Testing Documentation for Advanced Tic-Tac-Toe

1. Testing Strategy

- Unit Testing: Used the QtTest framework to test individual components (TestGameLogic, TestAI, TestUserAuth).
- Test-Driven Development: Tests were written to define expected behavior, and code was adjusted to pass tests.
- Isolation: Tests are independent, with initTestCase() and cleanupTestCase() ensuring a clean state.
- Coverage Goals: Cover critical functionalities (game mechanics, AI decisions, user authentication) and edge cases.
- Debugging: Added debug output (e.g., qDebug() in TestUserAuth) to trace behavior and used Qt Creator's debugger to fix crashes.

2. Detailed Test Cases

2.1 TestGameLogic

- initTestCase(): Initializes the game logic (gameLogic.resetGame()).
- testResetGame(): Verifies resetGame() clears the board and sets currentPlayer to PLAYER_X.
- testMakeMove(): Tests valid and invalid moves, ensuring players alternate turns.
- testCheckGameStatusHorizontal(): Tests horizontal win (positions 0, 1, 2 with CELL_X).
- testCheckGameStatusVertical(): Tests vertical win (positions 1, 4, 7 with CELL_O).
- testCheckGameStatusDiagonal(): Tests diagonal win (positions 0, 4, 8 with CELL_X).
- testCheckGameStatusDraw(): Tests draw condition with a full board and no winner.
- cleanupTestCase(): Performs cleanup (no specific actions needed).

2.2 TestAI

- initTestCase(): Initializes the game logic.
- testGetBestMove(): Verifies the AI makes a valid move on an empty board.
- testMinimaxWin(): Tests AI winning move (board: O X X | O O X | _ X O, AI places CELL_O at position 6).
- testMinimaxBlock(): Tests AI blocking opponent's win (board: X O _ | X _ _ | _ _ _, AI places CELL_O at position 6).
- testMinimaxDraw(): Tests AI forcing a draw (board: O O X | _ X O | O X X, AI places CELL_O at position 3).
- cleanupTestCase(): Performs cleanup (no specific actions needed).

2.3 TestUserAuth

- initTestCase(): Sets up the database (users.db), creates users and game_history tables.
- testRegisterUser(): Tests successful registration and duplicate user failure.
- testLogin(): Tests successful and failed login attempts.
- testSaveGameResult(): Tests saving game results and retrieving history (2 entries: "X won vs AI", "Draw vs Human").
- testGetGameHistory(): Tests retrieving history for existing and non-existent users.
- cleanupTestCase(): Closes database connections and deletes users.db.

3. Test Results

Test Suite	Tests Passed	Time Taken
TestGameLogic	8/8	1 ms
TestAI	6/6	15 ms
TestUserAuth	6/6	65 ms

Overall: 20/20 tests passed, no failures or crashes.

4. Test Coverage Report

4.1 Coverage Analysis

- GameLogic: ~90% (most methods tested, missing some edge cases like invalid inputs).
- AI Logic: ~70% (core functionality tested, but complex scenarios not fully validated).
- UserAuth: ~85% (typical use cases covered).
- Overall: ~85%.