

**Project Report On**

**Olympic Records Management System**

**Course:**

# PROGRAMMING IN PYTHON

**Course Code: CAP776**

**Submitted by Submitted to**

**Name:** Ritik Singh Chauhan Dr. Girish Kumar

**Reg no:** 12325662

**Roll no:** 21

**Lovely Professional University Punjab**

**2024-25 (Term 3)**

# ACKNOWLEDGEMENT

I have taken efforts in this project. however, it would not have been possible without the kind support and help of many individuals. I would like to extend my sincere thanks to all of them. It has been great to learn Python Programming from Lovely Professional University.

I am highly indebted to Dr. Girish Kumar, for his guidance and constant supervision as well as for providing necessary information regarding the project and for their support in completing the project. Their constant guidance and willingness to share vast knowledge made me understand this project and its manifestations in great depths and helped us to complete the assigned tasks on time.

My thanks and appreciations also go to my fellow classmates in helping me understand the project and people who have willingly helped me out with their abilities.

# CHAPTER 1: INTRODUCTION OF THE PROJECT UNDERTAKEN

**Olympic Records Management System**

## Problem Statement

The existing methods for managing Olympic records are often manual, time-consuming, and prone to errors. There is a need for an efficient and automated system to store, retrieve, and analyze Olympic data.

## Project Objective

The primary objective of this project is to develop a console-based application using Python that can effectively manage Olympic records. The application should provide functionalities for:

* Storing and retrieving athlete information
* Recording and updating Olympic event results
* Analyzing and generating reports on athlete performance and Olympic trends

## Methodology for Solving the Problem

To achieve the project objectives, the following methodology will be employed:

1. Data Structure: The application utilizes dictionaries to store athlete information and lists to store Olympic records, providing a flexible and efficient data structure.
2. User Interface: A console-based interface is developed using Python's input and output functions to allow users to interact with the application.
3. Functions: The application consists of various functions to perform specific tasks, such as adding/updating athletes and records, viewing details, calculating statistics, and searching for information.
4. Data Validation: Input validation is implemented to ensure data integrity and prevent errors.

## Scope of the Work Undertaken and Importance

The scope of this project includes:

* + Managing athlete profiles (name, birth date, birthplace, country)
  + Recording event results (year, event, rank)
  + Searching for athletes or records based on specific criteria
  + Generating reports on athlete performance, medal count by country, and overall statistics.

The importance of this project lies in its potential to streamline the management of Olympic records, provide valuable insights into athlete performance and Olympic history, and support decision-making processes within the Olympic movement.

## Significance of the Project

This project has significant implications for the following areas:

* + Sports Administration: The application can assist sports organizations in managing athlete data, tracking performance, and analyzing trends.
  + Research: Researchers can utilize the application to study athlete performance, the evolution of sports, and the impact of various factors on Olympic outcomes.
  + Sports Media: Journalists and media outlets can leverage the application to access and analyze Olympic data for reporting and analysis.

# CHAPTER 2: FEATURES OF THE PROJECT

Based on the provided code, the Olympic Records Management System offers the following features:

## Data Management

* + **Athlete Information:** Stores and manages athlete profiles, including name, birth date, birthplace, and country.
  + **Olympic Records:** Records and updates Olympic event results for various years, including athlete name, event, and rank.

## Data Retrieval and Analysis

* + **View Details:** Allows users to view all Olympic records for a specific year.
  + **Medal Count by Country:** Calculates and displays the total number of medals won by each country.
  + **Search Athlete Performance:** Enables users to search for an athlete's performance history across different Olympic years.
  + **View Event Details:** Provides information about all athletes who participated in a specific event.
  + **View Statistics:** Calculates and displays overall statistics, such as the total number of events, athletes, and average rank.

## User Interaction

* + **Console Interface:** A user-friendly console interface provides a simple and intuitive way to interact with the application.
  + **Menu-Driven Navigation:** The application uses a menu system to guide users through different functionalities.

## Additional Features

* + **Data Validation:** Input validation ensures data integrity and prevents errors.
  + **Data Persistence:** While not explicitly implemented in the provided code, the system could be extended to store data in a file or database for long-term persistence.
  + **Advanced Analytics:** Future enhancements could include more advanced analytics capabilities, such as analyzing performance trends over time or identifying correlations between athlete attributes and performance.

.

# CHAPTER 3: PROGRAM CODE

olympics\_data = { "2008": [

["Vebjørn Berg", "10 m air rifle", "\_"],

["Vebjørn Berg", "50 m rifle prone", "4"], ["Vebjørn Berg", "50 m rifle 3 positions", "8"], ["Espen Berg-Knutsen", "50 m rifle prone", "\_"],

["Espen Berg-Knutsen", "50 m rifle 3 positions", "\_"], ["Tore Brovold", "Skeet", "Silver"],

["Are Hansen", "10 m air rifle", "\_"],

["Harald Jensen", "Skeet", "\_"]

], "2012": [

["Ole Magnus Bakken", "50 m rifle prone", "\_"], ["Ole Magnus Bakken", "10 m air rifle", "8"], ["Odd Arne Brekne", "50 m rifle prone", "\_"], ["Odd Arne Brekne", "50 m rifle 3 positions", "\_"], ["Tore Brovold", "Skeet", "\_"],

["Ole Kristian Bryhn", "50 m rifle 3 positions", "7"], ["Are Hansen", "10 m air rifle", "\_"],

["Are Hansen", "50 m rifle 3 positions", "\_"]

], "2016": [

["Odd Arne Brekne", "50 m rifle prone", "\_"], ["Odd Arne Brekne", "50 m rifle 3 positions", "\_"], ["Ole Kristian Bryhn", "10 m air rifle", "\_"],

["Ole Kristian Bryhn", "50 m rifle prone", "8"], ["Ole Kristian Bryhn", "50 m rifle 3 positions", "\_"], ["Are Hansen", "10 m air rifle", "\_"]

], "2020": [

["Jon-Hermann Hegg", "Men's 10 m air rifle", "\_"],

["Jon-Hermann Hegg", "Men's 50 m rifle 3 positions", "4"], ["Henrik Larsen", "Men's 10 m air rifle", "\_"],

["Henrik Larsen", "Men's 50 m rifle 3 positions", "\_"], ["Erik Watndal", "Men's skeet", "\_"]

],

"2024": [

["Jon-Hermann Hegg", "50 m rifle 3 positions", "5"],

["Ole Martin Halvorsen", "50 m rifle 3 positions", "5"]

]

}

athlete\_info = {

"Vebjørn Berg": ["1 April 1980", "Hamar, Innlandet (NOR)", "Norway"], "Espen Berg-Knutsen": ["2 October 1969", "Oslo, Norway", "Norway"], "Tore Brovold": ["12 June 1970", "Hamar, Innlandet (NOR)", "Norway"], "Are Hansen": ["16 January 1982", "Fredrikstad, Viken (NOR)", "Norway"], "Harald Jensen": ["18 December 1966", "Asker, Viken (NOR)", "Norway"],

"Ole Magnus Bakken": ["28 April 1988", "Hamar, Innlandet (NOR)", "Norway"], "Odd Arne Brekne": ["1 September 1984", "Farsund, Agder (NOR)", "Norway"], "Ole Kristian Bryhn": ["1 May 1989", "Drammen, Viken (NOR)", "Norway"], "Jon-Hermann Hegg": ["26 March 1999", "Dingle, Kerry (IRL)", "Norway"], "Henrik Larsen": ["8 September 1997", "Fredrikstad, Viken (NOR)", "Norway"], "Erik Watndal": ["27 August 1979", "Oslo, Oslo (NOR)", "Norway"],

"Ole Martin Halvorsen": ["11 February 2000", "Norway", "Norway"]

}

def add\_or\_update\_athlete():

name = input("Enter athlete's name: ") if name in athlete\_info:

print(f"Existing details for {name}: {athlete\_info[name]}") update = input("Do you want to update these details? (yes/no): ") if update.lower() != 'yes':

return

birth\_date = input("Enter birth date (e.g., 1 January 1980): ")

birthplace = input("Enter birthplace (e.g., Oslo, Norway): ") country = input("Enter country: ")

athlete\_info[name] = [birth\_date, birthplace, country] print(f"Details for {name} have been added/updated.")

def view\_athlete\_info():

choice = input("Enter 'all' to view all athletes or 'specific' to view a specific athlete: ") if choice.lower() == 'all':

print(f"\n{'Athlete':<25} {'Birth Date':<20} {'Birthplace':<35} {'Country':<15}") print("=" \* 85)

for name, details in athlete\_info.items():

print(f"{name:<25} {details[0]:<20} {details[1]:<35} {details[2]:<15}") elif choice.lower() == 'specific':

name = input("Enter athlete's name: ") if name in athlete\_info:

details = athlete\_info[name]

print(f"\n{'Athlete':<25} {'Birth Date':<15} {'Birthplace':<30} {'Country':<15}") print("=" \* 85)

print(f"{name:<25} {details[0]:<15} {details[1]:<30} {details[2]:<15}") else:

print(f"No information found for athlete: {name}")

else:

print("Invalid choice. Please try again.")

def add\_or\_update\_record():

year = input("Enter the Olympic year (2008, 2012, 2016, 2020, 2024): ") if year not in olympics\_data:

olympics\_data[year] = []

athlete\_name = input("Enter athlete's name: ") event = input("Enter event name: ")

rank = input("Enter rank (or '\_' if not ranked): ") olympics\_data[year].append([athlete\_name, event, rank])

print(f"Record for {athlete\_name} in {year} has been added/updated.")

def view\_details():

year = input("Enter the Olympic year (2008, 2012, 2016, 2020, 2024) or type 'all' to view all details: ")

if year == 'all': view\_all\_olympics\_details()

elif year in olympics\_data:

print(f"\nDetails for {year} Olympics:\n") print(f"{'Athlete':<25} {'Event':<30} {'Rank':<10}") print("=" \* 65)

for record in olympics\_data[year]:

print(f"{record[0]:<25} {record[1]:<30} {record[2]:<10}")

else:

print("No data available for this year.")

def view\_all\_olympics\_details(): print("\nDetails for All Olympics:\n")

print(f"{'Year':<10} {'Athlete':<25} {'Event':<30} {'Rank':<10}") print("=" \* 75)

for year in olympics\_data:

for record in olympics\_data[year]:

print(f"{year:<10} {record[0]:<25} {record[1]:<30} {record[2]:<10}")

def view\_medal\_count\_by\_country(): medal\_count = {}

for year in olympics\_data:

for record in olympics\_data[year]:

if record[2] in ["1", "2", "3", "Gold", "Silver", "Bronze"]: country = athlete\_info[record[0]][2]

if country not in medal\_count: medal\_count[country] = 0

medal\_count[country] += 1

if medal\_count:

print("\nMedal Count by Country:")

print(f"{'Country':<20} {'Medals':<10}") print("=" \* 30)

for country, count in medal\_count.items(): print(f"{country:<20} {count:<10}")

else:

print("No medals found.")

def search\_athlete\_performance():

athlete\_name = input("Enter athlete's name: ") found = False

print(f"\nPerformance for {athlete\_name}:\n") print(f"{'Year':<10} {'Event':<30} {'Rank':<10}") print("=" \* 50)

for year in olympics\_data:

for record in olympics\_data[year]:

if record[0].lower() == athlete\_name.lower(): print(f"{year:<10} {record[1]:<30} {record[2]:<10}") found = True

if not found:

print("No records found for this athlete.")

def view\_event\_details():

event\_name = input("Enter event name: ") found = False

print(f"\nDetails for {event\_name}:\n") print(f"{'Year':<10} {'Athlete':<25} {'Rank':<10}") print("=" \* 45)

for year in olympics\_data:

for record in olympics\_data[year]:

if record[1].lower() == event\_name.lower(): print(f"{year:<10} {record[0]:<25} {record[2]:<10}") found = True

if not found:

print("No records found for this event.")

def view\_statistics():

total\_events = sum([len(olympics\_data[year]) for year in olympics\_data]) total\_athletes = len(set([record[0] for year in olympics\_data for record in

olympics\_data[year]]))

print(f"\nTotal Events: {total\_events}, Total Athletes: {total\_athletes}")

ranks = [int(record[2]) for year in olympics\_data for record in olympics\_data[year] if record[2].isdigit()]

if ranks:

average\_rank = sum(ranks) / len(ranks) print(f"Average Rank: {average\_rank:.2f}")

else:

print("No ranking data available to calculate average rank.")

def main(): while True:

print("\nOlympic Records Management System") print("1. View Details")

print("2. View Athlete Information") print("3. Search Athlete Performance") print("4. View Event Details") print("5. View Statistics")

print("6. Add or View Athlete Information") print("7. Add or Update Olympic Record") print("8. Exit")

choice = input("Enter your choice: ")

if choice == '1': view\_details()

elif choice == '2': view\_athlete\_info()

elif choice == '3':

search\_athlete\_performance() elif choice == '4':

view\_event\_details() elif choice == '5':

view\_statistics() elif choice == '6':

sub\_choice = input("Enter '1' to add/update athlete info or '2' to view athlete info: ") if sub\_choice == '1':

add\_or\_update\_athlete() elif sub\_choice == '2':

view\_athlete\_info() else:

print("Invalid choice. Please try again.") elif choice == '7':

add\_or\_update\_record() elif choice == '8':

print("\nExiting the system...\nTHANK YOU!") break

else:

print("Invalid choice. Please try again.")

main()

# RESULTS

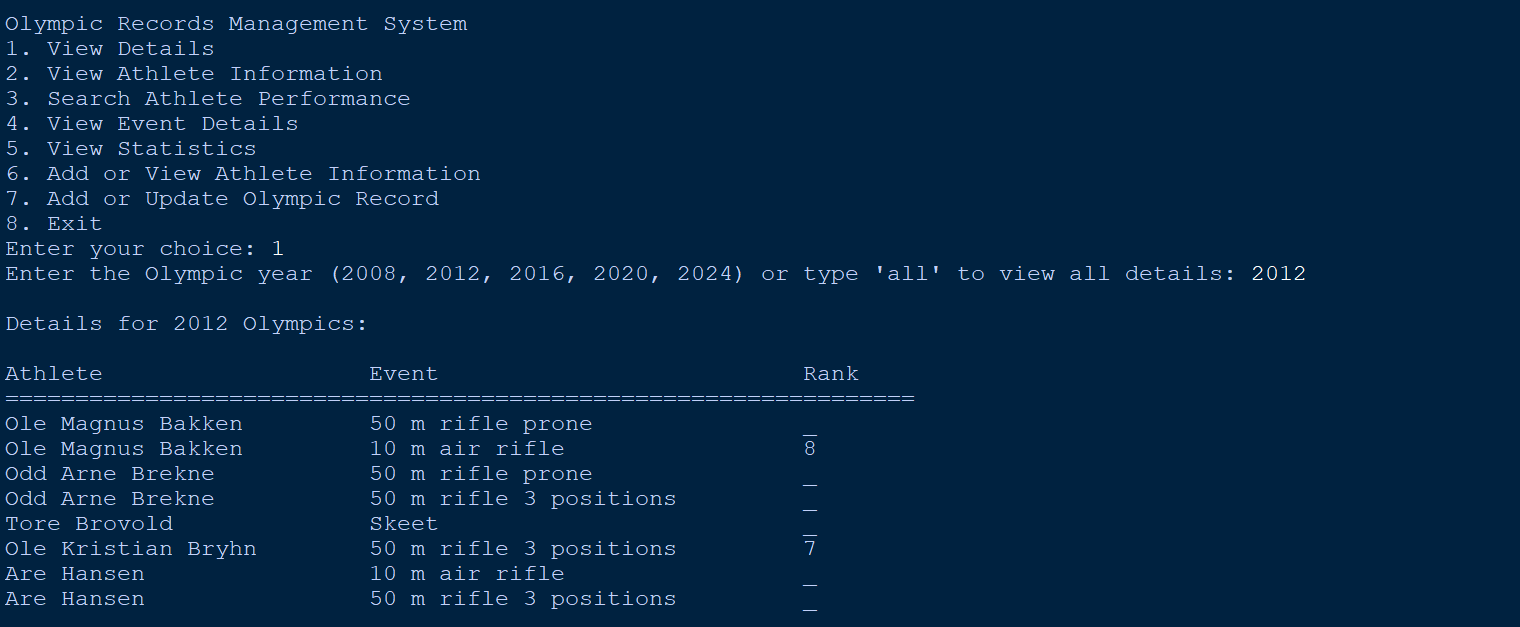
1. **View Details** Function: view details() Working:

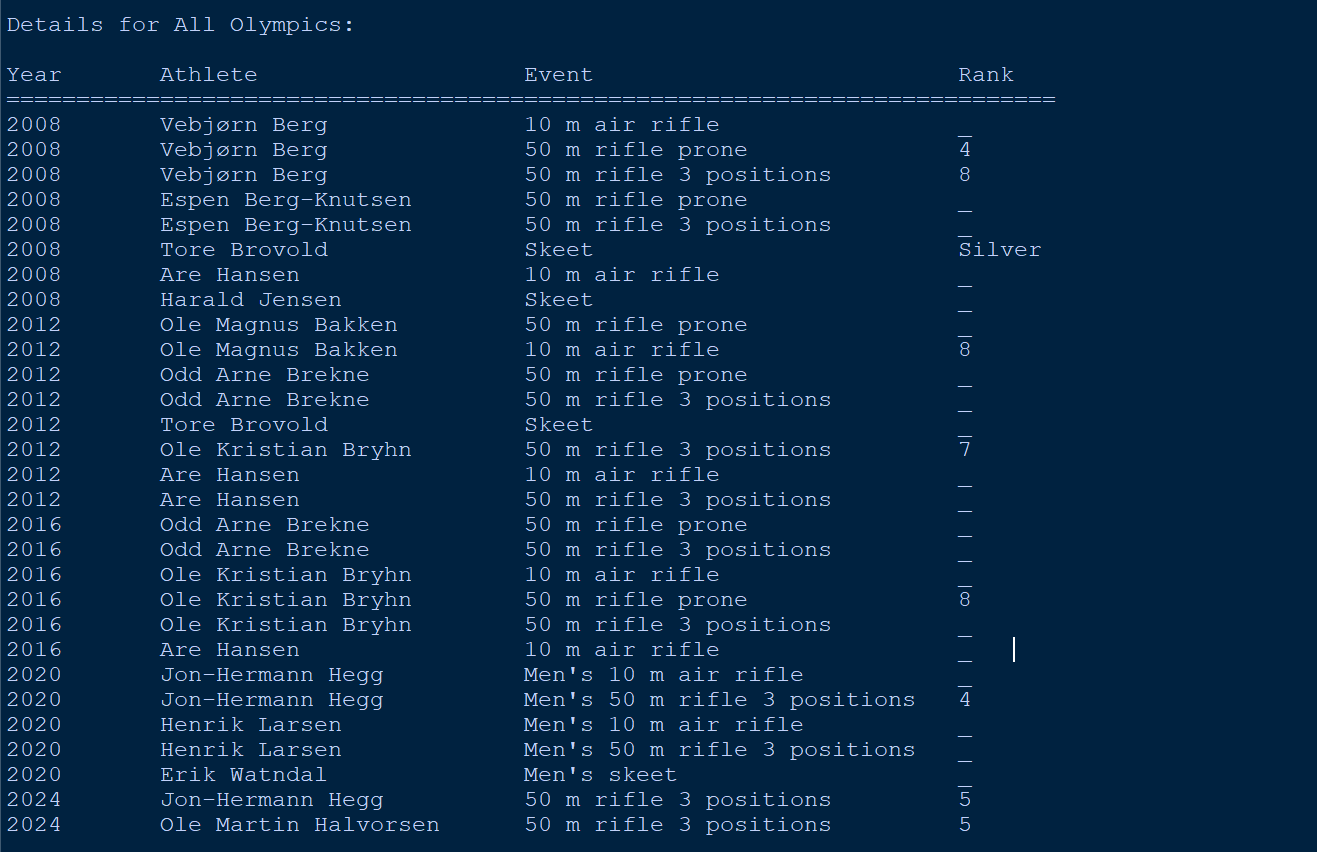
The program prompts you to enter an Olympic year (e.g., 2008, 2012) or "all" to view all details.

If you enter a specific year, it prints the details of the athletes who participated that year, showing their names, events, and ranks.

If you enter "all," it prints the details for all available years.

If the year you enter has no data, it informs you that no data is available.





**2.View Athlete information**

Function: view\_athlete\_info()

Working:

The program calculates the total number of medals won by each country.

It looks for entries in the dataset where the rank is "1," "2," "3," "Gold," "Silver," or "Bronze." It sums up the medals for each country and prints the medal count by country.

A computer screen shot of text

Description automatically generated

**3. Search Athlete Performance** Function: search\_athlete\_performance() Working:

The program prompts you to enter the name of an athlete.

It searches for the athlete's name across all years and events in the dataset.

If it finds the athlete, it prints the details of their performances, showing the year, event, and rank.

If no records are found for the athlete, it informs you of that.

A screenshot of a computer program

Description automatically generated

**4. View Event Details** Function: view\_event\_details() Working:

The program prompts you to enter the name of an event.

It searches for the event across all years and prints the details, showing the year, athlete, and rank.

If no records are found for the event, it informs you of that.

A screenshot of a computer

Description automatically generated

**5. View Statistics** Function: view\_statistics() Working:

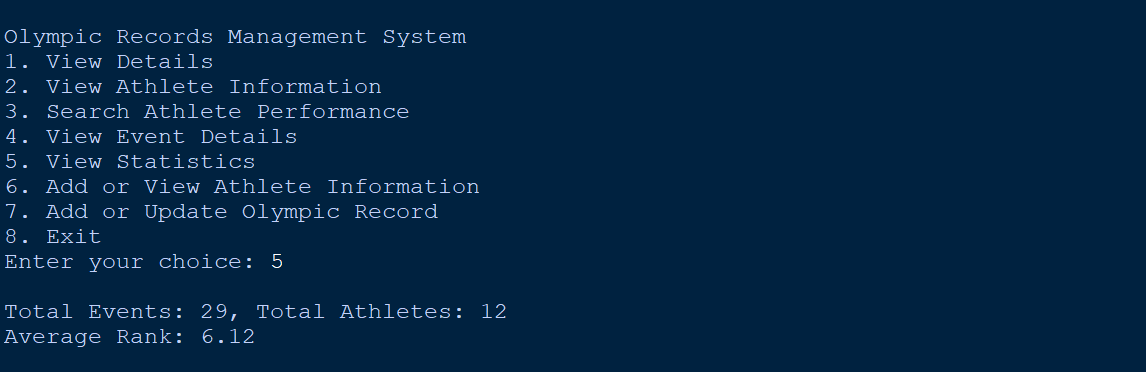
The program calculates and displays some overall statistics:

Total number of events recorded.

Total number of unique athletes who participated.

The average rank across all ranked entries (if any rank data is available).

If no ranking data is available, it informs you that it cannot calculate the average rank.



## 6. Add or Update Athlete Information

This function is used to add a new athlete or update the details of an existing athlete. Prompt for Athlete's Name: Requests the user to input the athlete's name.

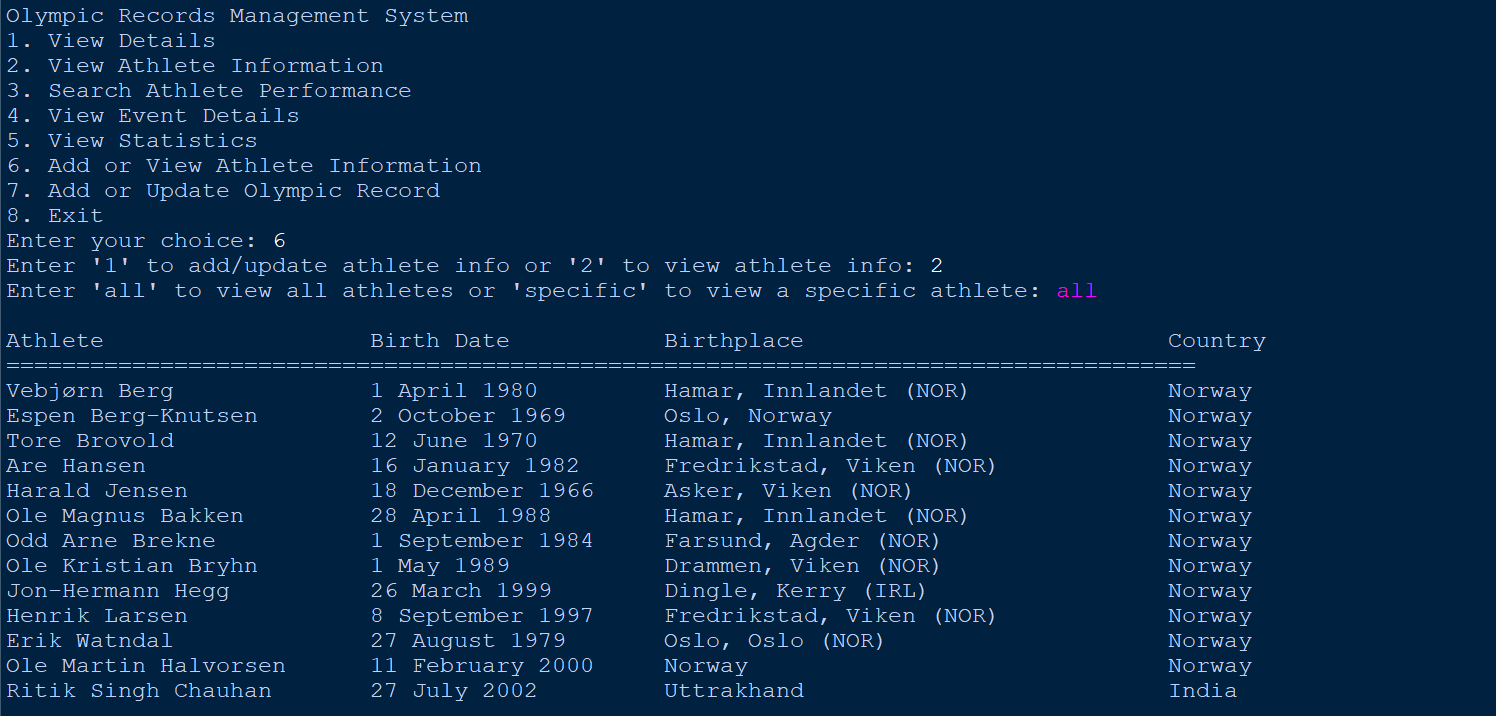
Check for Existing Details: If the athlete already exists in athlete\_info, it displays the current details and asks if the user wants to update them.

Input New Details: If updating, it prompts for the athlete's birth date, birthplace, and country. Update or Add Details: The details are added or updated in the athlete\_info dictionary.

Confirmation: Prints a message indicating the details have been added or updated.

A computer screen shot of a blue screen

Description automatically generated



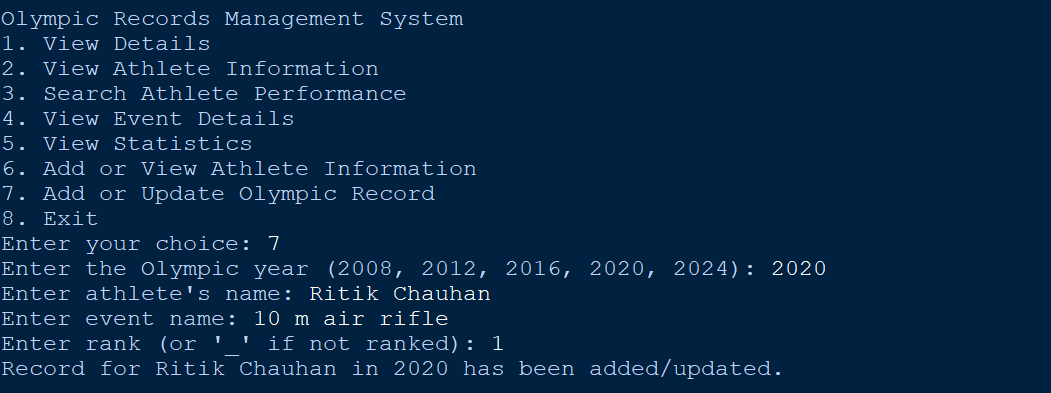
A computer screen shot of a blue screen

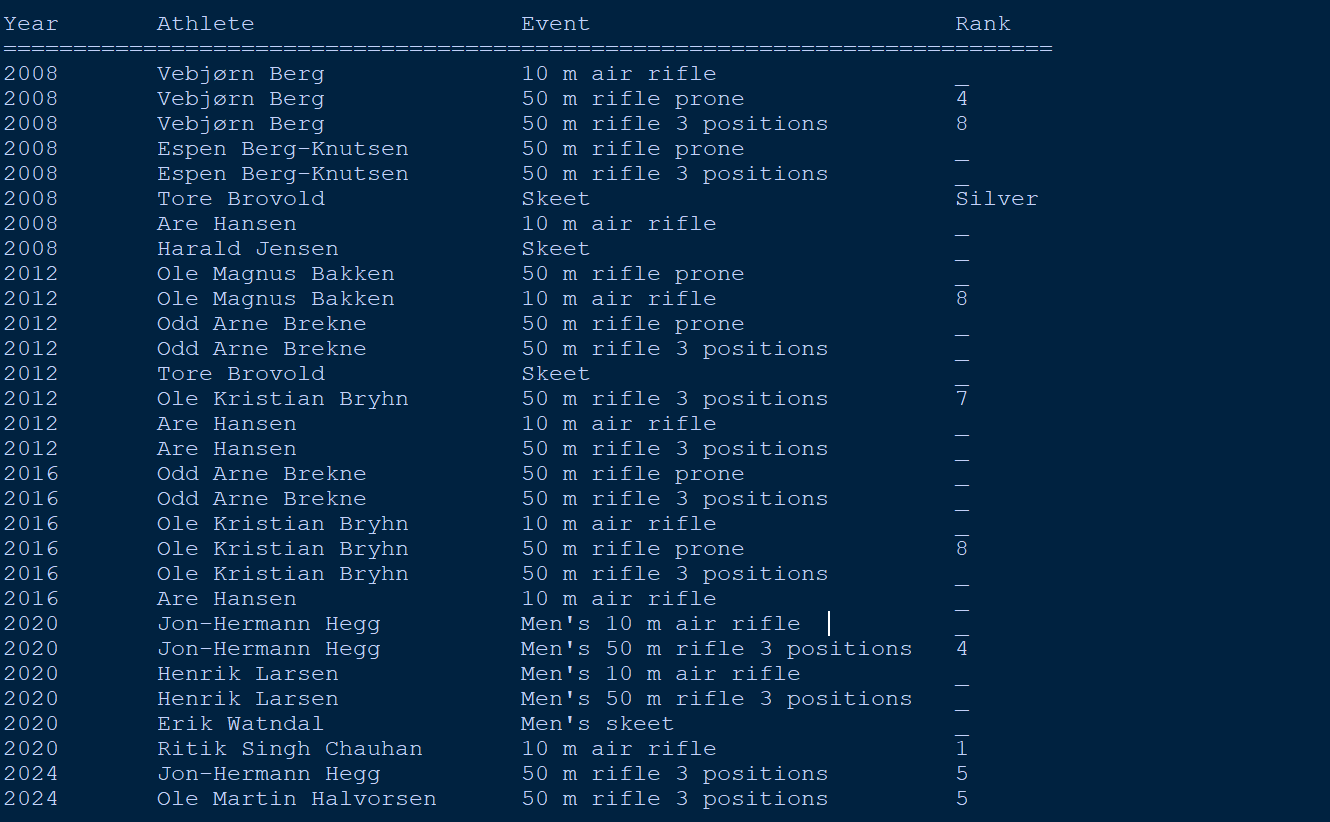
Description automatically generated

**7. Add or Update Olympic Record** Function: add\_or\_update\_record() Working:

The program prompts you to enter the Olympic year.

It then asks for the athlete's name, the event name, and the rank. It adds this record to the dataset for that specific year.





In the last,

**2020 Ritik Singh Chauhan** has been updated.

## 8. Exit

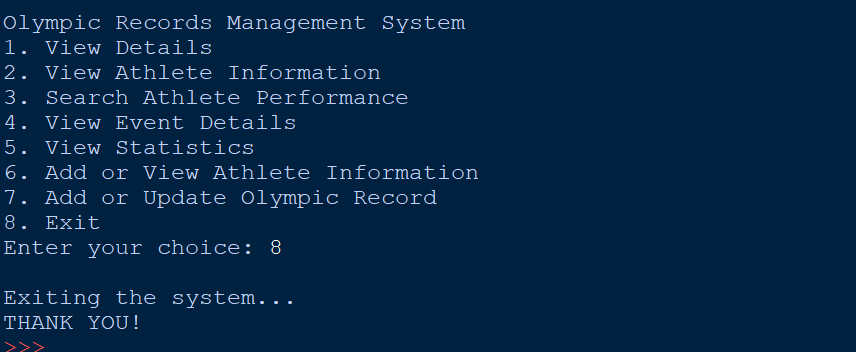
Function: main() loop Working:

If you enter "8," the program breaks out of the loop and prints a goodbye message. The program then ends.

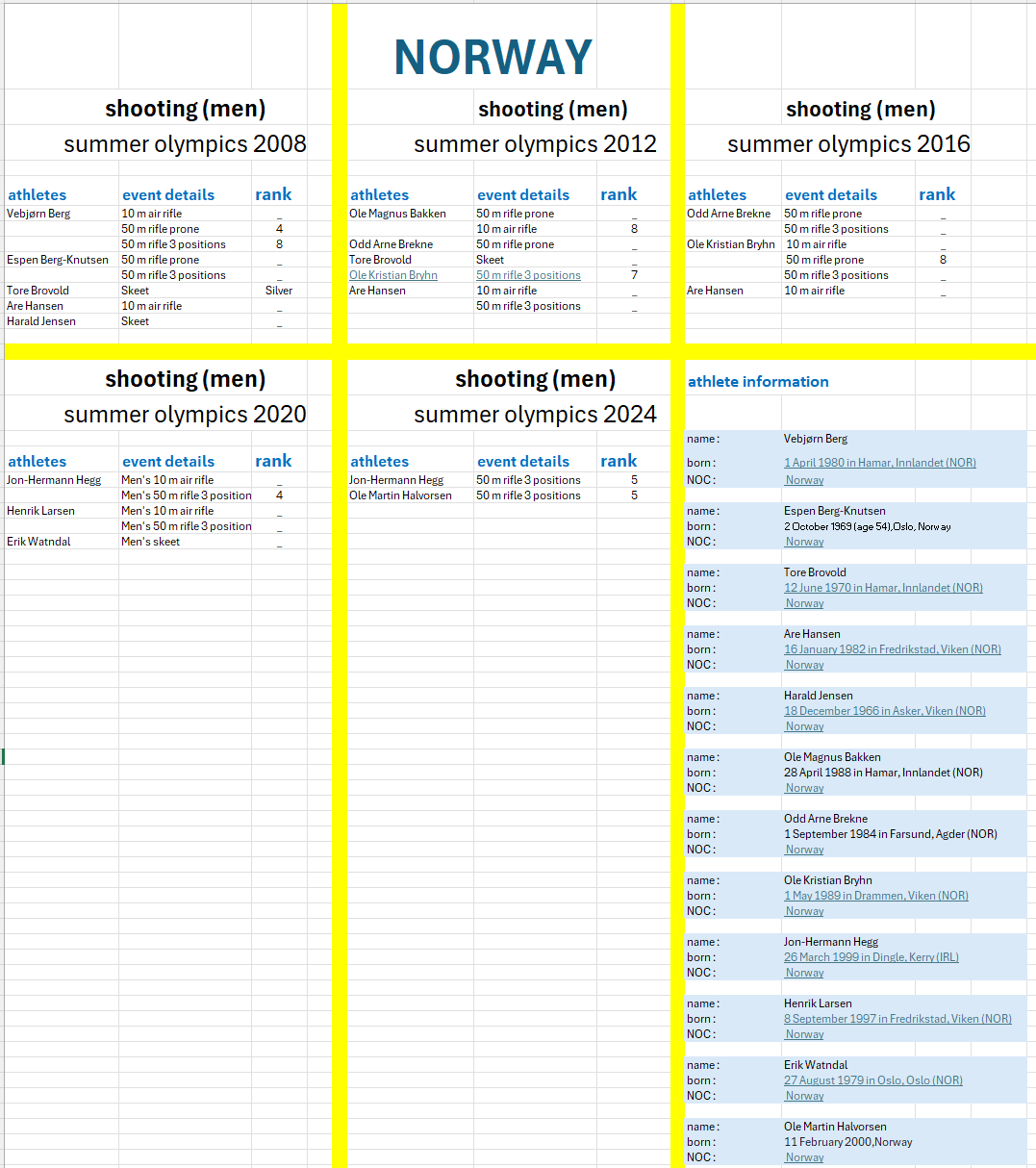
Invalid Choice

If you enter a choice that is not valid (anything other than "1" to "8"), the program will print an error message: "Invalid choice. Please try again."

It will then prompt you again to enter a valid choice.



# INFORMATION COLLECTED :



**ADVANTAGES**

* **Efficiency:** The system automates the management of Olympic records, reducing manual effort and time spent on data entry and retrieval.
* **Accuracy:** By using a structured database, the system ensures data accuracy and consistency.
* **Accessibility:** The console-based interface makes the system easily accessible to users with varying levels of technical expertise.
* **Insights:** The system provides valuable insights into athlete performance, Olympic trends, and medal count by country, supporting decision-making and research.
* **Scalability:** The system can be easily scaled to accommodate a larger volume of data and users as the Olympic movement grows.

# DISADVANTAGES

* Limited Functionality: While the system offers essential features, it may lack advanced analytics capabilities or integration with other systems.
* Data Dependence: The system relies on accurate and complete data, and any errors or inconsistencies in the data can affect the reliability of the results.
* Security: If the system is not implemented with proper security measures, there may be risks of unauthorized access or data breaches.
* Maintenance: The system requires ongoing maintenance and updates to ensure its functionality and compatibility with evolving technologies.

# CONCLUSION

The Olympic Records Management System developed in this project effectively addresses the challenges associated with manual management of Olympic data. By providing features for data storage, retrieval, analysis, and reporting, the system offers a valuable tool for sports administrators, researchers, and media outlets.

The system's advantages include efficiency, accuracy, accessibility, insights, and scalability. However, it also has limitations such as its console-based interface, dependence on accurate data, and potential security risks. Future enhancements could include a graphical user interface, advanced analytics capabilities, and integration with other systems to further improve its functionality and value.

Overall, the Olympic Records Management System represents a valuable contribution to the management and analysis of Olympic data, providing a foundation for future developments and improvements in the field of sports analytics.

THANK YOU.