LO1: Analyse requirements to determine appropriate testing strategies

All listed requirements have been categorized according to their respective levels: system, integration, or unit. This categorization follows a structured approach, where system-level requirements define the overarching functionality that the application must provide. These requirements often include specific criteria that serve as the foundation for meeting unit or integration-level needs. The distinction between unit and integration levels is clearly defined: unit-level requirements focus on individual, specific functionalities, while integration-level requirements encompass multiple unit-level criteria working together. Additionally, all non-functional requirements are classified exclusively at the system level, as they relate to the overall performance and characteristics of the complete system.

Functional requirements

ID	Requirement	Description	Priority	Level of Requirement
F1	Player Turn Management	The system shall allow two players to take turns placing a disc.	High	System
F2	Win Condition Determination	The system shall determine the winner when four discs are connected horizontally, vertically, or diagonally.	High	System
F3	Full Column Restriction	The system shall not allow a player to place a disc in a full column.	High	Unit
F4	Game Restart Option	The system shall provide an option to restart the game when one player win.	Medium	Integration
F5	Real-Time Board Display	The system shall display the current state of the game board in real time.	High	System
F6	Input Validation	The system shall prevent invalid user inputs, such as selecting non-existent columns.	High	Unit
F7	Board Initialization Test	The system shall initialize the board based on user input (rows, columns, rules) and display it correctly.	High	Integration
F8	Player Give up	The system shall allow a player to give up the game, ending it immediately and	Medium	Integration

		declaring the opponent as the winner.		
F9	NPC Opponent	The system shall provide an NPC opponent for single-player mode.	High	System
F10	Player Move Display	The system shall display the correct placement of player moves on the board.	High	Integration

Non-Functional Requirements:

ID	Requirement	Description	Priority	Level of Requirement
N1	Response Time	The system shall update the game board within 100 milliseconds after a move.	High	System
N2	Reliability	The system shall operate reliably with no crashes during 100 consecutive games.	High	System
N3	Memory Usage	The system shall use less than 50MB of memory during runtime.	Medium	System
N4	Platform Compatibility	The system shall be compatible with major operating systems, including Windows, macOS, and Linux.	Medium	System
N5	Safety	The system shall ensure input validation to prevent illegal actions that could cause crashes or incorrect behaviors.	High	System
N6	Privacy	The system shall not store or transmit any personal user data, ensuring complete anonymity for players.	High	System

Qualitative Requirements

Q1	User-Friendly Interface	The system shall provide a user-friendly interface, accessible to players of all ages.	Medium	System

Test approach for chosen attributes:

F1: Player Turn Management

- **Test Approach:** Perform Unit Testing to verify that the system alternates turns correctly between two players. Use parameterized testing to simulate multiple moves and ensure turn alternation behaves as expected.

F2: Win Condition Determination

- **Test Approach:** Perform Unit Testing and Integration Testing to test all possible win scenarios (horizontal, vertical, diagonal). Use predefined board states and edge cases such as a win occurring on the last move. Validate the winner detection logic and its integration with the display module.

F3: Full Column Restriction

- **Test Approach:** Perform Unit Testing to simulate moves that fill a column completely and attempt to add another disc. Ensure the validation logic rejects invalid moves. Use Boundary Value Testing to validate behavior at column limits.

F4: Game Restart Option

- **Test Approach:** Perform Integration Testing to simulate a game restart triggered by a win or mid-game. Verify that all modules (board state, player turn, and game rules) reset to their initial states. Use Regression Testing to ensure no residual data persists after a restart.

F5: Real-Time Board Display

- **Test Approach:** Perform System Testing to simulate multiple player moves and verify the board updates correctly in real-time. Use Visual Testing to ensure the updated board matches the internal game state. Validate performance using a timer to ensure updates occur within 100ms.

F6: Input Validation

- **Test Approach:** Perform Unit Testing to test invalid inputs such as negative numbers, out-of-bounds values, and non-integer inputs. Use Negative Testing to ensure the system provides proper feedback and prevents crashes.

F7: Board Initialization Test

- **Test Approach:** Perform Integration Testing to initialize the board with varying dimensions and rules (e.g., connect 3, connect 4). Validate the logic using Equivalence Partitioning and verify the visual display against the expected dimensions.

F8: Player Give Up

- **Test Approach:** Perform Integration Testing to simulate a player forfeiting mid-game. Validate that the game ends immediately, the opponent is declared the winner, and the display updates correctly. Use Behavioral Testing to ensure proper responses to this scenario.

F9: NPC Opponent

- **Test Approach:** Perform System Testing to simulate single-player mode with the NPC opponent. Test the NPC logic at different difficulty levels using AI Behavior Validation. Validate moves against the game rules and ensure the NPC's decisions are logical.

F10: Player Move Display

- **Test Approach:** Perform Integration Testing to simulate multiple player moves and verify the correct placement on the board. Use End-to-End Testing to validate the interaction between the logic module and the display module. Test with edge cases, such as moves near the board's boundaries.

N1: Response Time

- **Test Approach**: Perform **Performance Testing** to measure the time taken for the board to update after a move. Use tools like JMeter or custom timing scripts to simulate player moves and validate that updates occur within 100 milliseconds. Include tests under normal and high-load conditions.

N2: Reliability

- **Test Approach**: Conduct **Stability Testing** by simulating 100 consecutive games without restarting the application. Include random and edge-case moves to ensure the system remains stable. Use automated scripts to run games continuously and monitor for crashes or unexpected behaviors.

N3: Memory Usage

- **Test Approach**: Perform **Memory Profiling** during gameplay using tools such as VisualVM or YourKit. Test memory usage under various scenarios, including edge cases (e.g., full board, repeated game restarts). Verify that memory consumption does not exceed 50MB during runtime.

N4: Platform Compatibility

- **Test Approach**: Conduct **Compatibility Testing** by running the game on multiple operating systems (Windows, macOS, Linux). Test different versions of the operating systems to ensure consistent functionality and performance. Use virtualization or cloud-based platforms for wider coverage.

N5: Safety

- **Test Approach**: Perform **Security Testing** and **Negative Testing** to validate input handling. Test scenarios include invalid inputs (e.g., negative numbers, out-of-bound values) and injection attacks or malformed inputs. Verify that the system prevents crashes and handles invalid inputs gracefully.

N6: Privacy

- **Test Approach**: Conduct **Privacy Testing** to ensure no user data is stored or transmitted. Use network monitoring tools like Wireshark to verify that no sensitive information is sent over the network. Validate that game sessions are anonymous and no logs contain personal data.

Q1: User-Friendly Interface

- **Test Approach**: Conduct **Usability Testing** with a diverse group of users across different age groups. Collect feedback on the interface design, clarity of instructions, and ease of navigation. Use surveys or observational studies to identify potential improvements. Include **Accessibility Testing** for colorblind or visually impaired users to ensure inclusivity.