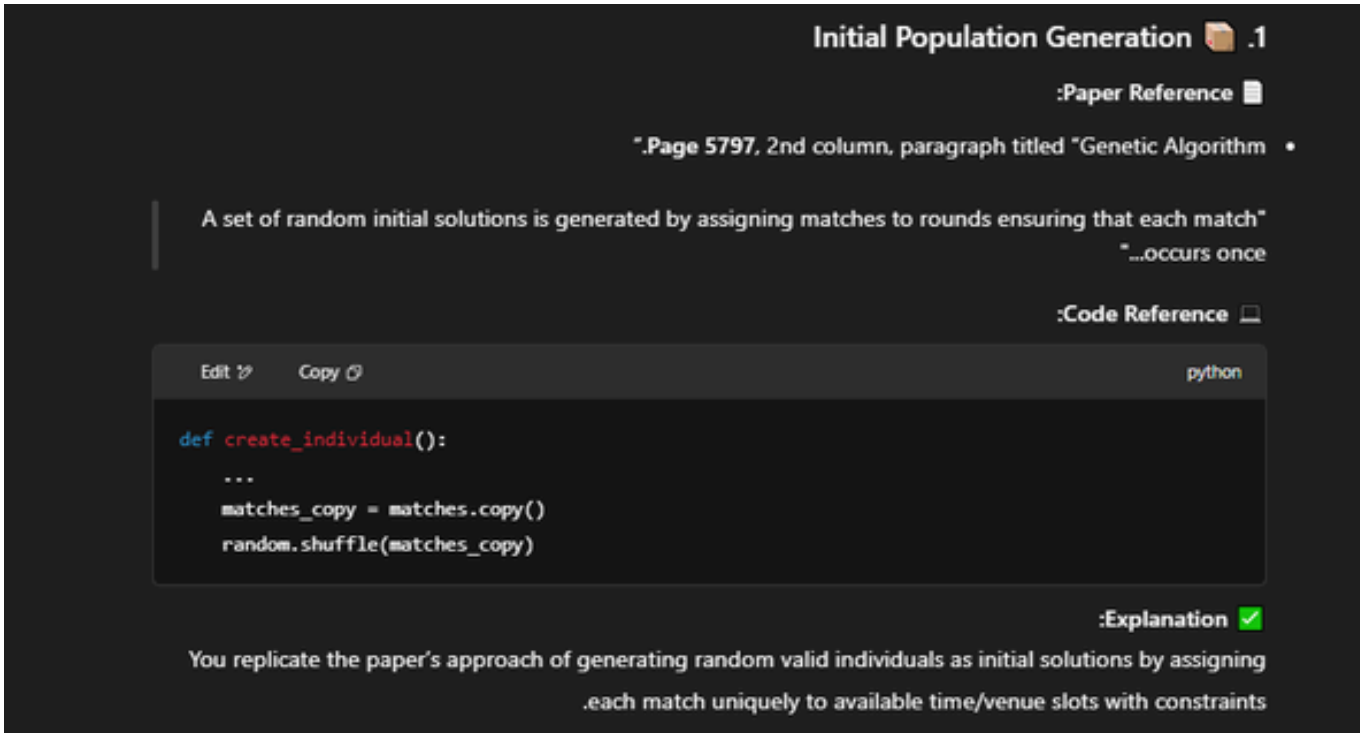
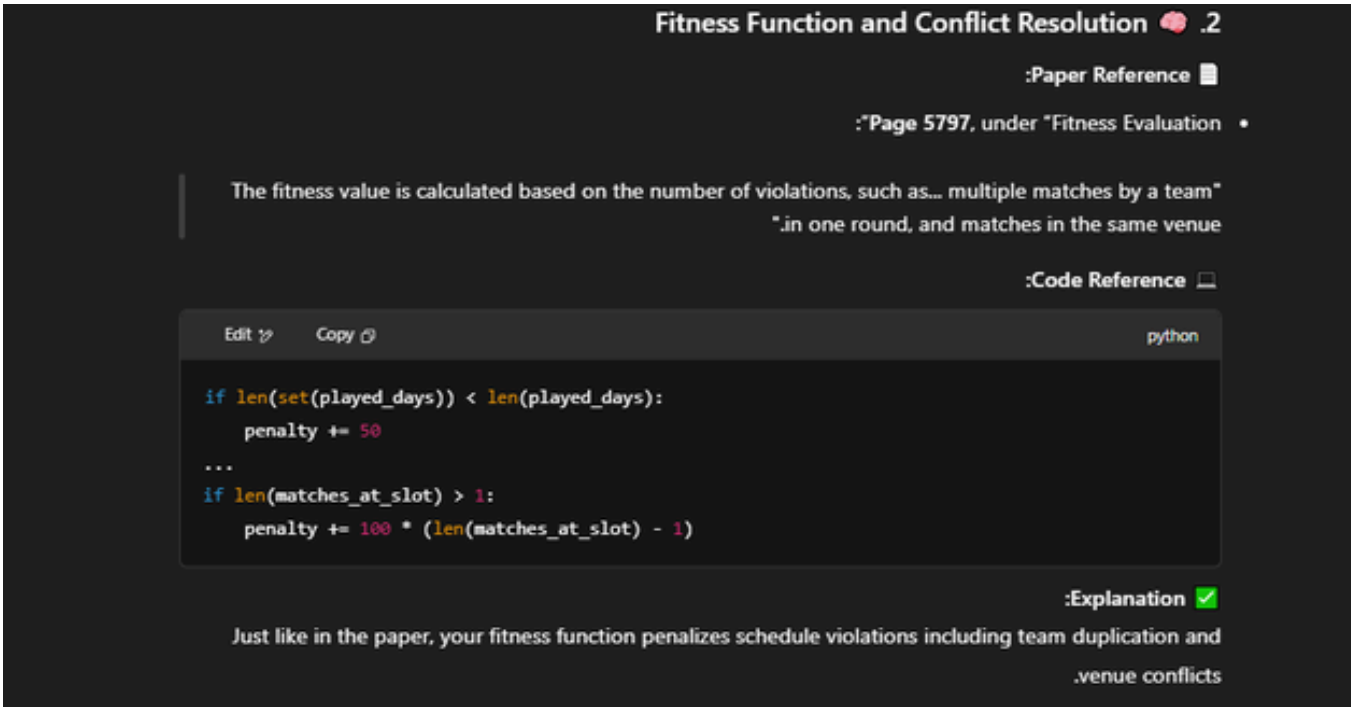
📄 Documentation with Reference to the Paper

Title: ***A Sport Tournament Scheduling by Genetic Algorithm with Swapping Method***
(Pages 5796–5800)





Swapping Method (Mutation) 🔄 .3

:Paper Reference 📄

- **:"Page 5798**, 1st column, under "Swapping Method

> The swap operation is performed by exchanging two matches in a schedule... this helps to reduce" ".conflicts

:Code Reference 💻

Edit Copy python

```python
def swap_mutation(schedule, mutation_rate=0.1):
    ...
    schedule[idx1], schedule[idx2] = schedule[idx2], schedule[idx1]
```

:Explanation ✅

You've implemented the Swapping Method as part of the mutation phase, which the paper suggests as a .means of improving solution quality

Crossover Mechanism 🔗 .4

:Paper Reference 📄

- **:Page 5797**, last paragraph

> In crossover operation, two parent schedules are recombined to produce new children... segments are" ".swapped while maintaining feasibility

:Code Reference 💻

Edit Copy python

```python
def order_crossover(parent1, parent2, crossover_rate=0.8):
    ...
    offspring1 = [match for match in parent2 if match not in segment]
```

:Explanation ✅

You apply **Order Crossover**, a common method that aligns with the general crossover logic discussed in the .paper (though the paper doesn't specify OX by name)

Selection and Elitism 🏆 .5

:Paper Reference 📄

- **:Page 5797**, last paragraph

> ".The best solutions are carried over to the next generation to retain the quality of solutions"

:Code Reference 💻

Edit Copy python

```python
def tournament_selection(...)
def survivor_selection(...)
```

:Explanation ✅

Your use of tournament selection and survivor elitism reflects the paper's emphasis on preserving the best .solutions across generations