CODING ACTIVITY: TIC-TAC-TOE V2.0

OVERVIEW

The focus of this activity is for students, working in teams of 2 or 3, to take an existing game application, a standard 3x3 tic-tac-toe game, and upgrade it to a new version with the required features. The game board options, to be chosen by the team, will be either a 3x3, 4x4, or 3x3x3 (3D). The stakeholder will specify the new version features through the use of user stories. The agile development process will be employed by the team and will include multiple sprints.

TEAM CONFIGURATION

1. Trello
    a. Create a Trello account for each team member.
    b. Have one team member create a Trello board for the project and share that board with each team member and the instructor. Name the Trello board with the team member's last names using the following convention. ***Tic-tac-toe V2.0 – Smith, Jones, and Black***.
    c. Create the following lists in the Trello board; ***Documentation, Project Backlog, Sprint 1, Sprint2, Sprint 3,*** and ***Competed***.
2. GitHub (Note: GitHub must be used for the final submission, but it is optional to use it with the team while developing the game.)
    a. Create a GitHub account for each team member.
    b. Have one team member create a repository for the project and share the link with each team member and the instructor. Name the repository with the team member's last names using the following convention. ***Tic_tac_toe_v2_Smith_Jones_Black***.
    c. Add the link to the Trello ***Documentation*** list in a card titled ***GitHub Repository***.

ASSUMPTIONS

- The tic-tac-toe game is a two-player game. The team can create a “Play the Computer” mode as a challenge, but this will not be graded with the assignment.
- The game will be played using the console in a Windows environment.
- The game will be played using the standard English language.
- The new version game will either be a 3x3, 4x4 or a 3D 3x3x3 game board.
- Historic stats for players will be stored in a text file using a comma delimitated, XML, or JSON format.
- The game will include a main menu with the following options.
    - Play a New Round
    - View the Rules
    - View the Current Game Stats
    - View Historic Game Stats
    - Save Game Results
    - Quit
- The chosen design pattern will be a modified MVC pattern where the GameView object will have access to the Gameboard object for use when displaying the game board and the round status, but only the Controller object will access the Gameboard object to make changes to it. The team may, if they decide, remove all dependency on the Gameboard object from the GameView object by passing the values of the Gameboard object’s properties, but this may prove more cumbersome than valuable.
- Each team will choose both the information to be stored regarding the historic game stats and the format of the text file.
- All sprint tasks will be assigned and managed within the Trello board.
- **Note**: The team may decide to implement the menu during any sprint.

SPRINT 1 – TEAM ORGANIZATION AND USE CASES

Sprint Goals

- Implement the team development environment.
- Convert all user stories to testable use cases.

Deliverables

- A completed and functional Trello board with task cards only in the ***Project Backlog***, ***Sprint 1***, or ***Completed*** lists
- A single document with all use cases and added to the ***Document*** list in the Trello board
- A flowchart for the new game and added to the **Document** list in the Trello board.

General Sprint Instructions

1. Play and completely explore the *Tic-tac-toe Game Ver. 1.0* application. Note the following.
    a. Game Flow
    b. Confusing User Interfaces/Experiences
    c. Existing Bugs
    d. Potential New Features
2. Choose the new version of the game your team will develop.
    a. Level 1: Add the required features to the existing 3x3 game.
    b. Level 2: Upgrade to a flat 4x4 board.
    c. Level 3: Upgrade to a 3D 3x3x3 board.
3. Update and add any additional user stories necessary to define the game behaviors.
4. Write a complete set of Use Cases for the game using the Simple Template.
5. Add *Sprint 1* tasks as **cards** to the ***Project Backlog***. Then assign the **cards** to team member(s) and move them into the ***Sprint 1*** list. Some Sprint 1 tasks are listed below.
    a. Setup Trello
    b. Setup GitHub
    c. Choose upgrade level.
    d. Add a use case **card** for each user story. More than one use case may be necessary for some user stories to completely define the behavior and test it. Use the *Use Case Template* located in the ***Course Resources*** module of the Moodle shell.
    e. Combine all use cases into one document and add that use case document to the Trello ***Documentation*** list.
6. As team members complete a task **card**, move the **card** to the ***Completed*** list in Trello.

SPRINT 2 – DEVELOPING AND IMPLEMENTING THE NEW MODEL

Sprint Goals

- Develop a new, or modify the existing, model to accommodate the features and game board of the new game.
- Implement the model in both the view and the controller.
- The game flow will be fully implemented and game play will be functional. Some validation may not be complete.

Sprint Deliverables

- A completed and functional Trello board with task cards only in the ***Project Backlog***, ***Sprint 2***, or ***Completed*** lists
- A fully functional game, pushed to the GitHub repository

General Sprint Instructions

1. Review the ***Sprint 1*** task cards. If they will not be implemented this sprint, move them to the ***Project Backlog*** and remove the assigned team member(s). If they will be implemented this sprint, review and modify if necessary the assigned team member(s) and move them to the ***Sprint 2*** list.
2. Add *Sprint 2* tasks as **cards** to the ***Project Backlog***. Then assign the **cards** to team member(s) and move them into the ***Sprint 2*** list. Some general *Sprint 2* tasks are listed below.
    a. Develop the model(s) to handle the new version of the game. Note that most, if not all, methods will need to be rewritten to manage the new game board concept.
    b. Modify the method(s) to display the game board to accommodate the new model.
    c. Modify the method(s) to get a player's choice to accommodate the new model.
    d. Add any new, or modify any existing, features agreed upon by the team and the stakeholder.
    e. Fully test the game against the use cases.
    f. Push the solution to the remote repository on GitHub.
3. As team members complete a task **card**, move the **card** to the ***Completed*** list in Trello.

SPRINT 3 – PERSISTENCE, A MENU, AND THE COMPLETED GAME

Spring Goals

- Implement a menu system in the game including the following options.
  - Play a New Round
  - View the Rules
  - View the Current Game Stats
  - View Historic Game Stats
  - Save Game Results
  - Quit
- Implement validation and feedback messages to all player inputs.
- Implement persistence with the historic game stats; saving the current and displaying past.

Sprint Deliverables

- A completed and functional Trello board with task cards only in the ***Project Backlog*** or ***Completed*** lists
- A fully functional and tested game application
- A complete set of instructions for the game
- A video walkthrough of the game
- Documentation itemizing all team member contributions to the project

General Sprint Instructions

1. Review the ***Sprint 2*** task cards. If they will not be implemented this sprint, move them to the ***Project Backlog*** and remove the assigned team member(s). If they will be implemented this sprint, review and modify if necessary the assigned team member(s) and move them to the ***Sprint 3*** list.
2. Add *Sprint 3* tasks as **cards** to the ***Project Backlog***. Then assign the **cards** to team member(s) and move them into the ***Sprint 3*** list. Some general *Sprint 3* tasks are listed below.
   a. As a team, determine what historic information will be saved and choose a data file structure, i.e. text or XML, depending on the complexity of the data.
   b. Implement the **Save Game Results** and **View Historic Game Stats** option in the **Main Menu**.
   c. Implement all necessary validation checks and feedback messages.
   d. Fully test the game against the implemented use cases.
   e. Upload the solution to the remote repository on GitHub.
   f. Create a document of instructions for the game and add it to the ***Documentation*** list.
   g. Create a document identifying all use cases not fully implemented and add it to the ***Documentation*** list.
   h. Create a document itemizing all team member contributions to the project and add it to the ***Documentation*** list.
3. As team members complete a task **card**, move the **card** to the ***Completed*** list in Trello.

USER STORIES

The players will see an Opening screen with options to Main Menu or Quit when they start the program.

If the players choose Quit at the opening screen, the application will immediately terminate.

At the start of a round, the players will be prompted to either choose which player should go first, alternate from the previous round if there was one, or let the computer make a random choice.

When the round begins, the players will see the Game Board screen including a representation of the current game board and a specified player will be prompted for their next move, a position on the game board.

If a player does not enter a valid number or a number within a valid range for the row or column (or level in 3x3x3), they will be provided helpful feedback and be prompted to reenter the number.

If a player does not enter a valid number or a number within a valid range for the row or column (or level in 3x3x3) 4 times in a row, the game will provide feedback, end the round, and display the Game Status screen.

If a player requests a position on the game board previously taken, they will be provided helpful feedback and prompted to request a new position on the game board.

If a player does not request a free position on the game board 4 times in a row, the game will provide feedback, end the round, and display the Game Status screen.

The players can, at any point in a round, exit the round and then see the Game Status screen.

After a player makes valid position choice, the game will check for a win or cats game and if neither, will display the new game board and prompt the next player for a move.

If a player wins or a cats game occurs, the game will provide feedback and the players will see the Game Status screen.

When the players are on the Game Status screen, they will be prompted to return to the Main Menu.

The Main Menu will display the following options; Play a New Round, View the Rules, View the Current Game Stats, View Historic Game Stats, Save Game Results, and Quit.

If the players chose Main Menu at the opening screen, they will see the Main Menu screen.

When the players choose Quit from the Main Menu screen they will see the closing screen.

When the players choose Play a New Round from the Main Menu screen they will be prompted to choose the starting player using one of the options and then show the players the Game Board screen.

When the players choose View the Current Game Stats from the Main Menu screen they will see the Game Stats screen and be prompted to return to the Main Menu.

When the players choose View Historic Game Stats from the Main Menu screen they will see the Historic Game Stats screen and be prompted to return to the Main Menu.

When the players choose Save Game from the Main Menu screen the required information will be added to the data file and the players will see a screen confirming the save.

September 2, 2016

Nate Tre Soxide
Laughing Leaf Productions
1234 Silly Maple Street
Hilarious Hollow, MI 99999

To the Development Team:

First, let me express our appreciation to your team for helping our company with this project.

We are developing a console-base application to facilitate the playing of tic-tac-toe between two players. Our company hired a development company last year to build out the application. During the development process the company, and the entire development team, disappeared into a sink hole created by the dumping of residue waters into the water table from a secret soup factory. The building and all of the occupants were never seen again.

The application was not completed, but the source code, stored on GitHub, was saved and is available. I am sorry to report though that all documentation was lost in the accident. Given our loss, we have decided to take this opportunity to expand our concept and features for the game. Please find included a set up user stories generated by our Concept and Marketing team. The team has designated John Velis as our liaison with your team. Please address all development question with John. He has been empowered to make any changes he and your team deem necessary to complete the project on time and push the app to the online store.

Again, let me express our deepest appreciation for your assistance and we look forward to working with you on this project.

Sincerely,

Nate

FEATURES AND REQUIREMENTS SPECIFICATION SHEET

	Level I	**Level II (include all Level I requirements)**	**Level III (include all Level II requirements)**
Game Board	Standard 3x3 Tic-tac-toe game board	A 4x4 Tic-tac-toe game board	A 3D, 3 level 3x3 Tic-tac-toe game board
Validation and Feedback Messages	Most potential user input errors are caught and appropriate feedback messages are provided to the user.	All potential user input errors are caught and appropriate feedback messages are provided to the user.	User-defined exception included and handled.
Persistence		Some form of historic game stats are stored and can be displayed via the Main Menu	Historic game stats are stored by user and can be presented using a user filter
Menu Options	• Play a New Round • View the Rules • View the Current Game Stats • Quit	• Play a New Round • View the Rules • View the Current Game Stats • View Historic Game Stats • Save Game Results • Quit	• Play a New Round • View the Rules • View the Current Game Stats • View Historic Game Stats • Save Game Results Quit
Design Pattern	Some identifiable design pattern in discernable.	MVC	MVC with view having read-only access to all models
Marking Value	**50 Points**	**10 Points**	**10 Points**

RUBRIC (MARKING GUIDE)

Conventions	• Solution, project and folder structures adhere to the standards for a given design pattern. • Files, classes and their elements adhere to the course naming conventions.	5
Readability	• File, class, and class element names are descriptive and consistent. • Whitespace is used effectively and consistently. Nested elements are indented and code blocks are separated by blank lines. • Classes, constructors and methods use XML tag commenting including parameter and return elements. • Significant code blocks use single line commenting.	5
Structure	• Class elements are organized and adhere to the course convention. • Methods perform a specific, single action. • Code blocks are not duplicated in more than one location. • Inheritance and polymorphism are used to organize classes. • Separation of concerns, modularity and reusability are taken into consideration.	5
Robustness	• The UI is clear regarding the type of user interaction and input, and provides guidance when an exception is generated by the user. • All potential exceptions are trapped and handled.	5
Efficiency & Elegance	• Appropriate algorithms are used and the code is small, yet still readable. • The code does not include unused or extraneous elements.	5
Features – Level I	All features and requirements are implemented completely.	50
Features – Level II	All features and requirements are implemented completely.	10
Features – Level III	All features and requirements are implemented completely.	10
Overall Quality, Creativity & Effort	• The application feels complete with attention paid to details. • The application demonstrates creativity on the part of the developer. • The application demonstrates significant effort on the part of the developer.	5