1. Scenario

The project is a simple **Video Game Tournament Management System** that allows users (Players and Organizers) to register, manage tournaments, and record game results. It simulates the operations of managing eSports events, tracking participants, and determining winners.

---

2. Design Paradigm / Functionalities to Demonstrate

- Class hierarchy with fields and various method signatures
- Interface implementation
- Method overloading and overriding (polymorphism)
- Use of data structures (ArrayList, HashMap, HashSet, Queue)
- File handling (reading/writing match results and player data)
- Java Stream API with Lambda expressions
- Comparable and Comparator for sorting players and games
- Unit testing using JUnit
- Git-based version control and collaboration
- Test-Driven Development (TDD)

---

3. Expected Output

- Users can register as Players or Organizers
- Organizers can create and manage tournaments
- Players can register for tournaments
- Record and view match results
- Rank players based on performance
- Save/load tournament data to/from a file
- Console-based navigation for the application

---

4. Class Hierarchies

Hierarchy 1 - Users:

- User (abstract class)
    - Player
    - Organizer

Hierarchy 2 - Events:

- Tournament (abstract class)
    - SoloTournament
    - TeamTournament

---

5. Interface

- Playable interface with method playMatch(Player p1, Player p2)
- Why needed: Ensures any tournament type can simulate a match between players

---

6. Runtime-Polymorphism

- registerTournament() and playMatch() overridden in SoloTournament and TeamTournament
- Also applied in overridden displayStats() in User subclasses

---

7. TextIO Usage

- Class: TournamentFileManager
- Purpose: Read/write player stats and tournament data to .txt files

---

8. Comparable and Comparator

- Player implements Comparable<Player> to sort by score
- PlayerUsernameComparator to sort alphabetically

---

9. Class Diagram

User (abstract)

|- username: String

|- id: String

|- displayStats(): void

|

|_ Player implements Comparable<Player>

|_ Organizer

Tournament (abstract)

|- name: String

|- participants: List<Player>

|- playMatch(Player, Player): void

|

|_ SoloTournament implements Playable

|_ TeamTournament implements Playable

interface Playable

|- playMatch(Player p1, Player p2): void

class PlayerUsernameComparator implements Comparator<Player>

class TournamentManager

|- users: List<User>

|- tournaments: List<Tournament>

class TournamentFileManager

|- saveTournaments()

|- loadTournaments()

|- savePlayers()

|- loadPlayers()

---

10. Deliverable 2 - Implementation Scope (50%)

- Create all classes and interfaces with method headers
- Complete UML in class diagram tool (included in PDF)
- JavaDoc documentation
- TODO blocks in all method bodies
- Write Unit test stubs using JUnit for main logic