**MODULE-1**

C-PROGRAM ON MULTIPLE PLATFORMS – PROJECT BASED LEARNING APPROACH

PROJECT:

TICTACTOE

Submitted by:

ASHWIN A CHANDAPUR

**Table of Contents**

[**1.0 DESCRIPTION** 2](#_Toc97248065)

[**2.0 REQUIREMENTS** 3](#_Toc97248066)

[2.1 High Level Requirements 3](#_Toc97248067)

[2.2 Low Level Requirements 4](#_Toc97248068)

[2.3 SWOT Analysis: 5](#_Toc97248069)

[2.4 4W's & 1H 5](#_Toc97248070)

[**3.0 ARCHITECTURE** 6](#_Toc97248071)

[3.1 Structural Diagrams: 6](#_Toc97248072)

[3.2 Behavioural Diagrams: 7](#_Toc97248073)

[**4.0 TEST PLAN AND OUTPUT** 9](#_Toc97248074)

[4.1 Test Plan 9](#_Toc97248075)

[4.2 Table 9](#_Toc97248076)

[4.2.1 Table : High level test plan 9](#_Toc97248077)

[4.2.2 Table : Low level test plan 9](#_Toc97248078)

[4.2.3 Output: 10](#_Toc97248079)

[**5.0 CONCLUSION** 11](#_Toc97248080)

[5.1 LEARNINGS: 11](#_Toc97248081)

# **1.0 DESCRIPTION**

Tic-tac-toe is fun, tactic game played on a three-by-three grid by two players, who alternately place the marks X and O in one of the nine spaces in the grid. The objective of Tic Tac Toe is to get three in a row. You play on a three by three game board. The first player is known as X and the second is O. Players alternate placing X’s and O’s on the game board until either opponent has three in a row or all nine squares are filled. X always goes first, and in the event that no one has three in a row, the stalemate is called a cat game. We can choose any symbol, as there is not restriction in that & any player can go first.

# **2.0 REQUIREMENTS**

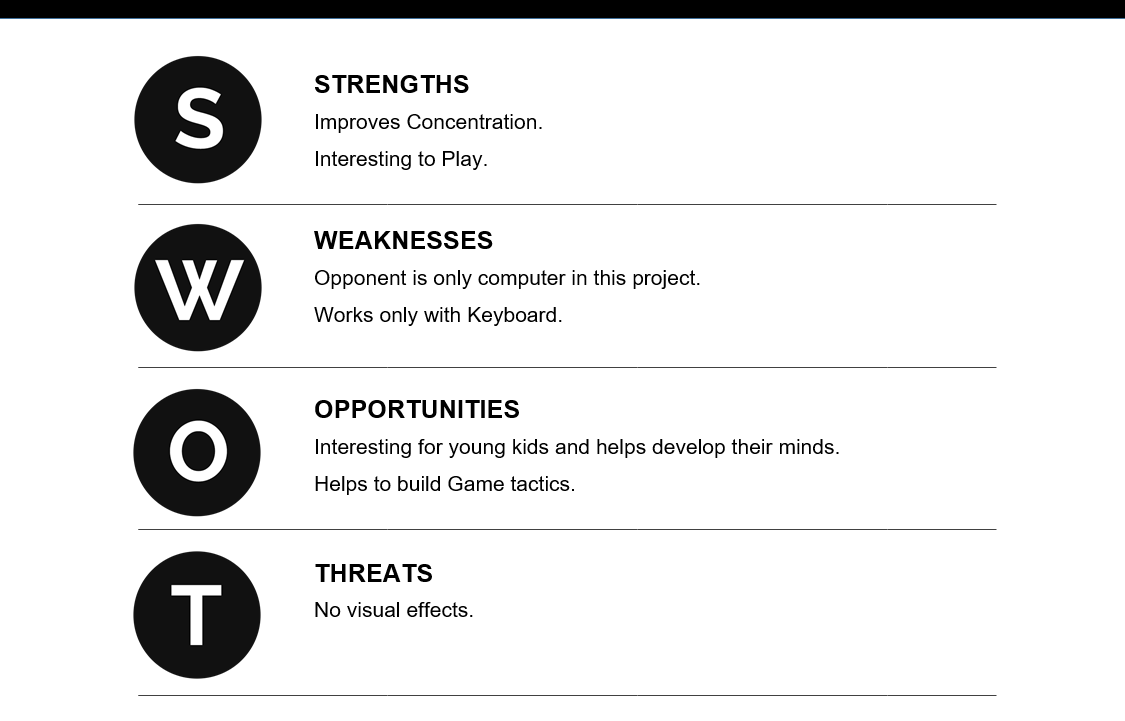
## 2.1 High Level Requirements

| **ID** | **Description** |
| --- | --- |
| HR01 | Player1 shall be able to choose 'X' to play. |
| HR02 | Player2 shall be able to choose 'O' to play. |
| HR03 | Player shall be able to Exit the game. |
| HR04 | Player1 shall win/lose. |
| HR05 | Player2 shall win/lose. |
| HR06 | Game shall end up in a Draw. |

## 2.2 Low Level Requirements

| **ID** | **Description** | **HLR ID** |
| --- | --- | --- |
| LR01 | If the user presses '1', he'll be play with 'X'. | HR01 |
| LR02 | If the user is playing with 'X', he'll get the first turn. | HR01 |
| LR03 | If the user presses '2', he'll be play with 'O'. | HR02 |
| LR04 | If the user is playing with 'O', he'll get the second turn. | HR02 |
| LR05 | If the user presses '3', it'll exit the game. | HR03 |
| LR06 | If the player1 gets 3 X’s or 3 O’s in vertical, horizontal or diagonal row, User will lose. | HR04 |
| LR07 | If the player2 gets 3 X’s or 3 O’s(as per his choice), in vertical, horizontal or diagonal row, he'll win. | HR05 |
| LR08 | If the total number of moves, i.e., 9 moves have been completed and player1 nor player2 has won, it'll end up in a draw. | HR06 |

## 2.3 SWOT Analysis:

[](https://user-images.githubusercontent.com/98833151/153712194-4c443cc9-2a43-4ec2-8b0d-8e2815cbc976.png)

## 2.4 4W's & 1H

Who

* Anyone who wants to play the game.

What

* This game is for fun purpose.

When

* Anytime when user wants to play.

Where

* Can be played anywhere.

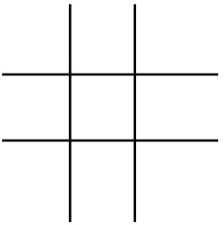
How

* By implementing the code using C-program.

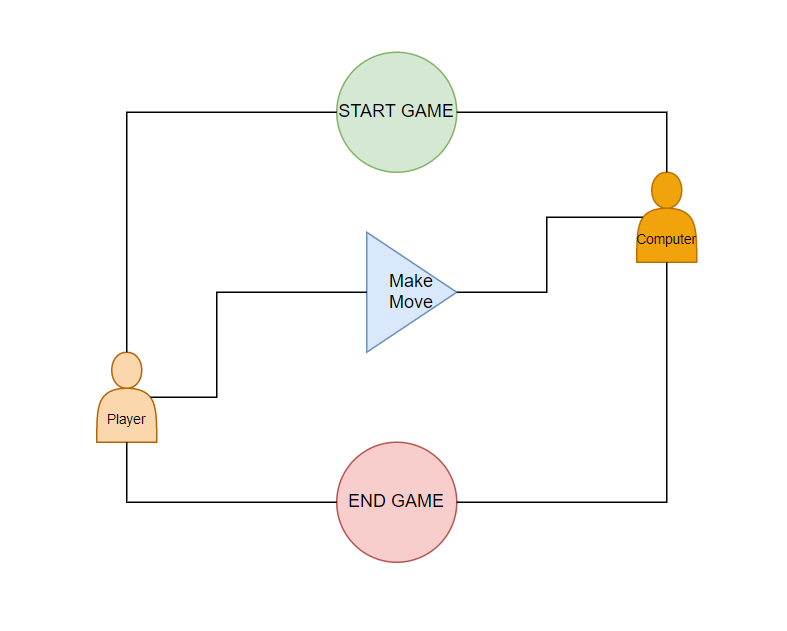
# **3.0 ARCHITECTURE**

## 3.1 Structural Diagrams:

* Basic Layout:

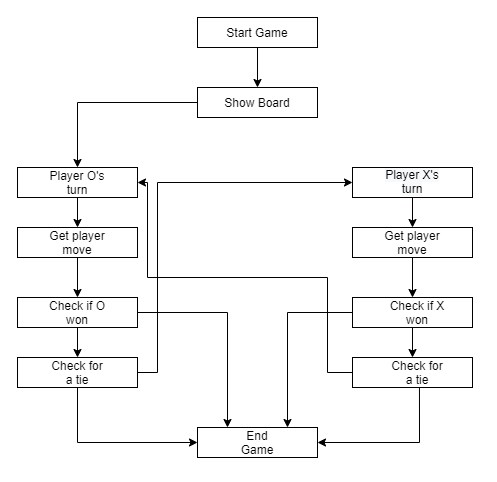
[](https://user-images.githubusercontent.com/98833151/153511942-f6dca797-1c4d-417a-be08-04928610a29f.png)

* Structural Diagram:

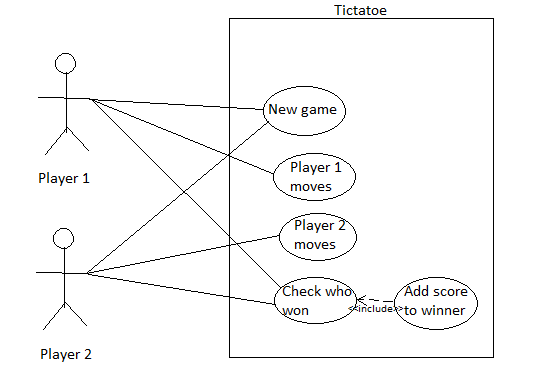
[](https://user-images.githubusercontent.com/98833151/153710438-ca2b19f6-0b9f-4f38-897e-ac63e1acb4c2.png)

## 3.2 Behavioural Diagrams:

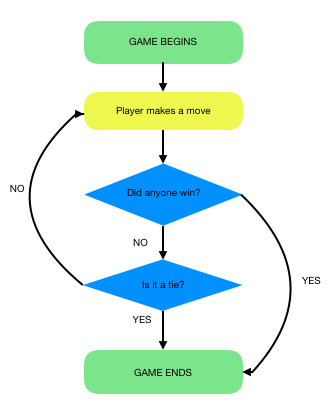
* Block Diagram:

[](https://user-images.githubusercontent.com/98833151/153589617-2d13ab36-2ec6-4f23-b8bf-09b8ba07ad84.png)

* USECASE Diagram:



* Flowchart:

[](https://user-images.githubusercontent.com/98833151/153509725-707fe79f-b9c6-4c07-8cb7-8015a130c763.png)

# **4.0 TEST PLAN AND OUTPUT**

## 4.1 Test Plan

* For every feature, define a test case
  + How to run that feature
  + Define expected behaviour
  + Capture the actual result

## 4.2 Table

* ID, Description of Test case, Input values, Expected Output, Actual Output

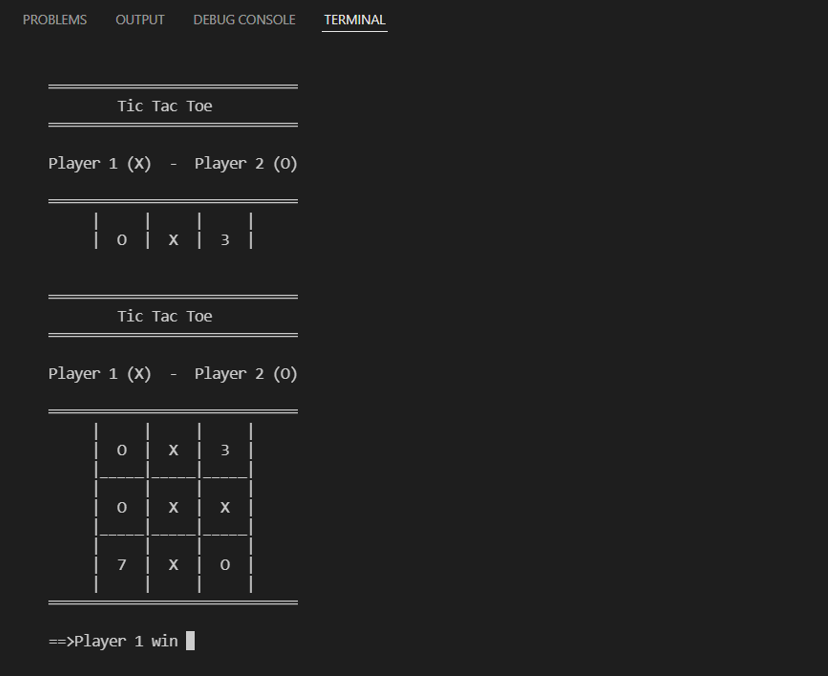
### 4.2.1 Table : High level test plan

| **Test ID** | **Description** | **Input values** | **Expected Output** |
| --- | --- | --- | --- |
| H\_01 | Check if the layout is created | Choose the symbol 'X' or 'O'. | 3x3 grid is formed |
| H\_02 | Check if player1/player2 got 3 of his symbols consecutively in vertical, horizontal or diagonal format. | 'X' or 'O' symbols as inputs from the player1/player2. | The player1/player2 won the game. |
| H\_03 | Check for draw. | 9 inputs from player1 and player2 together. | Game Over. |

### 4.2.2 Table : Low level test plan

| **Test ID** | **Description** | **Input values** | **Expected Output** |
| --- | --- | --- | --- |
| L\_01 | Checking if a 3X3 grid is drawn or not. This 3X3 grid is the basic need to play the game. | Player1 has to choose the symbol. | 3X3 graph is drawn. |
| L\_02 | Play proceeds with the player1 or player2 alternately placing their marks in any unoccupied cell. Check if player1/player2 got 3 of his symbols consecutively in vertical, horizontal or diagonal format. | 'X' or 'O' input from the player1/player2. | The player1/player2 won the game. |
| L\_03 | Check if a total of 9 moves have been made player1 + player2, the game ends up in a draw when neither the user nor the player2 is able to get 3 marks in a row. | 9 inputs from (player1 + player2). | The game is over. |

### 4.2.3 Output:





# **5.0 CONCLUSION**

## 5.1 LEARNINGS:

* Learnt about Product development lifecycle and applied it during this project to arrive at a feasible solution.
* Learnt to use GitHub for storing sharing and evolving the Project content and code.
* Learnt about different aspects of programming like Multifile programming and backend processing.
* Learnt about the execution of files in Linux operating system using different commands.
* Learnt about Unite testing, manual testing and about the efficient way to write a programs.
* Learnt about the process followed in the industry for solving of a problem.