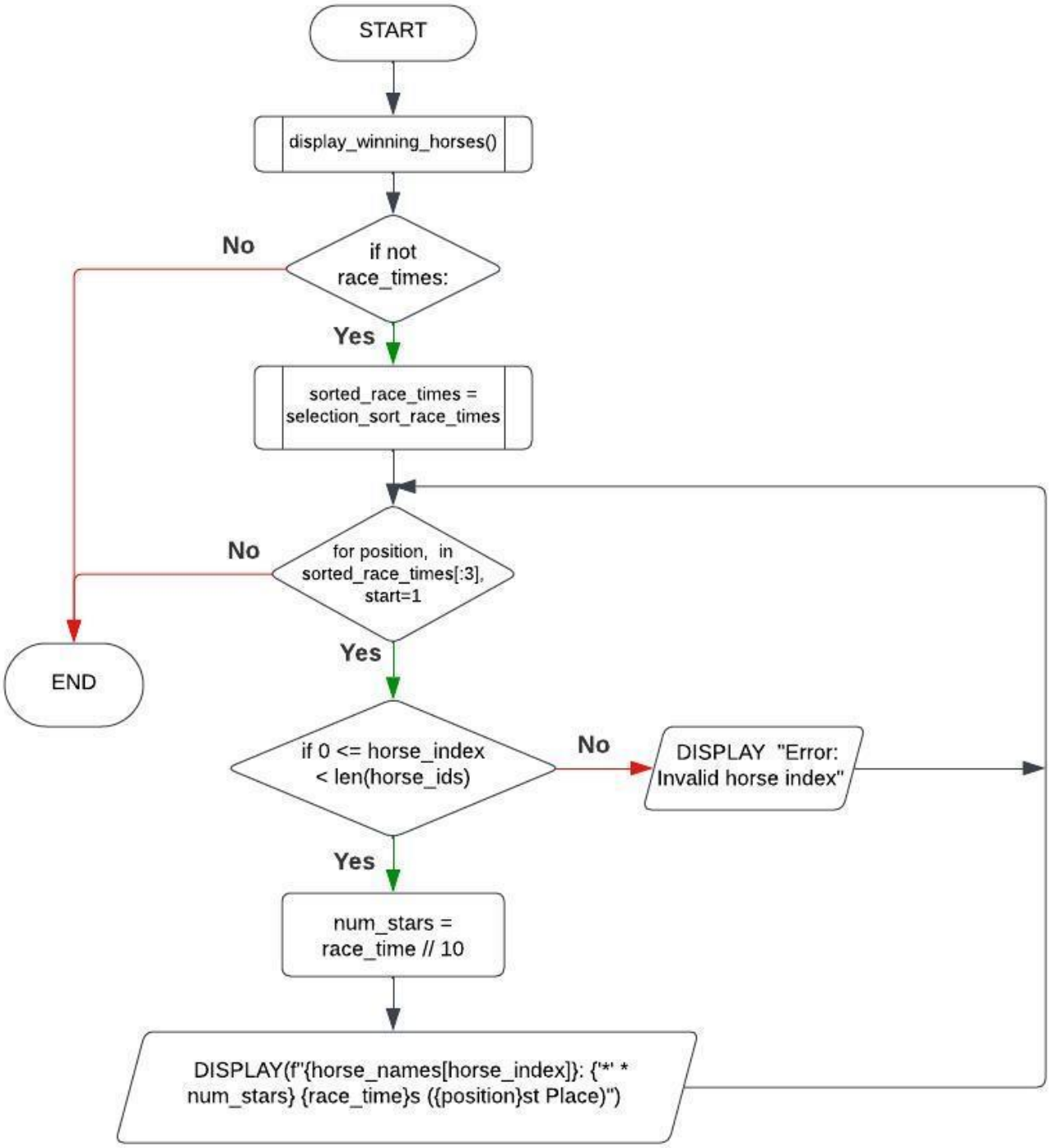

Figure 29 VWH Flowcharts

VWH Python Code

```
def visualize_winning_horses():
    global race_times

    # Check if the race has been simulated and winning times are available
    if not race_times:
        print("Race simulation has not been conducted. Please run the 'WHD' command
first.")
        return

    # Use selection sort for race_times based on race times
    sorted_race_times = selection_sort_race_times(list(race_times.items()))

    # Display the details of the winning horses with visualization
    print("\nVisualizing Winning Horses:")
    print("-" * 126)

    for position, (horse_index, race_time) in enumerate(sorted_race_times[:3], start=1):
# Display positions from 1 to 3
        # Check if horse_index is within the valid range
        if 0 <= horse_index < len(horse_ids):
            # Calculate the number of '*' based on 10s intervals
            num_stars = race_time // 10
            # Display the details of the winning horses
            print(f"{horse_names[horse_index]}: {'*' * num_stars} {race_time}s
({position}st Place)")
        else:
            print(f"Error: Invalid horse index {horse_index}.")

    print("-" * 126)
```

VWH Output

```
Select the option you want: vwh

Visualizing Winning Horses:
-----------------------------------------
Wind: ** 21s (1st Place)
Thunderbolt: ** 24s (2st Place)
Midnight Star: ** 29s (3st Place)
-----------------------------------------
```

Figure 30 VWH Output

VWH Description

This function visualizes the top 3 winning horses based on their race times. It checks if the race has been started and winning times are available. After the race times are sorted and each winning horse's name is displayed, along with a visual representation. Here their race times represents using asterisks("*"). The system visualized the time by One * for 10s.

10. Race Details

Race Details Flowcharts

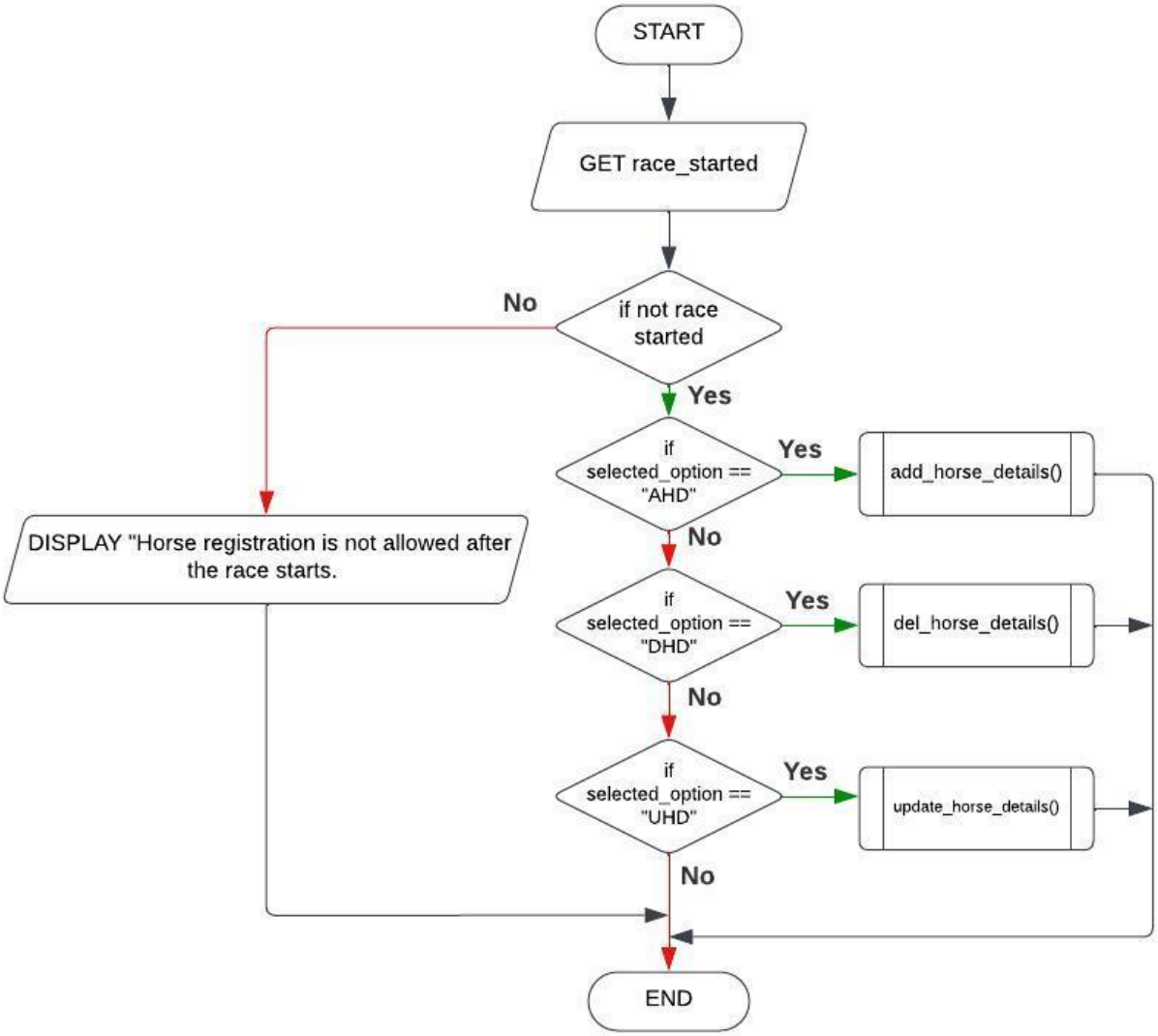

Figure 31 Race Details Flowchart

Race Details Python Code

```
def race_details():
    while True:
        try:
            # Get user input whether race started.
            race_started = input("\nIs race started? (Yes or No) ").lower()
            # Convert input to lowercase for case sensitivity.
            if race_started not in ["yes", "no"]:
                raise ValueError("Your answer should be Yes or No")
            race_started = race_started == "yes"  # Convert the input to  yes or no
            break
        except:
            print("Invalid input.")

    # Allow horse registration only if race hasn't started
    if not race_started:
        if selected_option == "AHD":
            add_horse_details()
        elif selected_option == "DHD":
            del_horse_details()
        elif selected_option == "UHD":
            update_horse_details()
    else:
        # If race started display this message.
        print("Horse registration is not allowed after the race starts.")
```

Race Details Function Description

This function manages race-related details, such as checking whether or not race has begun and enabling horse registration only if the race has begun. Its askes user to input whether the race has started. If the input is not either yes or no, it raises a ValueError with a message. If the race not started yet, it checks selection option and calls the respective function. Functions are AHD for adding horse details, DHD for deleting horse details, UHD for updating horse details.

11. Console menu system

Console menu system Flowchart

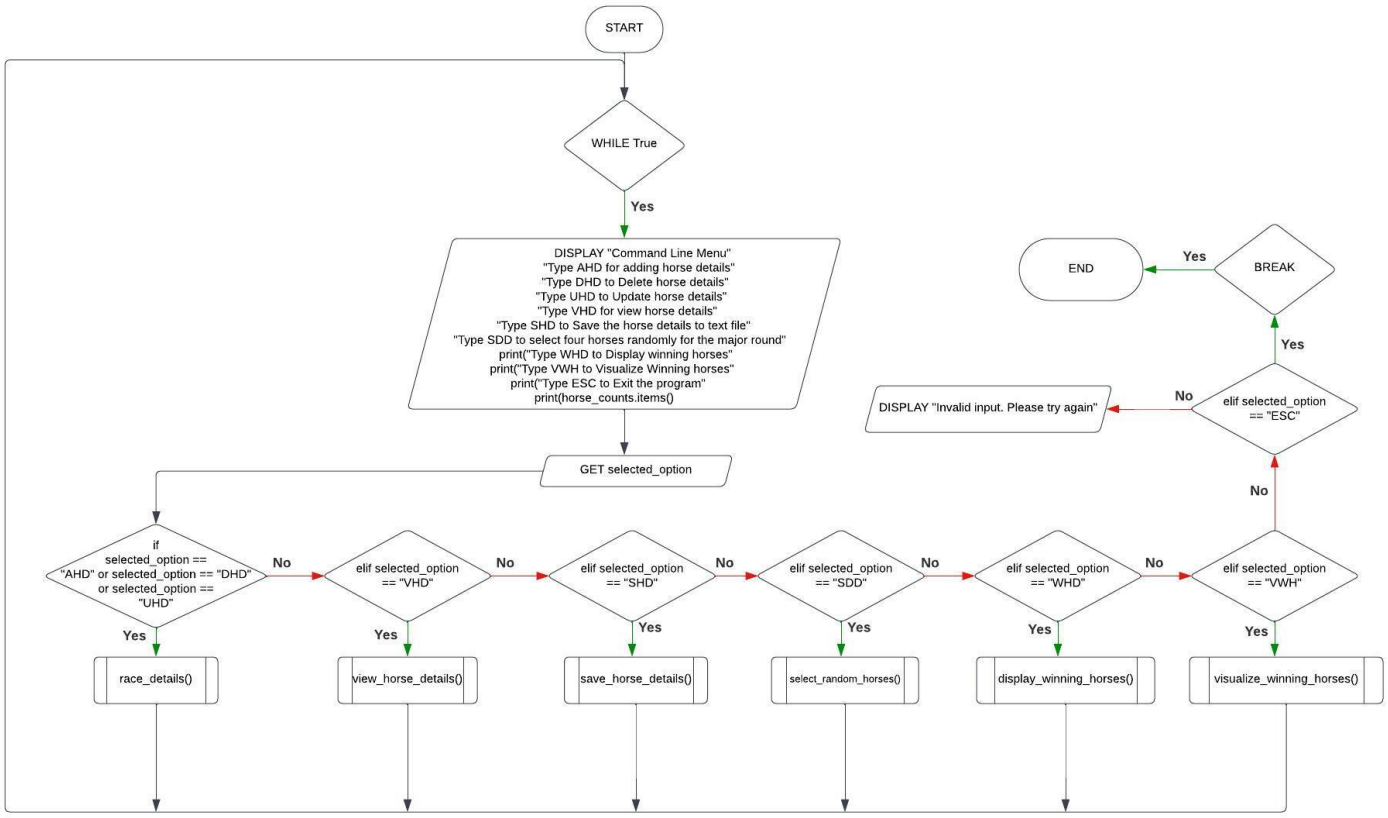

Figure 32 Console menu system flowchart

Console menu system Python Code

```
data_saved = False
final_selected = False
race_ended = False

while True:
    time.sleep(1)
    print("\nCommand Line Menu")
    print("Type AHD for adding horse details")
    print("Type UHD for Update horse details")
    print("Type DHD for deleting horse details")
    print("Type VHD for view horse details")
    print("Type SHD for Save the horse details to text file")
    print("Type SDD for select four horses randomly for the major round")
    print("Type WHD for Display winning horses")
    print("Type VWH for Visualize Winning horses")
    print("Type ESC to Exit the program")
    print(horse_counts.items())
```

```
    # Get user input for selected option.
    selected_option = input("Select the option you want: ").upper()

    # Handle different menu options.
    if selected_option == "AHD" or selected_option == "DHD" or selected_option == "UHD":
        # Call race_details() function for adding, deleting, or updating horse details.
        race_details()
    elif selected_option == "VHD":
        # Call view_horse_details() function for viewing horse details.
        view_horse_details()
    elif selected_option == "SHD":
        # call save_horse_details() funtion for save horse details.
        save_horse_details()
        data_saved = True
    elif selected_option == "SDD":
        # Check if data is saved before selecting horses for the major round.
        if not data_saved:
            print("\nPlease save the data first using SHD")
        else:
            # Call select_random_horses() function to select four horses randomly.
            horses_for_final = select_random_horses(horse_counts, horse_ids, horse_names,
jockey_names, horse_ages, horse_breeds, race_records, horse_groups)
            final_selected = True
    elif selected_option == "WHD":
        # Check if data is saved and horses are selected for the major round before
displaying winning horses.
        if not data_saved:
            print("\nPlease save the data first using SHD")
        elif not final_selected:
            print("\nPlease select final round horses using SDD")
        else:
            # Call display_winning_horses() function to display winning horses.
            race_times = display_winning_horses(horses_for_final, horse_ids, horse_names,
jockey_names)
            race_ended = True
    elif selected_option == "VWH":
        # Check if data is saved and the race has ended before visualizing winning
horses.
        if not data_saved:
            print("\nPlease save the data first using SHD")
        elif not race_ended:
            print("\nRace didn't start yet. To start the race use WHD")
        else:
            # Call visualize_winning_horses() function to visualize winning horses.
            visualize_winning_horses()
    elif selected_option == "ESC":
        # Exit the program if the user chooses to exit.
        print("Exiting the program")
        break
    else:
        # Print an error message for invalid input.
        print("Invalid input. Please try again.")
```

Console Menu System Output

```
Command Line Menu
Type AHD for adding horse details
Type UHD for Update horse details
Type DHD for deleting horse details
Type VHD for view horse details
Type SHD for Save the horse details to text file
Type SDD for select four horses randomly for the major round
Type WHD for Display winning horses
Type VWH for Visualize Winning horses
Type ESC to Exit the program
dict_items([('A', 5), ('B', 5), ('C', 5), ('D', 5)])
Select the option you want:
```

Figure 33 Console Menu System Output

Console Menu System Description

This code creates a command-line menu for horse race application. The program runs in loop, asking user for input, displaying a menu. Different options run corresponding functions. The program continuously displays the menu. The loop continuous until the user chooses to exit by entering "ESC."

Key Functions in the menu,

- race_details() : Allows horse registration. (Add, delete, and update horse details)
- view_horse_details() : Display horse details.
- save_horse_details() : Save horse details to a text file.
- select_random_horses() : Select 4 horses randomly for the final round.
- display_winning_horses() : Display the details of winning horses.
- visualize_winning_horses() : Visualize the winning horses.

12. References

Anon, 2019. *Python.org.* [Online]
Available at: https://docs.python.org/3/tutorial/index.html
[Accessed December 2023].

Barry, P., 2023. *Head First Python.* 3rd ed. s.l.:O'reilly.

Learn, T., 2019. *Codecademy.* [Online]
Available at: https://www.codecademy.com/learn/learn-python
[Accessed November 2023].

W3Schools, 2019. *W3schools.com.* [Online]
Available at: https://www.w3schools.com/python/
[Accessed December 2023].

WebsiteSetup, 2021. *Python Cheat Sheet ,* s.l.: s.n.