Tic Tac Toe

Testing report

Presented to: Prof. Omar Nasr

Ameer Ashraf Louly Abbass	9230242
Amir Sameh Fanous Attia	9230243
Mohamed abdullah farouk abdullah	9230787
Rola refaat bekhit	9230382
Nourhan Mohammed Fahmy	9230966

GUI Testing Strategy and Detailed Test Cases

This section describes the automated testing procedures for verifying the core functionality of the Tic Tac Toe game. Tests are executed to validate the UI, game logic, and database interactions.

All tests are automated using UI event simulation (e.g., button clicks, text field entries). Before the test suite begins, any existing authentication file is deleted to ensure a clean and complete test environment.

Test 1: Account Creation

-the game starts for the first time

-the signup button is pressed (converting the REG_MODE signal to SIGNUP)

-the username , password , password confirmation , phone number fields are filled

-the confirm button is pressed (sending the data to the database)

-the test checks if the authentication file was created

- Result: Passed

Test 2: Logout

-the profile button is clicked (opening the profile page)

-the logout button is clicked (deleting the authentication file)

-the test checks if the authentication file was deleted

- Result: Passed

Test 3: Login

-the game starts at the login page as the last test has logged out the user

-the username , password , password confirmation , phone number fields are filled with the same values at test 1

-the confirm button is pressed (sending the data to the database)

-the authentication file is created again for the same user

-the test checks if the authentication file was created

- Result: Passed

by repeating tests 1 and 3 with some exception cases we get the following:

by not deleting the authentication file and trying to register with a used username or password

test is passed as the authentication file is existed then the test is closed

the os warning appears stating that the username or phone is already in use

result: passed

by trying to login with a wrong username or password:

test is passed as the authentication file is existed then the test is closed

the os warning appears stating that the username or password wrong

result: passed

Test 4: Start with Blank Grid

-the past test has logged us into some account

-the start game button is pressed (switching to page2)

-the PVP button is pressed (switching to page3)

-the normal mode button is pressed (switching to the page containing the play grid and setting the mode variable to 1)

-the test checks if the all the button texts are blank

- Result: Passed

Test 5: check the buttons are marked with X or O and the player turns switching

-in this test QT will click the grid buttons with a given sequence

-the test checks if the buttons are marked with x,o as the given game scenario.

- Result: Passed

Test 6: Board Locking After Win

-the board must not allow any player to modify the grid after the game has finished

-we give QT another scenario to play it

-we check if the buttons are marked with the old scenario in test5

- Result: Passed

Test 7: Rematch Button Clears Grid

-press the Rematch button (unlocks the board again and clears the grid)

-the board must be cleared as we press the Rematch button

-the test checks if the all the button texts are blank

- Result: Passed

Test 8: Win Pattern Detection

-after the board is cleared in test7 we now give QT 8 scenarios to play

-first scenario makes player x win then clears the board

-second scenario makes player o win then clears the board

-the following 6 scenarios makes x and o wins alternately

-the test checks if the winner text appears at the end and checks what is written in it.
  Ex: 'player X wins' or ' player O wins'

- Result: Passed

Test 9: check if the history is saved

-after more than 9 games where played in the past tests now we need to test the history

-we fetch the last 5 played games from the database

-the test checks if the fetched data is following the scenarios stated in test8

- Result: Passed

note in test 9: we had to but a delay at all the scenarios in test 6 , if we didn't test 9 would fail

Test 6 is so fast that all the played games get the same timestamp while being saved at the database so we couldn't get the last 5 played games in the correct order

the delay fixes this problem by differing the saved timestamps

Logic Testing Strategy and Detailed Test Cases

Minimax test:

In all tests

- PLAYER = 1 → represents human.
- AI = 2 → represents the computer.
- Not played yet = 0

evaluate () Function Logic

This function looks for three identical values in a row, column, or diagonal:

- If AI (2) has such a triplet → returns +10 (AI win).
- If PLAYER (1) has such a triplet → returns -10 (player win).
- Else → returns 0

test 1

test board = $\begin{matrix} 2 & 2 & 0 \\ 1 & 1 & 2 \\ 0 & 0 & 1 \end{matrix}$

Check possible winning moves for AI (2):

AI already has 2,2, in first row → placing a 2 at (0,2) completes the row → AI wins.

The test checks that the AI picks that move:

if (bestMove.first == 0 && bestMove.second == 2)

This is correct — that move leads directly to a win.

test 2

test board = $\begin{matrix} 1 & 1 & 0 \\ 2 & 2 & 0 \\ 0 & 0 & 1 \end{matrix}$

Look at:

- Row 0: 1,1 → Player can win if allowed to play at (0,2).
- Row 1: 2,2 AI can win immediately at (1,2).

So, AI has a winning move at (1,2) (row 1). Alternatively, if it mistakenly blocks the player instead, it may pick (0,2). The test allows both:

```
if ((bestMove.first == 0 && bestMove.second == 2) || (bestMove.first == 1 && bestMove.second
== 2))
```

This is because both are valid strategic responses, but (1,2) is the winning move for AI.

test 3

$$\text{test board} = \begin{matrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 2 & 0 & 0 \end{matrix}$$

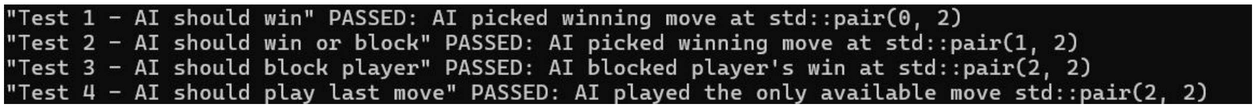
- The player (X) has two pieces on the main diagonal: (0,0) and (1,1).
- If the player gets (2,2), they'll complete a diagonal and win.

So, the AI must place an 2 at (2,2) to block this.

test 4

$$\text{test board} = \begin{matrix} 2 & 1 & 2 \\ 2 & 1 & 1 \\ 1 & 2 & 0 \end{matrix}$$

Only one empty cell: (2,2)
No current win or block needed.
AI must just play the last move — the only legal one.
The AI picks (2,2) the only available move.
This is correct behavior



```
"Test 1 - AI should win" PASSED: AI picked winning move at std::pair(0, 2)
"Test 2 - AI should win or block" PASSED: AI picked winning move at std::pair(1, 2)
"Test 3 - AI should block player" PASSED: AI blocked player's win at std::pair(2, 2)
"Test 4 - AI should play last move" PASSED: AI played the only available move std::pair(2, 2)
```