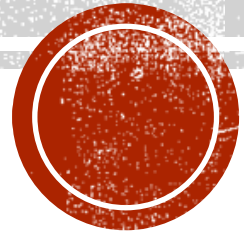
ATHLET MEDIA APPREARNCE SCHEDULER POC :



TEAM BYTE HOGS

TEJASHWINI VR : 3BR22CS173

SNEHA DEVALE : 3BR22CS164

SHRIDEVI : 3BR22CS158

YASHODHA : 3BR22CS187

INTRODUCTION TO THE PROJECT :

- **Problem Statement:**
    - The challenge of efficiently managing media appearances for athletes is a complex one, requiring careful coordination and organization.
    - Without a centralized system, scheduling conflicts and communication gaps can arise, leading to missed opportunities and disorganized appearances.
- **Objective:**
    - To develop a Proof of Concept (POC) for an Athlete Media Appearance Scheduler using object-oriented programming (OOP) and data structures and algorithms (DSA) principles in Python.

The POC aims to demonstrate the feasibility and functionality of such a system, focusing on CRUD operations for media schedules, organizing appearances for athletes, and managing relationships with media outlets. This project aims to develop a Proof-of-Concept (POC) application in Python to streamline the scheduling of media appearances for athletes. It leverages the power of Object-Oriented Programming (OOP) and Data Structures and Algorithms (DSA) principles.

CONTENT

- organize_ media_appearances (athlete_id): Organize media appearances for athletes.

1.Classes Defined

- MediaSchedule - Represents a media schedule.
- Athlete - Represents an athlete.
- Scheduler - Central class for managing schedules and relationships.
- Unittest class- to test a unit of source code.

2. Key Functionalities- CRUD operations for schedule,sorting and organizing schedules,managing relationships with media outlets.

3. Data Structures Used

- Lists for storing collections of objects.
- Dictionaries for efficient look-up and relationships

MODULE DESCRIPTION :

- The Athlete class : it provides a blueprint for creating objects that represent athletes. Each instance of this class encapsulates information about a specific athlete, including their unique identifier (athlete_id), name (name), and the sport (sport) they participate in.
- The MediaSchedule class : is designed to manage and organize media schedules associated with individual athletes within a software system. It encapsulates information related to scheduled media appearances or events for athletes, facilitating efficient tracking and management of media engagements.
- The Scheduler class : serves as a central component for managing scheduling operations within a software system. It provides functionality for organizing various schedules, including media engagements, events, or appointments. Additionally, it facilitates the coordination of schedules for athletes and media outlets.

- The TestScheduler class: is a vital component in software testing environments. It's designed to facilitate the scheduling, organization, and execution of tests within a testing framework or environment. This class serves as a central hub for managing various test cases, ensuring they run efficiently, and collecting results for analysis.
- create_media_schedule(schedule_id, athlete_id, date, media_outlet): Creates a new media schedule entry with the provided details and adds it to the list of schedules.
- read_media_schedule(schedule_id): Retrieves the media schedule entry corresponding to the given schedule ID.
- update_media_schedule(schedule_id, updated_data): Updates the details of the media schedule entry identified by the given schedule ID with the provided updated data.

- delete_media_schedule(schedule_id): Deletes the media schedule entry associated with the given schedule ID from the list of schedules.

  Organize_media_appearances(athlete_id): Retrieves all media appearances scheduled for the athlete with the provided athlete ID.

- setUp(self): This method is called before each test method to set up the testing environment. It creates an instance of the Scheduler class and initializes it with sample data, including media schedules and athlete information.
- test_create_media_schedule(self): This method tests the create_media_schedule method of the Scheduler class. It verifies whether the method correctly adds new media schedule entries to the scheduler's list of schedules.

```
60      def test_create_media_schedule(self):
61          self.assertEqual(len(self.scheduler.schedules), 3)
62          print("\nMedia Schedules:")
63          for schedule in self.scheduler.schedules:
64              print(f"Schedule ID: {schedule.schedule_id}, Athlete ID: {schedule.athlete_id}, Date: {schedule.date}, Media Outlet: {schedule.media_outlet}")
65
```

- test_read_media_schedule(self): This method tests the read_media_schedule method of the Scheduler class. It checks whether the method correctly retrieves a media schedule entry based on the provided schedule ID.

```
65
66      def test_read_media_schedule(self):
67          schedule = self.scheduler.read_media_schedule(1)
68          self.assertEqual(schedule.date, "2024-05-01")
69          print("\nRead Media Schedule:")
70          print(f"Schedule ID: {schedule.schedule_id}, Athlete ID: {schedule.athlete_id}, Date: {schedule.date}, Media Outlet: {schedule.media_outlet}")
71
```

- test_update_media_schedule(self): This method tests the update_media_schedule method of the Scheduler class. It verifies whether the method correctly updates the details of a media schedule entry based on the provided schedule ID and updated data.

```
72      def test_update_media_schedule(self):
73          self.scheduler.update_media_schedule(2, {'date': "2024-05-05"})
74          schedule = self.scheduler.read_media_schedule(2)
75          self.assertEqual(schedule.date, "2024-05-05")
76          print("\nUpdated Media Schedule:")
77          print(f"Schedule ID: {schedule.schedule_id}, Athlete ID: {schedule.athlete_id}, Date: {schedule.date}, Media Outlet: {schedule.media_outlet}")
```

- test_delete_media_schedule(self): This method tests the delete_media_schedule method of the Scheduler class. It checks whether the method correctly removes a media schedule entry from the scheduler's list of schedules based on the provided schedule ID.

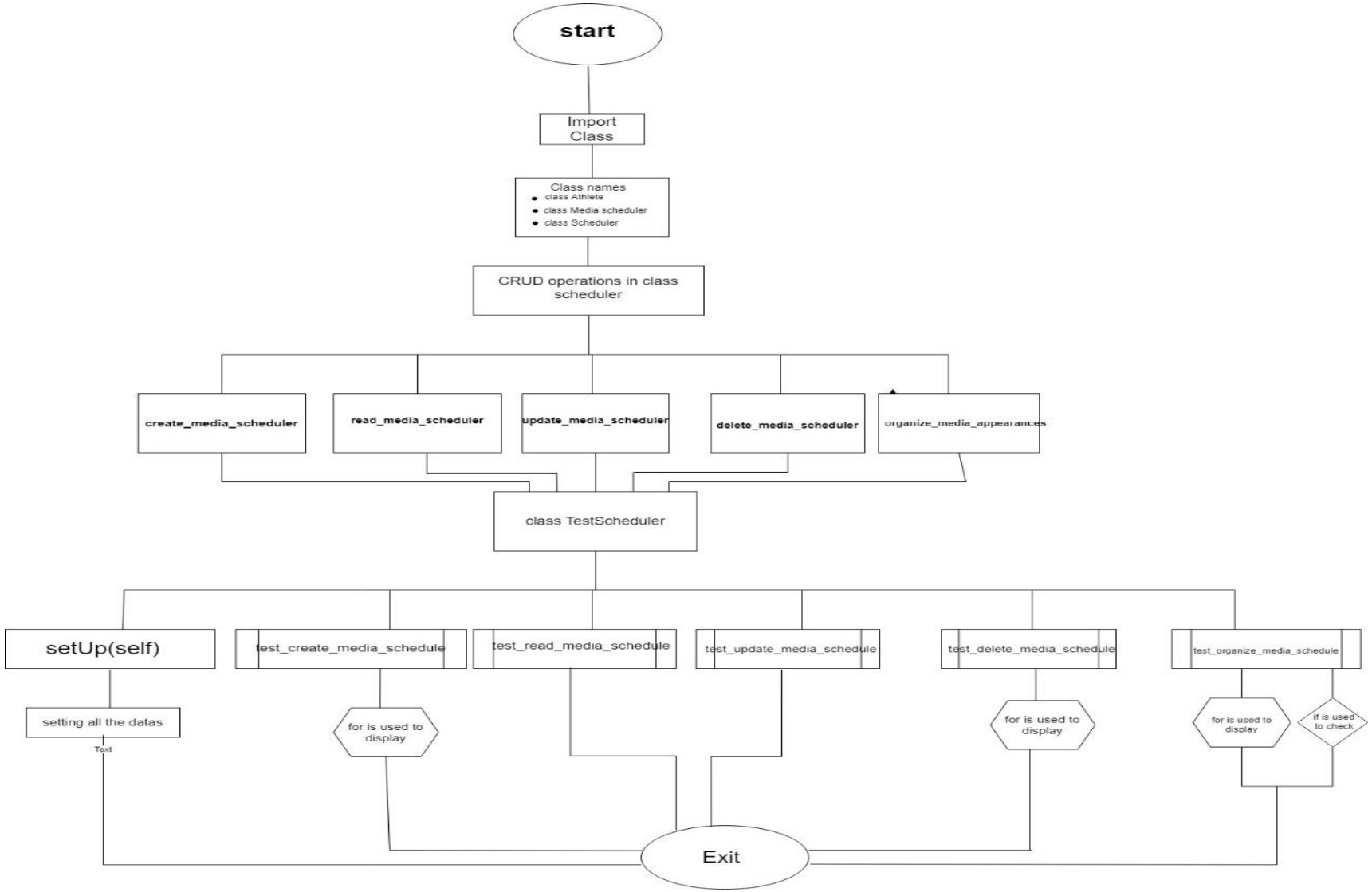
```
79  def test_delete_media_schedule(self):
80      self.scheduler.delete_media_schedule(1)
81      self.assertEqual(len(self.scheduler.schedules), 2)
82      print("\nRemaining Media Schedules:")
83      for schedule in self.scheduler.schedules:
84          print(f"Schedule ID: {schedule.schedule_id}, Athlete ID: {schedule.athlete_id}, Date: {schedule.date}, Media Outlet: {schedule.media_outlet}")
85
```

- test_organize_media_appearances(self): This method tests the organize_media_appearances method of the Scheduler class. It verifies whether the method correctly retrieves all media appearances scheduled for a specific athlete based on the provided athlete ID.

```
86      def test_organize_media_appearances(self):
87          athlete_id = 2
88          athlete_media_appearances = self.scheduler.organize_media_appearances(athlete_id)
89          if isinstance(athlete_media_appearances, list):
90              print(f"\nMedia Appearances for Athlete ID {athlete_id}:")
91              for appearance in athlete_media_appearances:
92                  print(f"Schedule ID: {appearance.schedule_id}, Date: {appearance.date}, Media Outlet: {appearance.media_outlet}")
93          else:
94              print(athlete_media_appearances)
95
```

- unittest.main() is a method provided by the unittest module, which is Python's built-in testing framework.The verbosity parameter controls the amount of detail displayed in the test results output.
- It accepts three levels of verbosity:
- 0: Quiet mode, only displays the total number of tests and errors.
- 1: Default mode, displays a dot for each successful test and F for each failed test, along with the total counts.
- 2: Verbose mode, displays the name of each test and its result, including successful tests.In your script, unittest.main(verbosity=0) is set to quiet mode (verbosity=0), meaning it will only show the total number of tests run and any errors encountered, without detailed test-by-test output.

FLOW CHART:


start
Import Class
Class names
class Athlete
class Media scheduler
class Scheduler
CRUD operations in class scheduler
create_media_scheduler
read_media_scheduler
update_media_scheduler
delete_media_scheduler
organize_media_appearances
class TestScheduler
setUp(self)
test_create_media_schedule
test_read_media_schedule
test_update_media_schedule
test_delete_media_schedule
test_organize_media_schedule
setting all the datas
Text
for is used to display
for is used to display
for is used to display
if is used to check
Exit

```python
import unittest

class Athlete:
    def __init__(self, athlete_id, name, sport):
        self.athlete_id = athlete_id
        self.name = name
        self.sport = sport

class MediaSchedule:
    def __init__(self, schedule_id, athlete_id, date, media_outlet):
        self.schedule_id = schedule_id
        self.athlete_id = athlete_id
        self.date = date
        self.media_outlet = media_outlet

class Scheduler:
    def __init__(self):
        self.schedules = []
        self.media_outlets = {}
        self.athletes = []

```

```python

    def create_media_schedule(self, schedule_id, athlete_id, date, media_outlet):
        self.schedules.append(MediaSchedule(schedule_id, athlete_id, date, media_outlet))

    def read_media_schedule(self, schedule_id):
        for schedule in self.schedules:
            if schedule.schedule_id == schedule_id:
                return schedule
        raise ValueError("Schedule ID not found")

    def update_media_schedule(self, schedule_id, updated_data):
        for schedule in self.schedules:
            if schedule.schedule_id == schedule_id:
                schedule.date = updated_data.get('date', schedule.date)
                schedule.media_outlet = updated_data.get('media_outlet', schedule.media_outlet)
                break

    def delete_media_schedule(self, schedule_id):
        self.schedules = [schedule for schedule in self.schedules if schedule.schedule_id != schedule_id]

```

```
41      def organize_media_appearances(self, athlete_id):
42          athlete_schedules = []
43          for schedule in self.schedules:
44              if schedule.athlete_id == athlete_id:
45                  athlete_schedules.append(schedule)
46          if not athlete_schedules:
47              return "No media appearances found for this athlete."
48          return athlete_schedules
49
50  class TestScheduler(unittest.TestCase):
51      def setUp(self):
52          self.scheduler = Scheduler()
53          self.scheduler.create_media_schedule(1, 1, "2024-05-01", "dd")
54          self.scheduler.create_media_schedule(2, 2, "2024-05-02", "star sports")
55          self.scheduler.create_media_schedule(3, 3, "2024-05-03", "sports +")
56          self.scheduler.athletes.append(Athlete(1, "aaaa", "Football"))
57          self.scheduler.athletes.append(Athlete(2, "bbbb", "cricket"))
58          self.scheduler.athletes.append(Athlete(3, "cccc", "volliball"))
59
```

```
60      def test_create_media_schedule(self):
61          self.assertEqual(len(self.scheduler.schedules), 3)
62          print("\nMedia Schedules:")
63          for schedule in self.scheduler.schedules:
64              print(f"Schedule ID: {schedule.schedule_id}, Athlete ID: {schedule.athlete_id}, Date: {schedule.date}, Media Outlet: {schedule.media_outlet}")
65
66      def test_read_media_schedule(self):
67          schedule = self.scheduler.read_media_schedule(1)
68          self.assertEqual(schedule.date, "2024-05-01")
69          print("\nRead Media Schedule:")
70          print(f"Schedule ID: {schedule.schedule_id}, Athlete ID: {schedule.athlete_id}, Date: {schedule.date}, Media Outlet: {schedule.media_outlet}")
71
72      def test_update_media_schedule(self):
73          self.scheduler.update_media_schedule(2, {'date': "2024-05-05"})
74          schedule = self.scheduler.read_media_schedule(2)
75          self.assertEqual(schedule.date, "2024-05-05")
76          print("\nUpdated Media Schedule:")
77          print(f"Schedule ID: {schedule.schedule_id}, Athlete ID: {schedule.athlete_id}, Date: {schedule.date}, Media Outlet: {schedule.media_outlet}")
78
79      def test_delete_media_schedule(self):
80          self.scheduler.delete_media_schedule(1)
81          self.assertEqual(len(self.scheduler.schedules), 2)
82          print("\nRemaining Media Schedules:")
83          for schedule in self.scheduler.schedules:
84              print(f"Schedule ID: {schedule.schedule_id}, Athlete ID: {schedule.athlete_id}, Date: {schedule.date}, Media Outlet: {schedule.media_outlet}")
85
```

```
    def test_organize_media_appearances(self):
        athlete_id = 2
        athlete_media_appearances = self.scheduler.organize_media_appearances(athlete_id)
        if isinstance(athlete_media_appearances, list):
            print(f"\nMedia Appearances for Athlete ID {athlete_id}:")
            for appearance in athlete_media_appearances:
                print(f"Schedule ID: {appearance.schedule_id}, Date: {appearance.date}, Media Outlet: {appearance.media_outlet}")
        else:
            print(athlete_media_appearances)

if __name__ == '__main__':
    unittest.main(verbosity=0)

```

```
>>> %Run 'python project.py'

  test_create_media_schedule (__main__.TestScheduler) ...
  Media Schedules:
  Schedule ID: 1, Athlete ID: 1, Date: 2024-05-01, Media Outlet: dd
  Schedule ID: 2, Athlete ID: 2, Date: 2024-05-02, Media Outlet: star sports
  Schedule ID: 3, Athlete ID: 3, Date: 2024-05-03, Media Outlet: sports +
  ok
  test_delete_media_schedule (__main__.TestScheduler) ...
  Remaining Media Schedules:
  Schedule ID: 2, Athlete ID: 2, Date: 2024-05-02, Media Outlet: star sports
  Schedule ID: 3, Athlete ID: 3, Date: 2024-05-03, Media Outlet: sports +
  ok
  test_organize_media_appearances (__main__.TestScheduler) ...
  Media Appearances for Athlete ID 2:
  Schedule ID: 2, Date: 2024-05-02, Media Outlet: star sports
  ok
  test_read_media_schedule (__main__.TestScheduler) ...
  Read Media Schedule:
  Schedule ID: 1, Athlete ID: 1, Date: 2024-05-01, Media Outlet: dd
  ok
  test_update_media_schedule (__main__.TestScheduler) ...
  Updated Media Schedule:
  Schedule ID: 2, Athlete ID: 2, Date: 2024-05-05, Media Outlet: star sports
  ok

  ----------------------------------------------------------------------
  Ran 5 tests in 0.085s

  OK

  Process ended with exit code 0.
```

SUMMARY

- The Athlete class represents an athlete with attributes such as athlete_id, name, and sport.
- The MediaScheduler class represents a media schedule with attributes such as schedule_id, athlete_id, date, and media_outlet.
- The Scheduler class manages media schedules and athletes.
- It has methods to create, read, update, and delete media schedules, as well as organize media appearances for a specific athlete.
- The TestScheduler class contains unit tests for the methods in the Scheduler class to ensure they work as expected.
- Overall, the code provides a basic framework for managing media schedules for athletes, including functionality for CRUD operations and organizing media appearances.

BIBLIOGRAPHY

- https://chat.openai.com/
- https://apphttps://app.diagrams.net/.diagrams.net/
- Googlr colab
- Thonny
- Google

GITHUB ACCOUNT :

- https://github.com/shridevi-23
- https://github.com/tejashwinivr/BITM
- https://github.com/Snehadevale
- https://github.com/yashodhakampli

Thank you

Any question?