Project -- Tic-Tac-Toe

The aim of the project is to create an application to play Tic-Tac-Toe against our common foe, the Bot.

Epic – Provide the ability to play a game of tic-tac-toe against a bot

User Stories—

1. As a user I want to have a grid of 9 to play the game so that I can enjoy a game of tic-tac-toe
   1. Acceptance Criteria-
      1. Given a user is playing the game, then he is presented with 9 grids
2. As a user I want to be able to start the game in any of the grids, so the user get the first choice
   1. Acceptance Criteria-
      1. Given a user has started playing the game, the game should wait until the user has chosen his first grid
3. As a user I want the bot to make its move based on the move I just made so that I can continue playing the game
   1. Acceptance Criteria-
      1. Given the user has made his move, then the bot takes over and makes a move keeping in mind the rules of the game as described here, https://en.wikipedia.org/wiki/Tic-tac-toe
4. As a user I want to play the game until it ends in a Win, Loss or Draw
   1. Acceptance Criteria-
      1. Given the user has made his move, then he and the bot continue to play until the game is ended in a Win, Loss or Draw for the player or the bot
5. As a user I want to have the ability to restart the game
   1. Acceptance Criteria-
      1. Given the user has finished playing the game, he then has the ability restart the game

Epic – Provide means to test code coverage and establish coding guidelines and best practices

User Stories—

1. As a developer, I want to know if the quality of my code is up to the mark and have the ability to track it.
   1. Acceptance Criteria-
      1. Given a developer has written a piece of code, he should then have the ability, on his local machine, to test its quality and generate a report
      2. Given a developer has written a piece of code and deployed it to the repository, then a continuous code quality system (SonarQube) is able to detect bugs, code smells and vulnerabilities based on rules
      3. Given a development manager wants to enforce a coding practice, then he is able to define a rule in SonarQube and ensure that the team abides by it or is flagged for breach.

Epic – Provide a Continuous Integration/Continuous Deployment pipeline to implement code changes into various other environments

Hierarchy of environments:

1. Local Host
2. Integration Server
3. UAT Server
4. Pre-Production Server
5. Production Server

User Stories—

1. As a developer, I would like to automatically build and deploy my code automatically into different environments
   1. Acceptance Criteria-
      1. Given a developer has deployed his code into the code repository – Master branch for now (Github) - then an automated server (Jenkins) detects the changes
      2. Given Jenkins detects changes, it runs a build and upon success of the build the change set is deployed into the Integration and UAT Server
      3. Given the build fails, Jenkins notifies the set of developers who committed code to the last changeset
      4. Given the success of the UAT, the tester/s, are able to push a button in Jenkins to deploy the code into the pre-Production environment
      5. Given the project team deems the code Production ready, they can push a button in Jenkins to deploy the code to the Production environment