- Describe next Tic Tac Toe Sprint Requirements as defined below
- Functional / Data Requirements:
    - Players
        - Create their own accounts on the server
            - Username (must be unique)
            - Password / Confirm Password
            - First / Last Name
        - Update their accounts
            - Change their name and/or password
            - Username can be changed, but must be unique at the time it is modified
        - Delete their accounts (Soft/Lazy deletion)
        - View the Game History for every game they have played and the player's current W-L-T record
    - Game / Game History
        - Start / End Time
        - Player that Created / Started the game
        - Players involved in the game
        - Human or Computer for each player involved
        - Game Viewers – a list of all players that are / were viewing the game but cannot make moves
        - A log of each move made in the game and who made it and the date/time the move was made.
        - Player that won the game or if the game ended in a tie
        - Unique id for the game (good idea to autogenerate this and make sure it is unique)
    - Game Server
        - Manages all games
        - UI at a minimum should show a list of all active games, a list of all completed games, a list of all registered players in the system.
        - Should be able to drill down into any game (active or completed) and see all the Game details as mentioned above
        - Should be able to see and modify player information
        - Should be a see a list of all active connections (player connections) and the game id of the game they are currently playing or viewing (if any)
        - Workflow:
            - Players REGISTER with the Game Server to Create their accounts
            - Once a player is registered, they can SIGNIN to the Game Server to see a list of active games to join, create a game, view their game history and record, or change their preferences
            - Player can SIGNOUT from the Game Server to disconnect
            - Game Play Scenarios:

        - Player A CREATES a game. The Game is created on the Game Server.
        - Player B JOINS the game. The Game then starts.
        - Player C can choose to join the game as a VIEWER
        - As a player makes a move on their screen, the move is executed on the Server and if valid, propagated to all observers
        - Server will notify all observers when the game is over
- Technical Requirements
    - Database storage
    - Microservices Architecture
    - Design Patterns
- Approach
    - Design / Architecture:
        - Think in terms of isolated modules and communication via message passing
        - Identify the key components of your system. It may help to start with the UI. Draw it out, but keep server and client separate
        - Identify the types of messages PUBLISHED (SENT) by each module and the types of messages each module should SUBSCRIBE TO (LISTEN FOR). This will help you come up with a collection of message types. Don't be too concerned about what goes IN the messages at first.
        - Due at our next class meeting. I will review each one. Make sure you think this through in detail.
    - Deliverables / Implementation:
        - Implement USER REGISTRATION / SIGNIN / SIGNOUT / USER PREFERNCE functionality. This includes server's ability to display all connected users (2 weeks)
        - Implement Game Logic flow (2 weeks)
        - Daily Standup Calls:
            - Did you get done what you said you would for today?
            - What are you going to do tomorrow?
            - Any obstacles in your way?
- Understanding Microservices Architecture
    - https://microservices.io/
    - Highly maintainable and testable
    - Loosely coupled
    - Independently deployable
    - Organized around business capabilities
    - Owned by a small team