**Team Project Sprint #1**

Instructions

Please read the instructions carefully. All members of your team should discuss the instructions together to ensure that everyone is on the same page. Each team should meet at least once a week.

**Objectives**

1. Create a brief project description.
2. Specify all the user stories and acceptance criteria of the target software that allows a ***human*** player to play against a ***human*** opponent.
3. Create an initial GUI design of the software.
4. Implement the primitive functions (i.e., board visualization and piece placement for both players).

**Deliverables and Grading Policy**

1. Project Report (**20 points**)

The project report should include the following sections:

* 1. Project description (micro-charter), which should result from group discussion **(1 point)**.
  2. User stories using the template discussed in class. **(2 points)**

Provide a complete list of user stories and estimated efforts for the target software that allows a ***human*** player to play against a ***human*** opponent.

* 1. Acceptance criteria using the template discussed in class. **(9 points)**

Provide complete acceptance criteria for each of the user stories. Note that, although some of the user stories will be implemented in the future sprints, their acceptance criteria need to be defined in the first sprint. You may continue to improve the user stories and acceptance criteria in the next sprint.

* 1. Initial user interface design. **(2 points)**

Describe all components of your user interface design. Screenshots may be included.

* 1. Implementation tasks **(4 points)**

Describe the production code, automated test code or manual test case for each user story and acceptance criterion related to the implementation of the primitive functions, including **board visualization** and **piece placement**. For each acceptance criterion of every user story for the primitive functions, you need to implement at least one test (either test code or manual test case).

* 1. Minutes of ALL meetings, including, but not limited to: project/sprint planning meeting, stand-up meeting, backlog grooming, retrospective meeting, and pair programming (or development) session. **(2 points)**
  2. A table of buddy ratings. Individual members may email their buddy ratings to the instructor or teaching assistant.

Each team only needs to submit one report. For an individual member to receive the credit for this part of the project, the team’s project report must include explicit evidence of his/her contribution (e.g., his/her name is listed as a developer).

2. Demonstration **(5 points)**

Submit a 5-minute video, clearly demonstrating that:

1. your project has implemented the working software for the primitive functions, i.e., board visualization and piece placement for both players.
2. for each acceptance criterion of every user story for the primitive functions, your project has implemented either an automated test method or performed an acceptance test manually.
3. your project has some unique features or enhancements (optional).

Grading of the demonstration: completion of the required functions (**4 points**) and overall presentation (**1 point**) using the following evaluation rubric:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Poor | Fair | Good | Very Good | Excellent |
| Was the demonstration logically organized |  |  |  |  |  |
| Were points made clearly and concisely |  |  |  |  |  |
| Were the instructor’s questions answered satisfactorily |  |  |  |  |  |

1. Source Code

Submit all source code. Make sure your project report is consistent with the source code.

**Team Project Sprint #1**

Sprint 1 Report

Team Name: Team 11 Online

Team Members: Madhuri Sarode  
 Debsankar Mukhopadhyay  
 Mounika chinna bandhyala  
 Neeharika jasti  
 Ajay reddy Byreddy

1. **Project Micro-Charter (no more than one page)**

Provide a brief description about the project, including the following elements:

Project name: Checkers Game Application

Vision statement: describe the future that you are trying to create: Creating an electronic version of American checker’s game.

Mission statement /project purpose: Creating a Java-based software that creates a two player checker’s game.

Elevator pitch: We will create a downloadable software that accommodates the end-users to play an American Checker’s game on their devices without any need to have a physical board.

Business value: A light-weight software interface for playing checker’s game.

Customers and users: Anyone that has an interest in American Checker’s game and has a computer at minimum.

Metrics: Ease of downloading and starting the software, and ease of playing the game. Getting access to a history of performance of each player.

Milestones:

1. Implementation of User Interface with Game Rules so two players can play the game (November 10)
2. Implementation of Database structure to maintain the account of the players and their performance history – December 10.

Risks: 1. Learning curve for the developers, 2. Complexity in the rules, 3. Automated testing of the Java Swing components, 4. Communication management in a virtual setup; 5. Choosing the technology.

Authors of this micro-charter: Debsankar Mukhopadhyay

GitHub project Hosting Site: <https://umkc-cs-5551.github.io/checkersgamedesktopapp/>

1. **User Stories**

Describe all user stories using the following template: As a <role>, I want <goal> [so that <benefit>]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **User Story Name** | **User Story Description** | **Priority** | **Estimated effort (hours)** | **Actual effort (if completed)** | **Status (completed, toDo, inProgress)** | **Developer names** |
| 1 | Create a welcome screen to start game, exit game and see results of past games | As a player, I need to have the options in the application to start game, exit game and see results of past games | High | 10 | 10 | Completed | Madhuri Sarode |
| 2 | Collect player’s information | As a player, I want to enter my name and other details to have an account | High | 2 | 2 | Completed | Madhuri Sarode |
| 3 | Create an empty Chess Board. | As a player, I want a checker board when I open the application so that I can view the checkers. | High | 16 | 24 | Completed | Debsankar Mukhopadhyay |
| 4 | Display pieces on the board | As a player, I want two sets of pieces on the board so that I can identify myself and the opponent. | High | 40 |  | to Do |  |
| 5 | Player – Checker correspondence | As a player, I want to move only one set of checkers on the board so that I can start play the actual game. | High | 40 |  | to Do |  |
| 6 | Implement the movement rules for diagonal movement | As a player, I want to move my piece only diagonally to an adjacent unoccupied place so that I can make a progress in playing the game. | High | 40 |  | To do |  |
| 7 | Capturing the opponent piece | As a player, I want to capture opponent piece on the diagonally adjacent square by jumping over it so that I can try to reduce the number of opponent to win the game. | High | 40 |  | To do |  |
| 8 | Win or lose the game | As a player, I want my opponent to lose the game if (s)he has no pieces remaining or cannot move and vice versa due to being blocked so that the game has an end result. | High | 40 |  | To do |  |
| 9 | Player Account Management | As a player, I want to save my name to the game database so that I can locate my profile and/or history of games. | High | 40 |  | To do |  |
| 10 | Player History | As a player, if I play the game use a specific name, I want to be able to see my historical performance as a pop up so that I can earn points or any other benefits. | High | 40 |  | To do |  |

1. **Acceptance Criteria (AC)**

Describe all acceptance criteria using the Given-When-Then template. We will groom the backlog during the middle of the sprint and add to the acceptance criteria based on requirements and scope statuses.

|  |  |  |  |
| --- | --- | --- | --- |
| **AC**  **ID** | **Description of Acceptance Criterion** | **Status (completed, toDo, inPprogress)** | **Developer Names** |
| 1.1 Welcome screen | AC 1.1 Welcome screen is displayed  Given A player starts the application and  When player clicks start game  Then the player’s information is collected and game is started | Completed | Madhuri Sarode |
| 1.2 Game results | AC 1.2 Game top score  Given A player starts the application and  When player clicks on results button  Then the top score from previous game is displayed | Completed | Madhuri Sarode |
| 1.3 Game Exit | AC 1.3 Game Application exit  Given A player starts the application and  When player clicks on exit button  Then the application should gracefully terminate by closing all windows. | Completed | Madhuri Sarode |
| 3.1  Create an empty Chess Board. | AC 3.1 A checkerboard is displayed  Given A player opens the application  When The application screen is displayed  Then The player will find a 8x8 area on a frame with alternate black and white squares. | Completed | Debsankar Mukhopadhyay |
| 3.2  Create an empty Chess Board. | AC 3.2 A checkerboard is displayed on screen  Given The application screen is opened  When A player clicks on the EXIT (X) located on the screen header  Then The entire screen is closed and the application exits gracefully. | Completed | Debsankar Mukhopadhyay |
| 4.1 Create checkers on the board | AC 4.1 A player starts the application  Given The player starts the application  When The checker board is displayed  Then Two sets of pieces with different colors (red and blue) will show up on the board representing two different players. | To Do |  |
| 4.2 Create checkers on the board | AC 4.2 Pieces are displayed on the board on the board  Given The board is displayed  When The Pieces are displayed on the board  Then There will be 8 pieces on the top two rows and 8 pieces with different colors on the bottom two rows on the board | To Do |  |
| 4.3 Create checkers on the board | AC 4.3 Pieces are displayed on the board on the board  Given The board is loaded on the screen  When The pieces show up on the board  Then The pieces can only show up on the dark squares of the board. | To Do |  |

1. **User Interface Design**

Describe all components of your user interface design. Screenshots may be included.

AC 1.1 Welcome Screen and collect player’s information if ‘Start game’ is selected

Graphical user interface, text, application

Description automatically generated

AC 1.1

Graphical user interface, application

Description automatically generated

AC 1.2 Show top results of past games

Graphical user interface, application

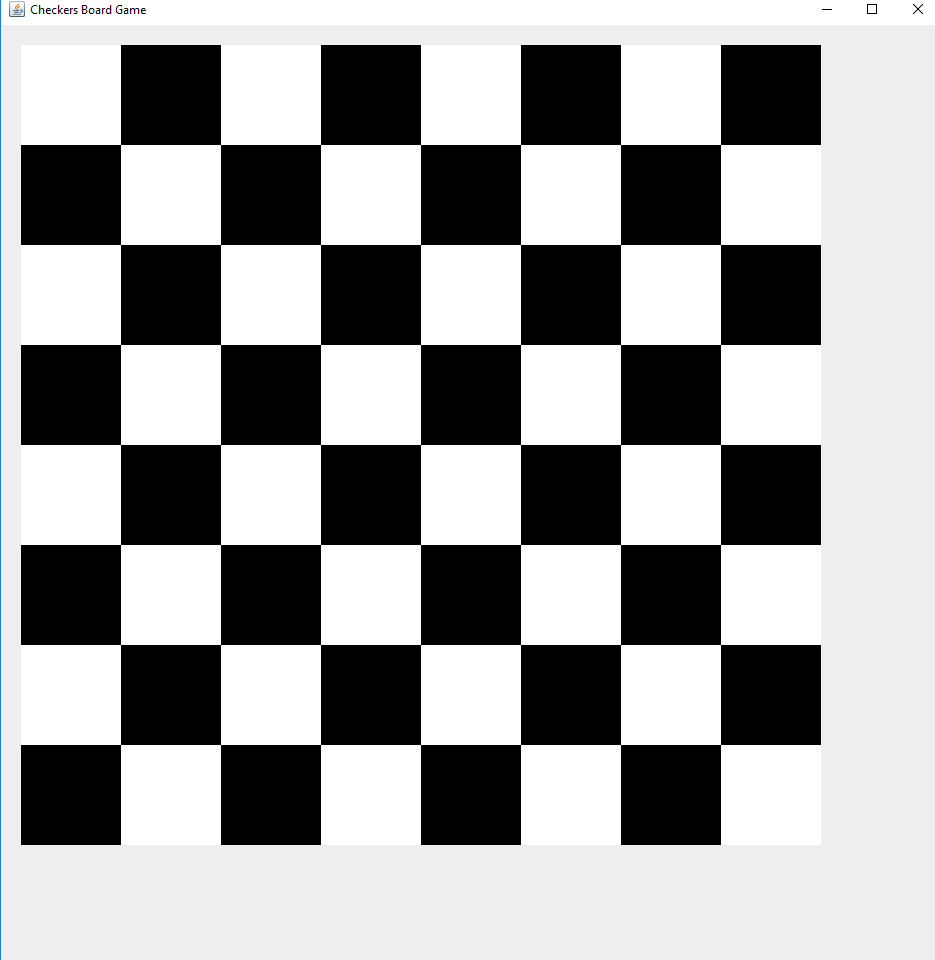
Description automatically generated

AC 1.3 Exit the application gracefully

Graphical user interface, application, Word

Description automatically generated

3.1/3.2 Create an Empty Game Board with exit option



1. **Implementation Tasks**
2. Summary of production code

Summarize how each user story/acceptance criterion is implemented in your production code (class name and method name etc.)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **User Story ID and Name** | **AC ID** | **Class Name(s)** | **Method Name(s)** | **Status (complete or not)** | **Developer Name(s)** | **Notes (optional)** |
| 1 | 1.1 | WelcomeScreen | StartGameMethod  clientInformation | Completed | Madhuri Sarode |  |
|  | 1.2 | WelcomeScreen | StartGameMethod  clientInformation | Completed | Madhuri Sarode |  |
|  | 1.3 |  | StartGameMethod  clientInformation | Completed | Madhuri Sarode |  |
|  | 1.4 | Create an empty Chess Board. | CheckerBoardUI. paintComponent  PlayCheckersGame() | Completed | Debsankar Mukhopadhyay |  |
|  | 1.5 | Create an empty Chess Board. | PlayCheckersGame() | Completed | Debsankar Mukhopadhyay |  |

1. Summary of automated test code directly corresponding to some acceptance criteria

Summarize how user stories/acceptance criterion are tested by your test code (class name and method name).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **User Story ID and Name** | **Acceptance Criterion ID** | **Class Name (s) of the Test Code** | **Method Name(s) of the Test Code** | **Description of the Test Case (input & expected output)** | **Status** | **Developer Name(s)** |
| 1 | 1.1 |  |  |  |  |  |
|  | 1.2 |  |  |  |  |  |
|  | 1.4 |  |  |  |  |  |
| 3 Create an Empty Game Board | 3.1 | AppTest | checkerBoardUITestHappy()  checkerBoardUIPresentTest() | INPUT: Create a JFrame and add a checkbosrdUI object  OUTPUT: The size of the board created is returned correctly.  Input: Instantiate the class PlayCheckersGame  OUTPUT: The created JFrame contains the class CheckerBoardUI | Passed | Debsankar Mukhopadhyay |
|  | 3.2 | AppTest | exitOptionTest() | INPUT: : Instantiate the class PlayCheckersGame  OUTPUT: The default close operation of the resulting JFrame is EXIT\_ON\_CLOSE | Passed | Debsankar Mukhopadhyay |

1. Summary of manual test cases directly corresponding to some acceptance criteria

Summarize how user stories/acceptance criterion are tested manually

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **User Story ID and Name** | **Acceptance Criterion ID** | **Test Case Input** | **Test Oracle (Expected Output)** | **Status** | **Notes** | **Developer Name(s)** |
| 1 | 1.1 |  |  |  |  |  |
|  | 1.2 |  |  |  |  |  |
|  | … |  |  |  |  |  |
| 3 Create an Empty Game Board | 3.1 | Execute the jar file | An empty board with checkers created | Passed |  | Debsankar Mukhopadhyay |
|  | 3.2 | Execute the jar file | An exit option symbol (X) is displayed on the top right corner of the screen | Passed |  | Debsankar Mukhopadhyay |

1. Summary of other automated or manual tests not corresponding to the acceptance criteria

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Number** | **Test Input** | **Expected Result** | **Class Name of the Test Code** | **Method Name of the Test Code** | **Status** | **Developer Name(s)** |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

1. **Meeting Minutes**

Report the minutes of all meetings, including, but not limited to: project/sprint planning meeting, stand-up meeting, backlog grooming, retrospective meeting, and pair programming session.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Date** | **Time and Duration** | **Place** | **Participant Names** | **Purpose of the Meeting** | **Specific Action Items** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

1. **Buddy Ratings**

If you don’t feel comfortable to include your ratings in this report, you may email your ratings to the instructor or grader.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Rating giver* | *Rating receiver* | | | | |  |
|  | Madhuri Sarode | Debsankar Mukhopadhyay | Mounika chinna bandhyala | Neeharika jasti | Ajay reddy byreddy |
| Madhuri Sarode | X | 1 | 1 | 1 | 1 |
| Debsankar Mukhopadhyay | 1 | X | 1 | 1 | 1 |
| Mounika chinna bandhyala | 1 | 1 | X | 1 | 1 |
| Neeharika jasti | 1 | 1 | 1 | X | 1 |
|  | Ajayreddy byreddy | 1 | 1 | 1 | 1 | X |
|  | *Average* | 1 | 1 | 1 | 1 | 1 |