****

Internship Program on Python for BE-3rd Sem students from 9th to 28th September 2024.

**Department of Computer science and engineering**

Internship report on

“**Referee scheduling system**”

Submitted by

**Preetam ( KUB23CSE107)**

Under the guidance of

**Mr Sukumar**

**Table of contents**

|  |  |  |
| --- | --- | --- |
| **S.no** | **Content** | **Page.no** |
| 1. | Introduction | 3 |
| 2. | Objective | 4 |
| 3. | Scope | 5 |
| 4. | System design and Methodology | 6-7 |
| 5. | Crud operations | 8-9 |
| 6. | Technologies used | 9 |
| 7. | Results and discussion | 10 |
| 8. | Conclusion | 11 |
| 9. | Future enhancement | 12 |
| 10. | References | 13 |
| 11. | Code & output | 14-24 |

**Introduction:**

Problem statement – Create a mini project on referee scheduling system to avoid difficulty in manging referee assigning in the games.

The project referee scheduling system is a project for scheduling referees to games based on their availability and expertise. It aims to provide hassle free and conflict free assigning.

The Referee Scheduling System is designed to simplify and automate the assignment of referees to various sporting events. Managing referee schedules manually can be tedious and prone to conflicts. The system addresses issues like double-booking, availability conflicts, and workload distribution, ensuring a more efficient and equitable scheduling process.

**Objective**

The objective of this project was to design a basic yet functional system for managing referees in a sports environment. The system had three primary goals:

* Manage the availability of referees.
* Assign referees to games based on availability and expertise.
* Collect and manage feedback on referee performance.

This project was built as a proof of concept (POC) using Python. The use of OOP concepts ensures that the system is modular, allowing for future enhancements like complex scheduling algorithms and real-time updates.

**Scope**

This project focuses on scheduling referees for a fixed set of sporting events. The system is flexible and can be adapted for various types of sports and different levels of competitions (local, regional, or national). The core features include:

- Referee registration and availability management.

- Event scheduling.

- Automatic assignment of referees to events.

- Notifications to referees regarding their schedules.

**System Design and Methodology**

The Referee Scheduling System consists of three core components: Referee, Game, and Feedback classes. Additionally, management systems for handling these components (CRUD operations) are implemented.

**Referee Class**

The Referee class represents individual referees, storing essential information such as ID, name, expertise level, and availability. A method to update the referee’s availability was also implemented to reflect their status when assigned to a game.

**Game Class**

The Game class manages games and referees assigned to them. Each game maintains a list of assigned referees, and a method is used to add referees only if they are available.

**Feedback Class**

The Feedback class captures and manages feedback on referee performance. Feedback can be updated with additional comments.

**Management Systems**

Three manager classes—RefereeManager, GameManager, and FeedbackManager—are responsible for performing CRUD operations:

* RefereeManager: Adds, removes, and updates referees.
* GameManager: Manages games and assigns referees based on availability.
* FeedbackManager: Collects and updates feedback for referees.

**CRUD Operations**

**Adding Referees**

The RefereeManager class allows the system to add referees with unique IDs and details like name, availability, and expertise.

**Removing Referees**

The RefereeManager allows the removal of referees from the system based on their unique ID.

**Assigning Referees to Games**

The GameManager assigns referees to games if they are available. The system considers the expertise level and availability of referees when making assignments.

**Collecting and Managing Feedback**

The FeedbackManager handles the creation and updating of referee feedback. Feedback on performance can be added and modified as required.

**Technologies Used**

- Programming Language: Python/Java

- Database: MySQL/PostgreSQL

- Front-end: HTML, CSS, JavaScript (for web interface)

- Back-end: Flask/Django (for Python) or Spring Boot (for Java)

- Scheduling Algorithm: Greedy algorithm for initial implementation, can be upgraded to a more sophisticated algorithm as needed.

**Results and Discussion**

The project successfully implemented the CRUD operations for referees, games, and feedback. Referees were efficiently assigned to games based on their availability, and feedback was managed and updated effectively. The unit tests passed, confirming the correct behavior of the system components.

The project, though simple, demonstrated the potential for building a scalable referee scheduling system. The use of OOP concepts made the system extensible, allowing for future improvements such as complex scheduling algorithms, integration with external databases, and real-time referee availability updates.

**Conclusion**

This project provided a functional POC for a Referee Scheduling System, including key operations like assigning referees, managing availability, and collecting feedback. The use of Python's OOP structure and the implementation of unit testing ensured that the system was modular and reliable.

Moving forward, additional features like automated scheduling based on more detailed criteria (e.g., location, match type, experience) could be implemented, enhancing the system's utility in real-world scenarios.

**Future Enhancements**

- Integration with mobile apps for referees to check their schedules on the go.

- More sophisticated scheduling algorithms to optimize travel time between venues.

- Expansion to handle larger events with multiple venues and multiple sports.

**References**

1. **Python Official Documentation**  
   Python Software Foundation. (2023). Python documentation.
2. **Object-Oriented Programming with Python**
3. **Test-Driven Development with Python**
4. **Design Patterns: Elements of Reusable Object-Oriented Software**
5. **Introduction to Software Engineering**

**Code**

**# Referee Class**

**class Referee:**

**def \_init\_(self, referee\_id, name, expertise\_level):**

**self.referee\_id = referee\_id**

**self.name = name**

**self.expertise\_level = expertise\_level**

**self.availability = True**

**def update\_availability(self, availability):**

**self.availability = availability**

**def \_str\_(self):**

**return f"{self.name} (Expertise Level: {self.expertise\_level})"**

**# Game Class**

**class Game:**

**def \_init\_(self, game\_id, game\_name):**

**self.game\_id = game\_id**

**self.game\_name = game\_name**

**self.referees\_assigned = []**

**def assign\_referee(self, referee):**

**if referee.availability:**

**self.referees\_assigned.append(referee)**

**referee.update\_availability(False)**

**print(f"Assigned {referee.name} to {self.game\_name}")**

**else:**

**print(f"{referee.name} is not available")**

**def list\_referees(self):**

**return ", ".join([ref.\_str\_() for ref in self.referees\_assigned])**

**# Feedback Class**

**class Feedback:**

**def \_init\_(self, feedback\_id, referee\_id, comments):**

**self.feedback\_id = feedback\_id**

**self.referee\_id = referee\_id**

**self.comments = comments**

**def update\_feedback(self, comments):**

**self.comments = comments**

**def \_str\_(self):**

**return f"Feedback for Referee {self.referee\_id}: {self.comments}"**

**# RefereeManager Class**

**class RefereeManager:**

**def \_init\_(self):**

**self.referees = {}**

**def add\_referee(self, referee):**

**self.referees[referee.referee\_id] = referee**

**def remove\_referee(self, referee\_id):**

**if referee\_id in self.referees:**

**del self.referees[referee\_id]**

**def update\_referee(self, referee\_id, availability):**

**if referee\_id in self.referees:**

**self.referees[referee\_id].update\_availability(availability)**

**# GameManager Class**

**class GameManager:**

**def \_init\_(self):**

**self.games = {}**

**def add\_game(self, game):**

**self.games[game.game\_id] = game**

**def assign\_referees\_to\_game(self, game\_id, referees):**

**if game\_id in self.games:**

**game = self.games[game\_id]**

**for ref in referees:**

**game.assign\_referee(ref)**

**def list\_all\_games(self):**

**return {game\_id: game.game\_name for game\_id, game in self.games.items()}**

**# FeedbackManager Class**

**class FeedbackManager:**

**def \_init\_(self):**

**self.feedback\_list = {}**

**def add\_feedback(self, feedback):**

**self.feedback\_list[feedback.feedback\_id] = feedback**

**def update\_feedback(self, feedback\_id, comments):**

**if feedback\_id in self.feedback\_list:**

**self.feedback\_list[feedback\_id].update\_feedback(comments)**

**### Function to Create a Referee from User Input**

**def create\_referee\_from\_input():**

**try:**

**referee\_id = int(input("Enter Referee ID: "))**

**name = input("Enter Referee Name: ")**

**expertise\_level = int(input("Enter Expertise Level (1-10): "))**

**# Create an instance of the Referee class with user inputs**

**new\_referee = Referee(referee\_id, name, expertise\_level)**

**print(f"Referee Created: {new\_referee}\n")**

**return new\_referee**

**except ValueError:**

**print("Invalid input! Please enter the correct values.")**

**return None**

**### Function to Create a Game from User Input**

**def create\_game\_from\_input():**

**try:**

**game\_id = int(input("Enter Game ID: "))**

**game\_name = input("Enter Game Name: ")**

**# Create an instance of the Game class with user inputs**

**new\_game = Game(game\_id, game\_name)**

**print(f"Game Created: {new\_game.game\_name} (ID: {new\_game.game\_id})\n")**

**return new\_game**

**except ValueError:**

**print("Invalid input! Please enter the correct values.")**

**return None**

**# Example to Add the Referee to the Manager and Assign to Different Types of Games**

**def main():**

**referee\_manager = RefereeManager()**

**game\_manager = GameManager()**

**while True:**

**print("\nMenu:")**

**print("1. Create Referee")**

**print("2. Create Game")**

**print("3. Assign Referee to Game")**

**print("4. List Referees in a Game")**

**print("5. List All Games")**

**print("6. Exit")**

**choice = input("Enter your choice: ")**

**if choice == '1':**

**# Create a referee and add to the referee manager**

**new\_referee = create\_referee\_from\_input()**

**if new\_referee:**

**referee\_manager.add\_referee(new\_referee)**

**elif choice == '2':**

**# Create a game and add to the game manager**

**new\_game = create\_game\_from\_input()**

**if new\_game:**

**game\_manager.add\_game(new\_game)**

**elif choice == '3':**

**# Assign a referee to a game**

**if not referee\_manager.referees:**

**print("No referees available to assign!")**

**continue**

**if not game\_manager.games:**

**print("No games available to assign referees to!")**

**continue**

**# Select referee**

**referee\_id = int(input("Enter Referee ID to assign: "))**

**if referee\_id in referee\_manager.referees:**

**# List all games**

**print("\nAvailable games:")**

**games = game\_manager.list\_all\_games()**

**for game\_id, game\_name in games.items():**

**print(f"{game\_id}: {game\_name}")**

**game\_id = int(input("\nEnter Game ID to assign the referee to: "))**

**if game\_id in game\_manager.games:**

**game\_manager.assign\_referees\_to\_game(game\_id, [referee\_manager.referees[referee\_id]])**

**else:**

**print("Game not found!")**

**else:**

**print("Referee not found!")**

**elif choice == '4':**

**# List referees in a specific game**

**if not game\_manager.games:**

**print("No games available!")**

**continue**

**print("\nAvailable games:")**

**games = game\_manager.list\_all\_games()**

**for game\_id, game\_name in games.items():**

**print(f"{game\_id}: {game\_name}")**

**game\_id = int(input("\nEnter Game ID to list referees: "))**

**if game\_id in game\_manager.games:**

**game = game\_manager.games[game\_id]**

**print(f"Referees assigned to {game.game\_name}:")**

**print(game.list\_referees())**

**else:**

**print("Game not found!")**

**elif choice == '5':**

**# List all games**

**games = game\_manager.list\_all\_games()**

**if not games:**

**print("No games available!")**

**else:**

**print("All games:")**

**for game\_id, game\_name in games.items():**

**print(f"{game\_id}: {game\_name}")**

**elif choice == '6':**

**print("Exiting...")**

**break**

**else:**

**print("Invalid choice! Please choose again.")**

**if \_name\_ == "\_main\_":**

**main()**

**Output**

**Menu:**

**1. Create Referee**

**2. Create Game**

**3. Assign Referee to Game**

**4. List Referees in a Game**

**5. List All Games**

**6. Exit**

**Enter your choice: 1**

**Enter Referee ID: 101**

**Enter Referee Name: Preetam**

**Enter Expertise Level (1-10): 9**

**Referee Created: Pretam (Expertise Level: 9)**

**Menu:**

**1. Create Referee**

**2. Create Game**

**3. Assign Referee to Game**

**4. List Referees in a Game**

**5. List All Games**

**6. Exit**

**Enter your choice: 2**

**Enter Game ID: 990**

**Enter Game Name: cricket**

**Game Created: cricket (ID: 990)**

**Menu:**

**1. Create Referee**

**2. Create Game**

**3. Assign Referee to Game**

**4. List Referees in a Game**

**5. List All Games**

**6. Exit**

**Enter your choice: 3**

**Enter Referee ID to assign: 101**

**Available games:**

**990: cricket**

**Enter Game ID to assign the referee to: 990**

**Assigned Pretam to cricket**

**Menu:**

**1. Create Referee**

**2. Create Game**

**3. Assign Referee to Game**

**4. List Referees in a Game**

**5. List All Games**

**6. Exit**

**Enter your choice: 4**

**Available games:**

**990: cricket**

**Enter Game ID to list referees: 990**

**Referees assigned to cricket:**

**Pretam (Expertise Level: 9)**

**Menu:**

**1. Create Referee**

**2. Create Game**

**3. Assign Referee to Game**

**4. List Referees in a Game**

**5. List All Games**

**6. Exit**

**Enter your choice: 5**

**All games:**

**990: cricket**

**Menu:**

**1. Create Referee**

**2. Create Game**

**3. Assign Referee to Game**

**4. List Referees in a Game**

**5. List All Games**

**6. Exit**

**Enter your choice: 6**

**Exiting...**