**Mini Project Report**

Project Title: **“ATHLETE HEALTH TRACKING AND APPOINTMENT SCHEDULER ”**

Student Reg. No: KUB23CSE121

Student Name: Sai Pawan

Name of Institution: Kishkinda University

Department Name: CSE

Year of Study: 2024

Date: Submission Date: 28-09-2024

**Table of Contents:**

Introduction

Objective

Methodology

Results/Findings

Conclusion

References Introduction

**1. Introduction :**

The "Athlete Health Tracking and Appointment Scheduler" project aims to address the complexities involved in managing healthcare appointments for athletes. With the increasing focus on athlete health and wellness, it has become crucial to streamline the scheduling of medical appointments and monitor health progress efficiently.

Athletes often face challenges in managing their medical appointments and tracking their recovery progress. Traditional methods can be cumbersome, leading to missed appointments, inadequate health monitoring, and inefficient use of resources. This project seeks to provide a digital solution that facilitates the scheduling of appointments and allows for the systematic tracking of an athlete's health journey.

The primary purpose of this project is to develop a user-friendly application that:

Simplifies Appointment Management: Allows users to create, read, update, and delete appointments for athletes.

Enhances Tracking of Health Progress: Enables athletes and healthcare providers to log and review health progress effectively.

Creation and management of athlete profiles.

Scheduling and management of medical appointments.

Tracking health progress through a structured approach.

Limited User Interface: The application relies on a command-line interface, which may not be intuitive for all users

No Database Integration: Data is stored in memory, meaning it will not persist after the program exits.

User Authentication: The current version lacks user authentication features, limiting security and data protection.

By addressing these issues, the project aims to serve as a foundation for more comprehensive solutions in athlete health management.

**2.Objective:**

Objective and Goals of the Project

The primary objective of the "Athlete Health Tracking and Appointment Scheduler" project is to create an efficient, user-friendly application that facilitates the management of medical appointments and the tracking of health progress for athletes. By automating these processes, the project aims to improve communication between athletes and healthcare providers, enhance scheduling accuracy, and promote better health outcomes.

Implement a robust CRUD (Create, Read, Update, Delete) functionality to manage clinic appointments seamlessly.

Allow easy retrieval and modification of appointment details to accommodate changes in schedules or athlete needs.

Develop a systematic approach for tracking athletes' health progress through detailed logging and retrieval of health data.

Enable athletes to easily add and review their health updates, ensuring continuous monitoring of recovery and wellness.

Create a method to manage all medical appointments associated with a specific athlete, providing an overview of their healthcare journey.

Ensure that athletes can quickly access their appointment history and upcoming schedules.

Design an intuitive command-line interface that guides users through the various functionalities, making it accessible even for those with minimal technical expertise.

Provide clear prompts and feedback to enhance user experience.

Employ object-oriented programming principles to create modular classes (Athlete, Appointment, Clinic), ensuring ease of maintenance and scalability.

**3.Methodology:**

1. **Programming** **Language:** Python
   * Chosen for its simplicity and readability, making it ideal for rapid development and prototyping.
2. **Development Environment:**

* Integrated Development Environment (IDE) such as PyCharm or VS Code to facilitate coding, debugging, and testing.

**3. Version Control:**

* Git for version control, enabling collaborative development and tracking changes in the codebase.

**4. Documentation:**

* Markdown for writing project documentation, ensuring clarity and accessibility.

**5. Data Structures:**

* Lists and dictionaries for efficient storage and retrieval of athlete and appointment information.

**4. Results**

# " Athlete Health Tracking and Appointment Scheduler "

class Athlete:

    def \_\_init\_\_(self, athlete\_id, name, age, sport):

        """

        Initializes an Athlete object.

        :param athlete\_id: Unique ID for the athlete.

        :param name: Name of the athlete.

        :param age: Age of the athlete.

        :param sport: Sport played by the athlete.

        """

        self.athlete\_id = athlete\_id

        self.name = name

        self.age = age

        self.sport = sport

        self.health\_progress = []  # List to track health progress

    def add\_health\_progress(self, progress\_data):

        """

        Adds health progress data for the athlete.

        :param progress\_data: A dictionary of health progress details.

        """

        self.health\_progress.append(progress\_data)

    def view\_health\_progress(self):

        """

        Returns the health progress of the athlete.

        """

        return self.health\_progress

class Appointment:

    def \_\_init\_\_(self, appointment\_id, athlete\_id, date, time, reason):

        """

        Initializes an Appointment object.

        :param appointment\_id: Unique ID for the appointment.

        :param athlete\_id: ID of the athlete for whom the appointment is scheduled.

        :param date: Date of the appointment.

        :param time: Time of the appointment.

        :param reason: Reason for the appointment (e.g., injury check, physio).

        """

        self.appointment\_id = appointment\_id

        self.athlete\_id = athlete\_id

        self.date = date

        self.time = time

        self.reason = reason

class Clinic:

    def \_\_init\_\_(self):

        """

        Initializes a Clinic object.

        Maintains dictionaries to store athletes and appointments.

        """

        self.athletes = {}  # Store athletes with athlete\_id as key

        self.appointments = {}  # Store appointments with appointment\_id as key

    def create\_appointment(self):

        """

        Creates a new appointment for an athlete based on user input.

        """

        appointment\_id = int(input("Enter Appointment ID: "))

        athlete\_id = int(input("Enter Athlete ID: "))

        date = input("Enter Appointment Date (YYYY-MM-DD): ")

        time = input("Enter Appointment Time (HH:MM): ")

        reason = input("Enter Appointment Reason (e.g., Injury Check, Physiotherapy): ")

        if athlete\_id in self.athletes:

            new\_appointment = Appointment(appointment\_id, athlete\_id, date, time, reason)

            self.appointments[appointment\_id] = new\_appointment

            print(f"Appointment {appointment\_id} created for athlete {athlete\_id}.")

        else:

            print(f"Athlete {athlete\_id} does not exist. Please add the athlete first.")

    def read\_appointment(self):

        """

        Retrieves and prints details of an appointment based on user input.

        """

        appointment\_id = int(input("Enter Appointment ID: "))

        appointment = self.appointments.get(appointment\_id)

        if appointment:

            print(f"Appointment Details: {appointment.date}, {appointment.time}, {appointment.reason}")

        else:

            print(f"Appointment {appointment\_id} not found.")

    def update\_appointment(self):

        """

        Updates an existing appointment based on user input.

        """

        appointment\_id = int(input("Enter Appointment ID: "))

        appointment = self.appointments.get(appointment\_id)

        if appointment:

            date = input("Enter new date (leave blank to keep the current date): ")

            time = input("Enter new time (leave blank to keep the current time): ")

            reason = input("Enter new reason (leave blank to keep the current reason): ")

            if date:

                appointment.date = date

            if time:

                appointment.time = time

            if reason:

                appointment.reason = reason

            print(f"Appointment {appointment\_id} updated.")

        else:

            print(f"Appointment {appointment\_id} not found.")

    def delete\_appointment(self):

        """

        Deletes an appointment based on user input.

        """

        appointment\_id = int(input("Enter Appointment ID: "))

        if appointment\_id in self.appointments:

            del self.appointments[appointment\_id]

            print(f"Appointment {appointment\_id} deleted.")

        else:

            print(f"Appointment {appointment\_id} not found.")

    def manage\_medical\_appointments(self):

        """

        Lists all appointments for a specific athlete based on user input.

        """

        athlete\_id = int(input("Enter Athlete ID: "))

        athlete\_appointments = [appt for appt in self.appointments.values() if appt.athlete\_id == athlete\_id]

        if athlete\_appointments:

            for a in athlete\_appointments:

                print(f"Appointment on {a.date} at {a.time} for {a.reason}")

        else:

            print(f"No appointments found for athlete {athlete\_id}.")

    def track\_health\_progress(self):

        """

        Tracks and updates the health progress of an athlete based on user input.

        """

        athlete\_id = int(input("Enter Athlete ID: "))

        progress\_data = input("Enter health progress details (e.g., Improving flexibility): ")

        athlete = self.athletes.get(athlete\_id)

        if athlete:

            athlete.add\_health\_progress({"date": "2024-09-25", "progress": progress\_data})

            print(f"Health progress for athlete {athlete\_id} updated.")

        else:

            print(f"Athlete {athlete\_id} not found.")

    def add\_athlete(self):

        """

        Adds a new athlete to the system based on user input.

        """

        athlete\_id = int(input("Enter Athlete ID: "))

        name = input("Enter Athlete Name: ")

        age = int(input("Enter Athlete Age: "))

        sport = input("Enter Athlete Sport: ")

        if athlete\_id not in self.athletes:

            self.athletes[athlete\_id] = Athlete(athlete\_id, name, age, sport)

            print(f"Athlete {athlete\_id} added.")

        else:

            print(f"Athlete {athlete\_id} already exists.")

    def get\_athlete(self):

        """

        Retrieves details of an athlete based on user input.

        """

        athlete\_id = int(input("Enter Athlete ID: "))

        athlete = self.athletes.get(athlete\_id)

        if athlete:

            print(f"Athlete {athlete\_id}: {athlete.name}, Age: {athlete.age}, Sport: {athlete.sport}")

        else:

            print(f"Athlete {athlete\_id} not found.")

# Initialize the Clinic

clinic = Clinic()

# Example of user interaction:

while True:

    print("\n1. Add Athlete")

    print("2. Create Appointment")

    print("3. View Appointment")

    print("4. Update Appointment")

    print("5. Delete Appointment")

    print("6. List Athlete's Appointments")

    print("7. Track Athlete Health Progress")

    print("8. Get Athlete Info")

    print("9. Exit")

    choice = input("Enter your choice: ")

    if choice == '1':

        clinic.add\_athlete()

    elif choice == '2':

        clinic.create\_appointment()

    elif choice == '3':

        clinic.read\_appointment()

    elif choice == '4':

        clinic.update\_appointment()

    elif choice == '5':

        clinic.delete\_appointment()

    elif choice == '6':

        clinic.manage\_medical\_appointments()

    elif choice == '7':

        clinic.track\_health\_progress()

    elif choice == '8':

        clinic.get\_athlete()

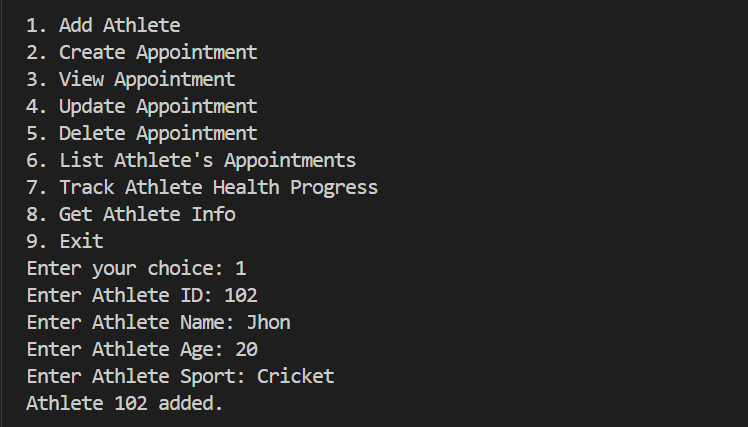
    elif choice == '9':

        break

    else:

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

**4. OUTPUT**

****

**5.Conclusion :**

1. **CRUD Operations for Clinic Appointments**: Efficiently handle the creation, reading, updating, and deletion of appointments.
2. **Manage Medical Appointments for Athletes**: Provide a focused interface to manage medical appointments using athlete IDs.
3. **Track Health and Recovery Progress**: Monitor and report on the health and recovery progress of athletes.

**6.References :**

**Books :**

1. "Wearable Technology in Sports: Principles and Practices" by Anupama S. Chaudhary
2. "Sports Analytics: A Guide for Coaches, Managers, and Other Decision Makers" by Benjamin Alamar

**Software** **Requirements** :

* Python 3.12.6 (64-bits)
* Visual Studio Code

**Hardware** **Requirements** :

* Processor : Intel i5 12th Gen
* Ram : 8 GB
* Hard Drive : 512 GB SSD