Project Report: Football Tournament Winner Search System

A Search System Application for Retrieving FIFA World Cup and UEFA Champions League Winners

BY:

AYUSH CHANDRA DEWAN

25BET10020

• CONTENTS

1 Introduction

The Football Tournament Winner Search System is a simple, search system developed in Python. The primary goal of this project is to provide users with a quick and efficient way to look up historical winners of two of the world's most prominent football competitions: the FIFA World Cup and the UEFA Champions League. This system serves as a foundational example of data storage (using in-memory dictionaries) and interactive user input handling in Python.

2 Problem Statement

The objective is to create a reliable and intuitive console-based tool that centralizes historical winner data for the FIFA World Cup (1930–2022) and the UEFA Champions League (2000–2024). The system must allow users to search for a specific winner by year or view a complete list of all stored winners, mitigating the need to search multiple external sources for this common sports trivia.

3 Functional Requirements

Functional requirements define the specific actions the system must perform.

1. **Display Main Menu:** The system must present a clear menu with options to search, view all winners, or exit.
2. **Search FIFA Winner:** The system must accept a year input and return the winner of the FIFA World Cup for that year, or an error message if the year is invalid or data is missing.
3. **Search UCL Winner:** The system must accept a year input and return the winner of the UEFA Champions League for that year, or an error message if the year is outside the available range (2000–2024).
4. **View All FIFA Winners:** The system must iterate through and display the entire list of FIFA World Cup winners (Year: Team format).
5. **View All UCL Winners:** The system must iterate through and display the entire list of UEFA Champions League winners (Year: Team format).
6. **Exit Functionality:** The system must allow the user to gracefully exit the application loop.

4 Non-functional Requirements

Non-functional requirements specify criteria used to judge the operation of a system, rather than specific behaviors.

1. **Usability:** The console interface must be intuitive, providing clear prompts and readable output.
2. **Maintainability:** The core data structures (winner dictionaries) must be easy to locate and update when new tournament winners are crowned.

3. **Reliability:** The program must not crash upon receiving non-numeric or out-ofrange input; robust error handling is required.
4. **Performance:** Search operations must be instantaneous, leveraging the constant time complexity ($O(1)$) of dictionary lookups.

5 System Architecture

The system employs a simple, single-tier, monolithic architecture, common for small console applications.

- **Presentation Layer (I/O):** Handled by Python's built-in input() and print() functions, forming the user interface.
- **BusinessLogicLayer:** Contained within the search_fifa(), search_ucl(), and main_menu() functions, which manage user interaction and data access.
- **Data Layer:** The data is stored in-memory using two global Python dictionaries, fifa_world_cup_winners and ucl_winners.

6 Design Diagrams

Due to the constraints of single-file compilation, the diagrams below are represented conceptually using structured text.

6.1 Use Case Diagram

- **Actor:** User
- **Use Cases:**
  1. Search Winner (FIFA or UCL)
  2. View All Winners (FIFA or UCL)
  3. Exit System
  4. Manage Menu
- **Relationship:** The User interacts directly with the Manage Menu, which includes the other three use cases.

6.2 Workflow Diagram

1. **Start:** Execution begins with main_menu().
2. **Loop/Menu Display:** Print the menu options.
3. **Input Choice:** Get user input (1-5).
4. **Process Choice:**

- If 1 or 2: Call the respective search() function.
- If 3 or 4: Iterate and print the respective dictionary data.
- If 5: Break the loop and Exit.
- If Invalid: Print error and continue loop.

5. **End:** Application terminates.

6.3 Sequence Diagram

1. **Participant:** User
2. **Participant:** System (main_menu) 3. **Participant:** Data Layer (dictionaries)

4. **Sequence (Search Scenario):**

   - User → System: Start (main_menu())
   - System → User: Display Menu
   - User → System: Choice (e.g., ’1’)
   - System → User: Prompt for Year
   - User → System: Year Input (e.g., 2014)
   - System → Data Layer: Lookup (2014 in fifa_world_cup_winners)
   - Data Layer → System: Return ”Germany”
   - System → User: Display ”FIFA World Cup 2014 Winner: Germany”
   - System → User: Display Menu (Loop)

6.4 Class/Component Diagram

- **Component 1: Data Store** (Global Dictionaries)
  - fifa_world_cup_winners: Dictionary {Year: Country}
  - ucl_winners: Dictionary {Year: Club}
- **Component 2: Functionality**
  - search_fifa(): Reads input, accesses Data Store.
  - search_ucl(): Reads input, accesses Data Store.
  - main_menu(): Controls application flow, calls other functions.

6.5 ER Diagram

An Entity-Relationship (ER) Diagram is not applicable to this project as it utilizes in-memory Python dictionaries for data storage, not a relational database. The data is stored as keyvalue pairs, which is a conceptual NoSQL/Map structure.

7 Design Decisions & Rationale

1. **In-Memory Dictionaries for Data Storage: • Rationale:** For small, static datasets like historical winners, in-memory dictionaries are the fastest and simplest storage method in Python. They offer $O(1)$ (constant time) complexity for lookups, ensuring immediate retrieval.
2. **Primary Key as Year:**
   - **Rationale:** The year is the natural, unique identifier for each tournament instance. Using the year as the dictionary key allows for direct, immediate access to the winner.
3. **Single main_menu() Loop:**
   - **Rationale:** Encapsulating the program flow within a while True loop managed by a central function simplifies control flow, allowing users to perform multiple operations without restarting the script.
4. **Use of int(input(...)):**
   - **Rationale:** Enforcing integer input ensures data integrity before the dictionary lookup. However, this is a minor weakness that could be improved with tryexcept blocks to catch ValueError if the user enters non-numeric text.

8 Implementation Details

The system is implemented entirely in Python, utilizing standard libraries only.

1. **DataInitialization:** The two large dictionaries (fifa_world_cup_winners and ucl_winners) are defined globally at the start of the script, making them easily accessible by all functions.
2. **Search Functions (search_fifa, search_ucl):**
   - Take user input for the year.
   - Use the if year in dictionary: check for validation. This is efficient as it checks for the existence of the key.
   - If the key exists, the result is printed using an f-string: f"Winner: {dictionary[year]}".
3. **Viewing All Winners:**
   - Options 3 and 4 use a simple for year, team in dictionary.items(): loop to iterate over all key-value pairs and print them one by one.

9 Screenshots / Results

The following outlines illustrate the expected console output for key operations.

```
-- Menu Display --
============================== FOOTBALL
  WINNERS SEARCH
============================== ...
Enter your choice (1-5): 2
```

```
-- Search UCL Success --
Enter a UEFA Champions League year (2000–2024): 2024
  UEFA Champions League 2024 Winner: Real Madrid

-- Search FIFA Failure --
Enter a FIFA World Cup year (1930–2022): 2023 No World
  Cup held this year OR invalid input.

-- View All FIFA Winners (Partial Output) -ALL FIFA WORLD
  CUP WINNERS:
1930: Uruguay
1934: Italy ...
2022: Argentina
```

10 Testing Approach

A manual, black-box testing approach was employed due to the project's small scope and reliance on user input.

1. **Positive Testing:**
   - **Case 1 (Search):** Input valid years (e.g., 1986 for FIFA, 2011 for UCL) and verify the correct winner is returned.
   - **Case 2 (View):** Select options 3 and 4 and confirm all stored data is displayed correctly.
   - **Case 3 (Exit):** Select option 5 and confirm the program terminates.
2. **Negative Testing:**
   - **Case 1 (Invalid Year):** Input a year outside the data range (e.g., 1900, 2050) and confirm the "No World Cup held this year OR invalid input" message appears.
   - **Case 2 (Invalid Choice):** Input a non-menu option (e.g., 6, A) and confirm the "Invalid option, try again!" message appears.
   - **Case 3 (Error Handling - Conceptual):** Although not implemented with tryexcept, inputting non-numeric text (e.g., 'abc') would be expected to cause a ValueError in the current version, highlighting an area for improvement.

11 Challenges Faced

The primary challenge of this project was minimal, owing to the straightforward nature of the request.

- **Robust Input Handling:** The current implementation relies on int(input()), which is susceptible to crashing if the user enters letters instead of numbers. A more robust solution would require implementing try-except blocks for type conversion, which was deferred to the future enhancements stage to keep the initial code simple.

- **Data Organization:** Ensuring all historical data was correctly transcribed and formatted into the Python dictionaries was a tedious but necessary task.

12 Learnings & Key Takeaways

- **Efficiency of Dictionaries:** Reinforced the understanding that Python dictionaries (hash maps) provide extremely efficient key-based lookups, making them ideal for small data retrieval systems.
- **Program Control Flow:** Gained practical experience in managing continuous program execution using a while loop and conditional statements (if/elif/else) for menu navigation.
- **F-String Utility:** Applied f-strings for clear and concise output formatting in the console.

13 Future Enhancements

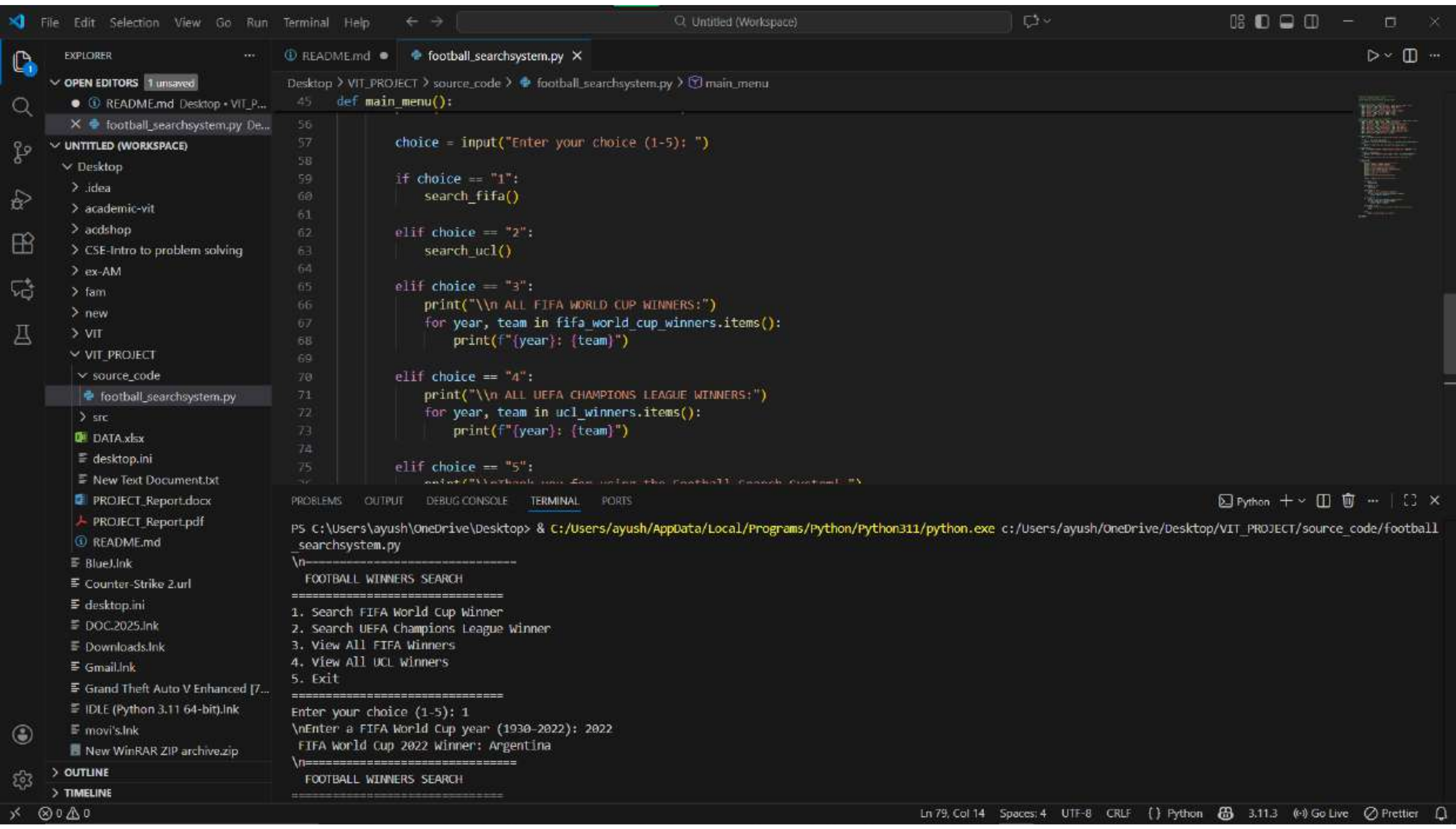
1. **Error Handling Improvement:** Implement try-except blocks around input() calls to gracefully handle ValueError when non-numeric input is provided by the user.
2. **Persistence Layer:** Migrate the data from in-memory dictionaries to a local file (e.g., JSON or CSV) or a simple SQLite database to ensure data persistence across sessions.
3. **Search by Team Name:** Implement a reverse lookup feature that allows the user to input a team name and see all years that team won either tournament.
4. **Graphical User Interface (GUI):** Transition the application from a CLI to a simple GUI for enhanced usability.

14 References

Python Language Documentation (Version 3.x) – python.org

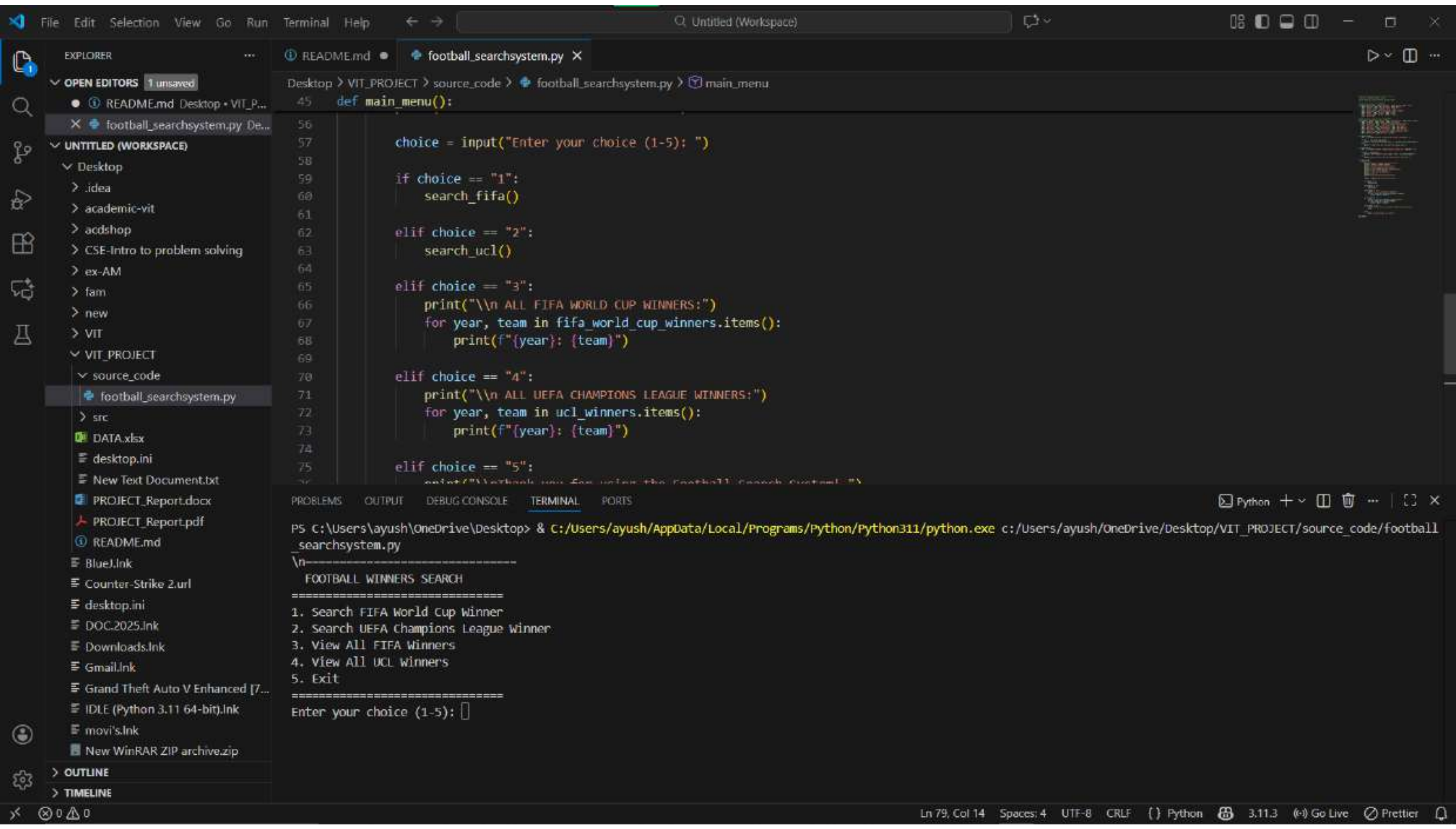
FIFA World Cup Historical Records –https://www.fifa.com

UEFA Champions League Historical Records – https://www.uefa.com



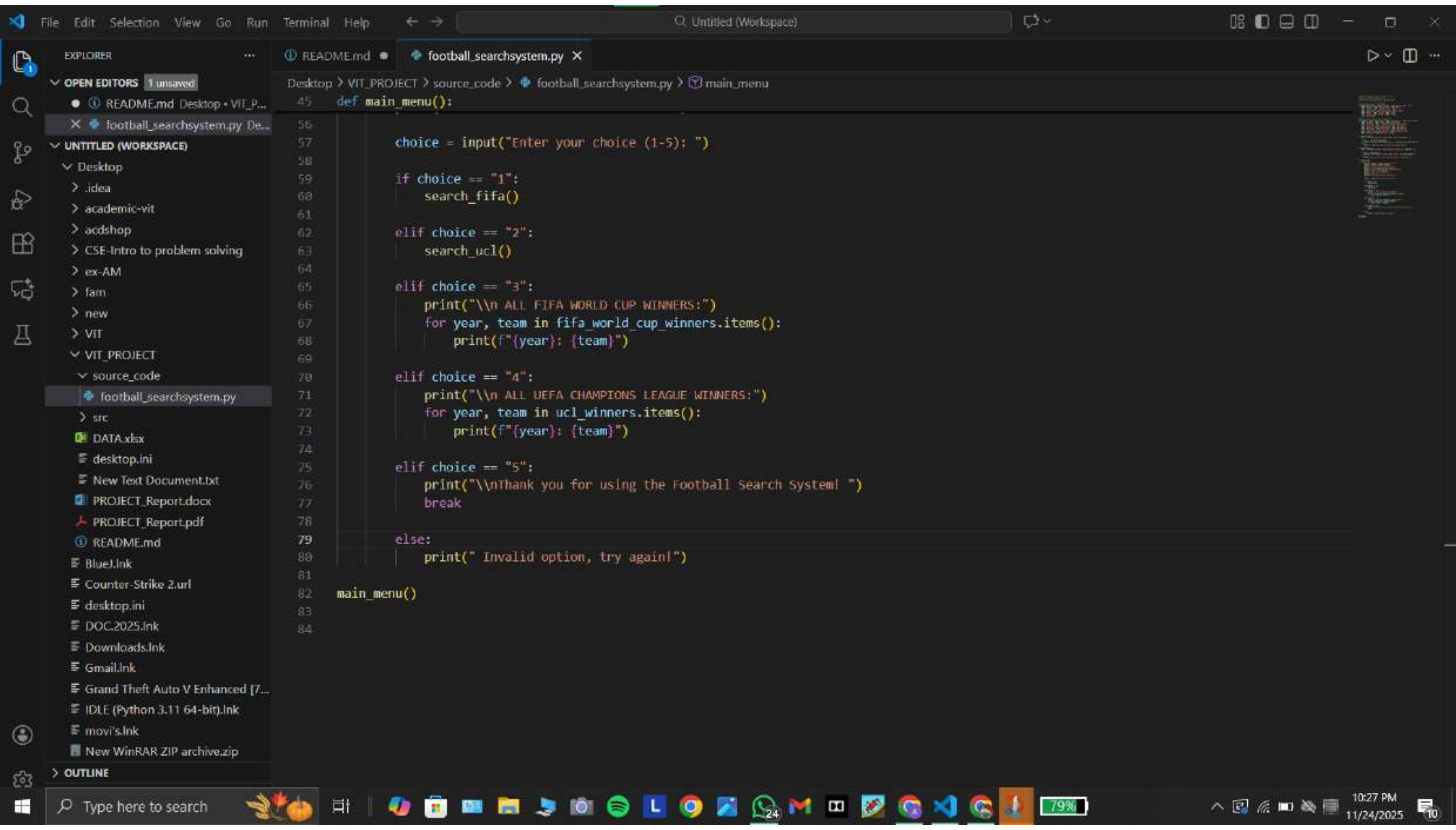
```python
def main_menu():

        choice = input("Enter your choice (1-5): ")

        if choice == "1":
            search_fifa()

        elif choice == "2":
            search_ucl()

        elif choice == "3":
            print("\\n ALL FIFA WORLD CUP WINNERS:")
            for year, team in fifa_world_cup_winners.items():
                print(f"{year}: {team}")

        elif choice == "4":
            print("\\n ALL UEFA CHAMPIONS LEAGUE WINNERS:")
            for year, team in ucl_winners.items():
                print(f"{year}: {team}")

        elif choice == "5":
```

```
PS C:\Users\ayush\OneDrive\Desktop> & C:/Users/ayush/AppData/Local/Programs/Python/Python311/python.exe c:/Users/ayush/OneDrive/Desktop/VIT_PROJECT/source_code/football
_searchsystem.py
\n==============================
  FOOTBALL WINNERS SEARCH
==============================
1. Search FIFA World Cup Winner
2. Search UEFA Champions League Winner
3. View All FIFA Winners
4. View All UCL Winners
5. Exit
==============================
Enter your choice (1-5): 1
\nEnter a FIFA World Cup year (1930-2022): 2022
 FIFA World Cup 2022 Winner: Argentina
\n==============================
  FOOTBALL WINNERS SEARCH
==============================
```

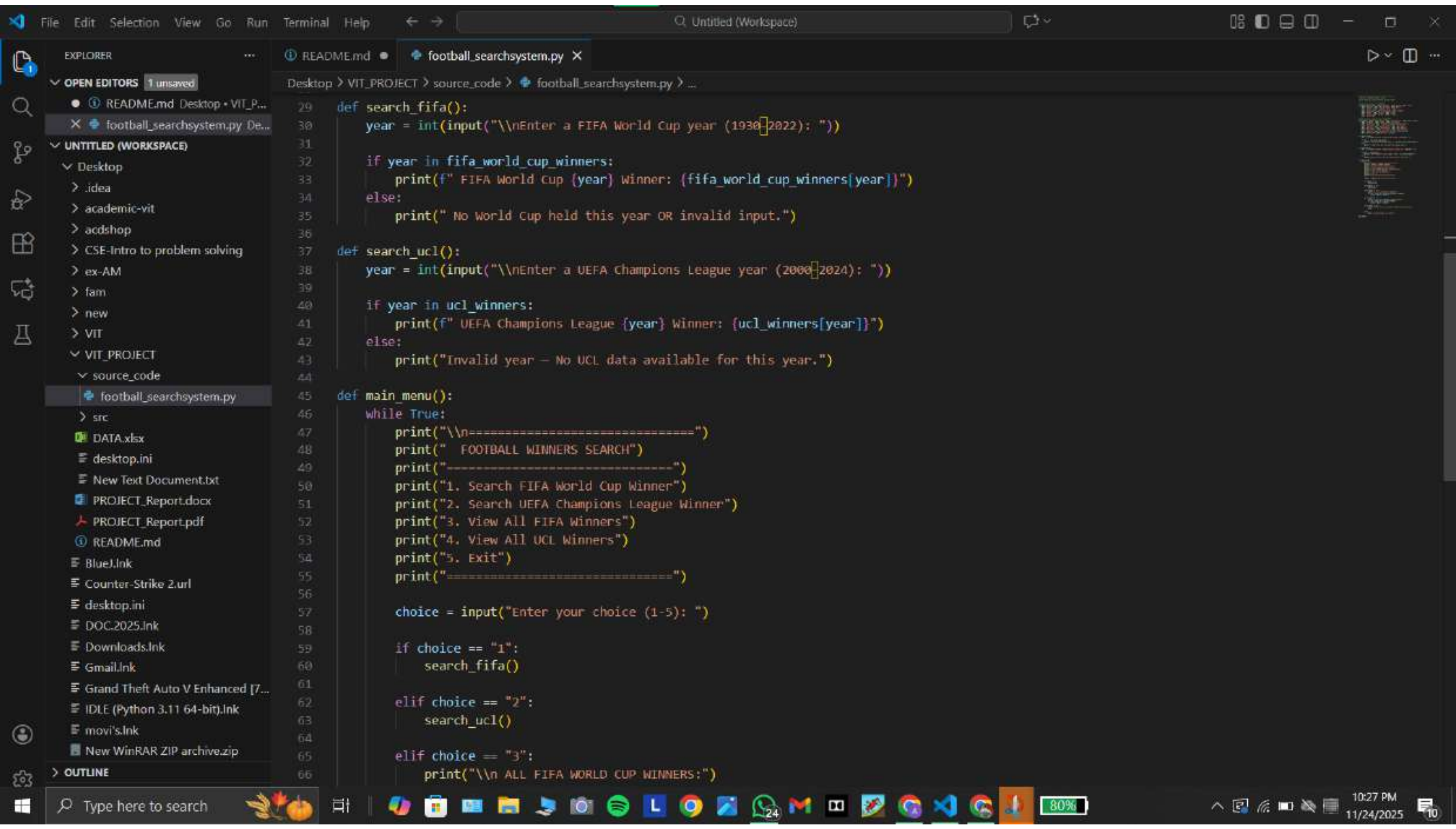


```python
def main_menu():

        choice = input("Enter your choice (1-5): ")

        if choice == "1":
            search_fifa()

        elif choice == "2":
            search_ucl()

        elif choice == "3":
            print("\\n ALL FIFA WORLD CUP WINNERS:")
            for year, team in fifa_world_cup_winners.items():
                print(f"{year}: {team}")

        elif choice == "4":
            print("\\n ALL UEFA CHAMPIONS LEAGUE WINNERS:")
            for year, team in ucl_winners.items():
                print(f"{year}: {team}")

        elif choice == "5":
```

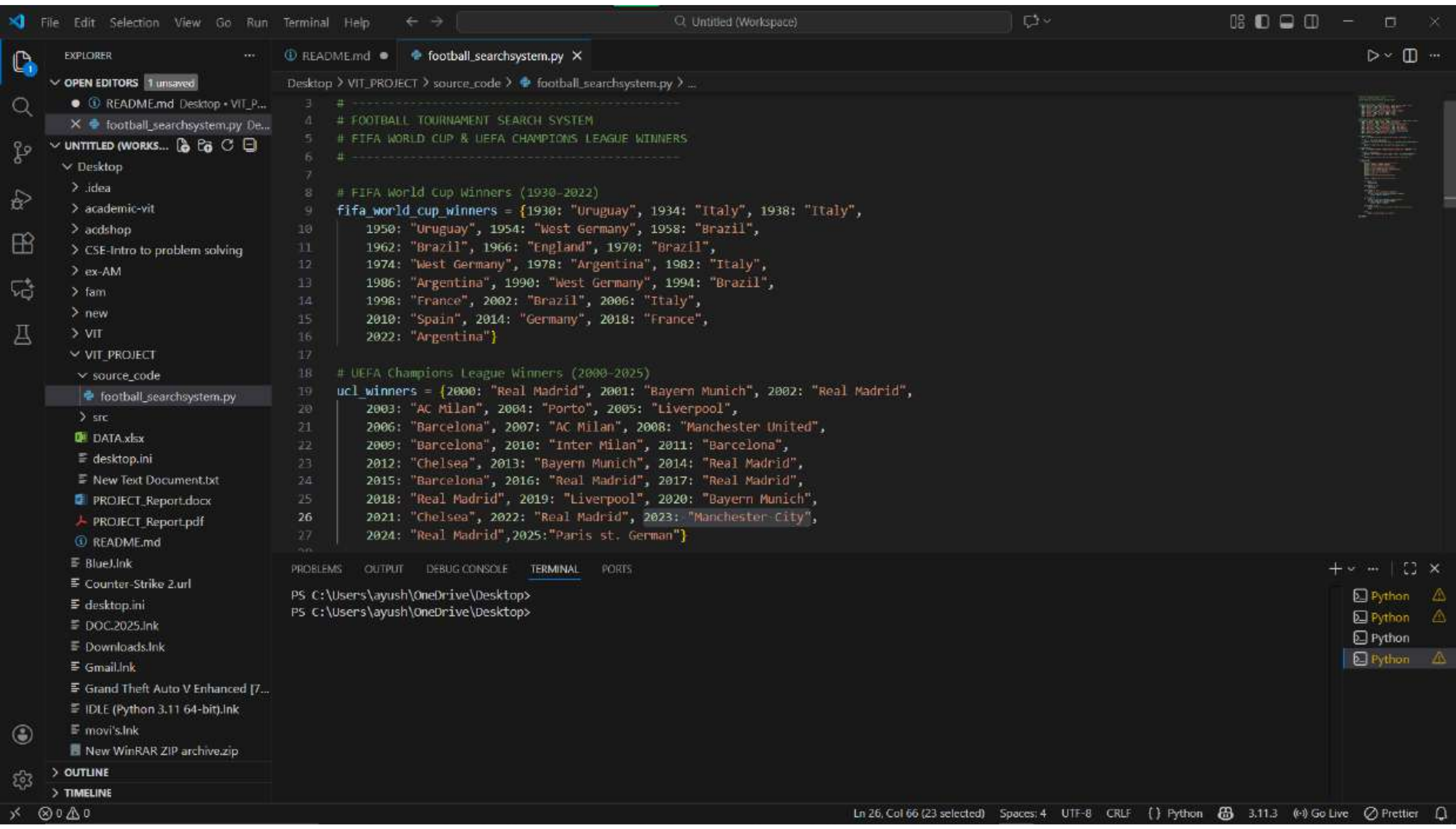
```
PS C:\Users\ayush\OneDrive\Desktop> & C:/Users/ayush/AppData/Local/Programs/Python/Python311/python.exe c:/Users/ayush/OneDrive/Desktop/VIT_PROJECT/source_code/football
_searchsystem.py
\n------------------------------
  FOOTBALL WINNERS SEARCH
==============================
1. Search FIFA World Cup Winner
2. Search UEFA Champions League Winner
3. View All FIFA Winners
4. View All UCL Winners
5. Exit
==============================
Enter your choice (1-5): 
```


README.md
football_searchsystem.py
Desktop > VIT_PROJECT > source_code > football_searchsystem.py > main_menu

```python
def main_menu():

        choice = input("Enter your choice (1-5): ")

        if choice == "1":
            search_fifa()

        elif choice == "2":
            search_ucl()

        elif choice == "3":
            print("\\n ALL FIFA WORLD CUP WINNERS:")
            for year, team in fifa_world_cup_winners.items():
                print(f"{year}: {team}")

        elif choice == "4":
            print("\\n ALL UEFA CHAMPIONS LEAGUE WINNERS:")
            for year, team in ucl_winners.items():
                print(f"{year}: {team}")

        elif choice == "5":
            print("\\nThank you for using the Football Search System! ")
            break

        else:
            print(" Invalid option, try again!")

main_menu()
```



```python
def search_fifa():
    year = int(input("\\nEnter a FIFA World Cup year (1930-2022): "))

    if year in fifa_world_cup_winners:
        print(f" FIFA World Cup {year} Winner: {fifa_world_cup_winners[year]}")
    else:
        print(" No World Cup held this year OR invalid input.")

def search_ucl():
    year = int(input("\\nEnter a UEFA Champions League year (2000-2024): "))

    if year in ucl_winners:
        print(f" UEFA Champions League {year} Winner: {ucl_winners[year]}")
    else:
        print("Invalid year – No UCL data available for this year.")

def main_menu():
    while True:
        print("\\n===============================")
        print("  FOOTBALL WINNERS SEARCH")
        print("-------------------------------")
        print("1. Search FIFA World Cup Winner")
        print("2. Search UEFA Champions League Winner")
        print("3. View All FIFA Winners")
        print("4. View All UCL Winners")
        print("5. Exit")
        print("===============================")

        choice = input("Enter your choice (1-5): ")

        if choice == "1":
            search_fifa()

        elif choice == "2":
            search_ucl()

        elif choice == "3":
            print("\\n ALL FIFA WORLD CUP WINNERS:")
```


File Edit Selection View Go Run Terminal Help
Untitled (Workspace)
EXPLORER
OPEN EDITORS 1 unsaved
README.md Desktop • VIT_P...
football_searchsystem.py De...
UNTITLED (WORKS...
Desktop
.idea
academic-vit
acdshop
CSE-Intro to problem solving
ex-AM
fam
new
VIT
VIT_PROJECT
source_code
football_searchsystem.py
src
DATA.xlsx
desktop.ini
New Text Document.txt
PROJECT_Report.docx
PROJECT_Report.pdf
README.md
BlueJ.lnk
Counter-Strike 2.url
desktop.ini
DOC.2025.lnk
Downloads.lnk
Gmail.lnk
Grand Theft Auto V Enhanced [7...
IDLE (Python 3.11 64-bit).lnk
movi's.lnk
New WinRAR ZIP archive.zip
OUTLINE
TIMELINE
README.md
football_searchsystem.py
Desktop > VIT_PROJECT > source_code > football_searchsystem.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python
Python
Python
Python
Ln 26, Col 66 (23 selected) Spaces: 4 UTF-8 CRLF {} Python 3.11.3 Go Live Prettier

```python
# ---------------------------------------------
# FOOTBALL TOURNAMENT SEARCH SYSTEM
# FIFA WORLD CUP & UEFA CHAMPIONS LEAGUE WINNERS
# ---------------------------------------------

# FIFA World Cup Winners (1930-2022)
fifa_world_cup_winners = {1930: "Uruguay", 1934: "Italy", 1938: "Italy",
    1950: "Uruguay", 1954: "West Germany", 1958: "Brazil",
    1962: "Brazil", 1966: "England", 1970: "Brazil",
    1974: "West Germany", 1978: "Argentina", 1982: "Italy",
    1986: "Argentina", 1990: "West Germany", 1994: "Brazil",
    1998: "France", 2002: "Brazil", 2006: "Italy",
    2010: "Spain", 2014: "Germany", 2018: "France",
    2022: "Argentina"}

# UEFA Champions League Winners (2000-2025)
ucl_winners = {2000: "Real Madrid", 2001: "Bayern Munich", 2002: "Real Madrid",
    2003: "AC Milan", 2004: "Porto", 2005: "Liverpool",
    2006: "Barcelona", 2007: "AC Milan", 2008: "Manchester United",
    2009: "Barcelona", 2010: "Inter Milan", 2011: "Barcelona",
    2012: "Chelsea", 2013: "Bayern Munich", 2014: "Real Madrid",
    2015: "Barcelona", 2016: "Real Madrid", 2017: "Real Madrid",
    2018: "Real Madrid", 2019: "Liverpool", 2020: "Bayern Munich",
    2021: "Chelsea", 2022: "Real Madrid", 2023: "Manchester City",
    2024: "Real Madrid",2025:"Paris st. German"}
```

```
PS C:\Users\ayush\OneDrive\Desktop>
PS C:\Users\ayush\OneDrive\Desktop>
```