Tic-Tac-Toe Game Analysis

1. User Stories

1. Player Input
   - As a player,
   - I want to enter my name before the game starts,
   - So that my identity is displayed during the game.

2. Game Play
   - As a player,
   - I want to place my symbol (X or O) on the board,
   - So that I can participate in the game.

3. Turn Management
   - As a player,
   - I want the game to alternate turns between players,
   - So that both players have an equal chance to win.

4. Win Condition
   - As a player,
   - I want the game to notify me when someone wins,
   - So that I can celebrate my victory or acknowledge the winner.

5. Draw Condition
   - As a player,
   - I want the game to notify me when the game ends in a draw,
   - So that I can recognize that no one has won.

6. Save Game State

- As a player,
- I want to save the current state of the game,
- So that I can continue playing later without losing progress.

7. Load Saved Game

- As a player,
- I want to load the most recent saved game,
- So that I can continue from where I left off.

2. Minimum Requirements

1. Game Functionality

- Allow two players to take turns placing their symbols on a 3x3 grid.
- Check for a win condition after every move.
- Check for a draw condition when the board is full with no winner.

2. User Interface

- Provide a simple graphical interface displaying the game board.
- Include buttons for each cell in the grid.

3. Player Management

- Collect and display the names of both players.
- Ensure each player has a distinct symbol (X for Player 1, O for Player 2).

4. Persistence

- Allow players to save their game state to a database.

- Enable players to load the latest saved game from the database.

5. Notifications

   - Display messages for the current player, game results, and prompts for replaying or exiting the game.

3. Class Identification

Based on the analysis above, the following classes are identified for the Tic-Tac-Toe application:

1. Board

   - Responsibilities:

     - Manage the state of the game board.

     - Handle player moves and check for win/draw conditions.

2. Player

   - Responsibilities:

     - Store player information, including name and symbol.

     - Provide methods to retrieve player details.

3. Game

   - Responsibilities:

     - Coordinate overall game logic, including player turns.

     - Store references to players and the board.

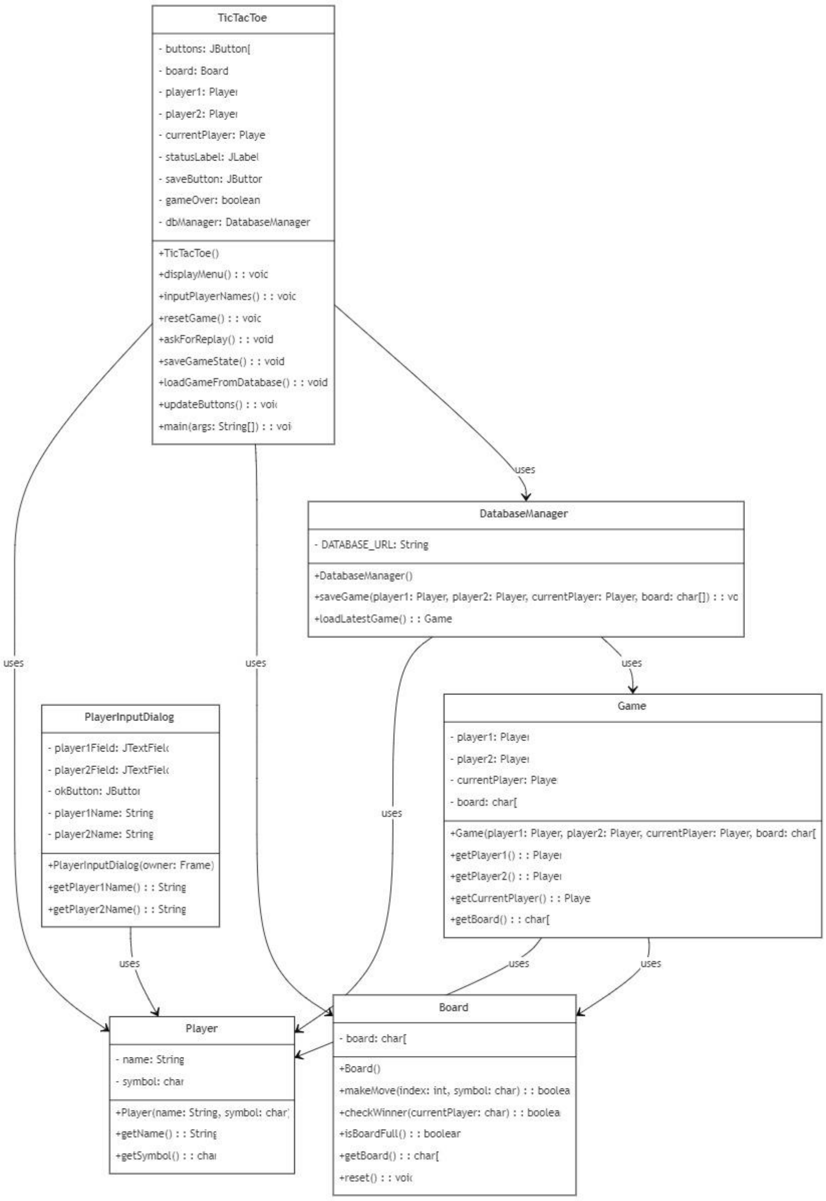
4. DatabaseManager

- Responsibilities:
  - Manage database connections and operations.
  - Save and load game states to/from a SQLite database.

5. PlayerInputDialog
   - Responsibilities:
     - Provide a dialog for players to enter their names.
     - Capture user input and return it to the main application.

6. TicTacToe (Main Application Class)
   - Responsibilities:
     - Initialize and manage the graphical user interface.
     - Handle user interactions and the flow of the game.

TicTacToe UML Clas Diagram


TicTacToe
- buttons: JButton[
- board: Board
- player1: Player
- player2: Player
- currentPlayer: Playe
- statusLabel: JLabel
- saveButton: JButtor
- gameOver: boolean
- dbManager: DatabaseManager
+TicTacToe()
+displayMenu() : : voic
+inputPlayerNames() : : voic
+resetGame() : : voic
+askForReplay() : : void
+saveGameState() : : void
+loadGameFromDatabase() : : void
+updateButtons() : : voi
+main(args: String[]) : : voi
uses
DatabaseManager
- DATABASE_URL: String
+DatabaseManager()
+saveGame(player1: Player, player2: Player, currentPlayer: Player, board: char[]) : : vo
+loadLatestGame() : : Game
uses
uses
uses
uses
Game
- player1: Player
- player2: Player
- currentPlayer: Playe
- board: char[
+Game(player1: Player, player2: Player, currentPlayer: Player, board: char[
+getPlayer1() : : Player
+getPlayer2() : : Player
+getCurrentPlayer() : : Playe
+getBoard() : : char[
PlayerInputDialog
- player1Field: JTextFiel
- player2Field: JTextFiel
- okButton: JButtor
- player1Name: String
- player2Name: String
+PlayerInputDialog(owner: Frame)
+getPlayer1Name() : : String
+getPlayer2Name() : : String
uses
uses
uses
Player
- name: String
- symbol: char
+Player(name: String, symbol: char)
+getName() : : String
+getSymbol() : : char
Board
- board: char[
+Board()
+makeMove(index: int, symbol: char) : : boolea
+checkWinner(currentPlayer: char) : : boolea
+isBoardFull() : : boolear
+getBoard() : : char[
+reset() : : voic