Comprehensive Exercise Report

Team MMS of Section ADB

Manuga Nilwara Rupatunge 211ADB076

Monish Ramchandani 211ADB054

K G Senuri N D Perera 211ADB084

NOTE: You will replace all placeholders that are given in <<>>

[**Requirements/Analysis**](#_uwgqwd5ezv2w) **2**

[Journal](#_lsityg2iq9m6) 2

[Software Requirements](#_2h0vru1u2mla) 3

[**Black-Box Testing**](#_prhaxdxmf8n8) **4**

[Journal](#_18f11w613jft) 4

[Black-box Test Cases](#_2xn4jzot820y) 5

[**Design**](#_24fdizefyocn) **6**

[Journal](#_esp2ocs9j6bk) 6

[Software Design](#_aifbl1x6rddt) 7

[**Implementation**](#_hya8f3jqkba6) **8**

[Journal](#_acupzfhai7gz) 8

[Implementation Details](#_ojhtwkms2z3b) 9

[**Testing**](#_3qvya3vi836q) **10**

[Journal](#_ckfs4xbl5pyr) 10

[Testing Details](#_bzt1547yxzxi) 11

[**Presentation**](#_hdjvrbf45b1p) **12**

[Preparation](#_xbiquwtmf36n) 12

[**Grading Rubric**](#_u0hfnmdgusmf) **13**

# Requirements/Analysis

Week 2

## Journal

The following prompts are meant to aid your thought process as you complete the requirements/analysis portion of this exercise. Please respond to each of the prompts below and feel free to add additional notes.

* After reading the client’s brief (possibly incomplete description), write one sentence that describes the project (expected software) and list the already known requirements.
  + In a board of 6 by 7 in which two player puts the disc alternatively and the player who gets four discs in a row without a break wins the game.
    - The parameters of the grid should be 6 rows and 7 columns.
    - There must be 21 discs of each color in total 42 discs
    - Player can drop disc either on middle or on the edge of grid.
    - Two different color of disc is required to each player.
    - In order to win the game player need to get 4 in a row, It can be in row in column or diagonally.
    - Disc can be stacked up upward horizontally and diagonally.
    - The game will automatically stop after the win the winners name and the greeting will be pop up on screen.
    - There will be an option to restart the game after it ends.
* After reading the client’s brief (possibly incomplete description), what questions do you have for the client? Are there any pieces that are unclear? After you have a list of questions, raise your hand and ask the client (your instructor) the questions; make sure to document his/her answers.
  + Are there any unique solution forms which we can take references.
  + When is the deadline of the project.
  + Do we need some kind of special visuals.
  + IS there any specific technology, framework, database or any other programming language which you prefer in order to develop the software.
  + Are there any limitations with the budget.
* Does the project cover topics you are unfamiliar with? If so, look up the topics and list your references.
  + Play with more than two players.
  + Game AI Bot for single player.
* Describe the users of this software (e.g., small child, high school teacher who is taking attendance).
  + Anyone who wants to exercise their brain muscle and understands the rules and goals of the game can play this game. It can be either a small child or an elder person.
* Describe how each user would interact with the software.
  + They can simply play it on computer or laptop by using the mouse or trackpad.
* What features must the software have? What should the users be able to do?
  + Frist by making a simple User-friendly UI which is easy to navigate and understand, so player of any age can enjoy the game without extra effort.
  + There is an option to make friends or challenge friends.
  + The design of the game board should be clean and simple.
* Other notes:
  + There might be a chat box to interact with the other players.

## Software Requirements

<<Use your notes from above to complete this section of the formal documentation by writing a detailed description of the project, including a paragraph overview of the project followed by a list of requirements (see lecture for format of requirements). You may also choose to include user stories.>>

The game is basically the digitalized version of the board game. There is a grid of 6 x 7 a player needs to drop disc either in middle or at the edge of the grid. The player / team need to get 4 disc in a row in order to win the game. Disc can either in a row, or horizontally or diagonally. The software will keep track of how many disc are left for each player or team and give them feedback about their gameplay.  
  
Requirements   
  
●The game must be played by two player either a human or a bot.

●Player should have power or authority to choose the color of their disc before starting the game.

●The game board must be 6 x 7 grid.

●There should be total of 21 disc of each color for each player.

●Player can be dropped at the middle of the game.  
●Disc stacked upward horizontally and diagonally.  
●The software should provide a restart after each round.

● users to play using a mouse and trackpad.

User stories:

● user said, they would prefer simple user interface.

● user said, they would prefer there is a option to choose disc color.

● user said, they want to be able to stack my discs upwards, horizontally, or diagonally.

● user said, they want to be able to win the game by getting four of my discs in a row either horizontally,

vertically, or diagonally.  
● user said, they want the option to play with friends

● user said, after I win display a winning greeting.

● user said, they want the option to restart the game after it ends.

# Black-Box Testing

Instructions: Week 4

## Journal

***Remember:*** Black box tests should only be based on your requirements and should work independent of design.

The following prompts are meant to aid your thought process as you complete the black box testing portion of this exercise. Please review your list of requirements and respond to each of the prompts below. Feel free to add additional notes.

● What does input for the software look like (e.g., what type of data, how many pieces of data)?

○ Players will be adding their disks in the columns and that would be the input.

○ Player can decide who they want to play against, it could be another player online or a robot.

○ Starting the game, pausing and resuming and ending the game is always a choice.

● What does output for the software look like (e.g., what type of data, how many pieces of data)?

○ As an output the game should display the game board in each and every move the players make.

○ At the end of the game, the winner’s name should be displayed, along with the “you won!” message.

○ Amidst the game, the current points and the status of the players will be displayed in the score board.

● What equivalence classes can the input be broken into?

○ Each player’s ID or the name of the player, can be taken as an equivalence class.

○ Also, which player’s turn is it to move the disk and which place he/she moves the disk.

○ Option that the players can select, human vs human, human vs robot.

● What boundary values exist for the input?

○ In the game board there has to be a yes or no option.

○ The maximum value the game board can have been the ‘full’ state and the minimum value the game can be having the ‘empty’ state.

● Are there other cases that must be tested to test all requirements?

○ Invalid input - This includes trying the place the disk in the wrong range of boxes which is not given by the game or trying to insert the wrong format of data into the game.

○ Valid input – This includes the players placing the disks correctly in the correct format and within the given or suggested range of the game board.

○ Error conditions – This is made sure that the game is properly following the conditions given about the errors that could happen while playing the game. This is to testify that the conditions about the errors such as not allowing the players to play the disks outside the range, the same player cannot play constantly, are being properly marked as errors and handled well.

● Other notes:

○ User interface – This is to make sure that the user interface is easy to navigate, user friendly and the performance is in good quality and its fun and interesting for the players.

## Black-box Test Cases

Use your notes from above to complete the black-box test plan section of the formal documentation by writing black box test cases (other than actual results since no program currently exists). Remember to test each equivalence class, boundary value, and requirement.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Description** | **Expected Results** | **Actual Results** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Design

Instructions: Week 6

## Journal

***Remember:*** You still will not be writing code at this point in the process.

The following prompts are meant to aid your thought process as you complete the design portion of this exercise. Please respond to each of the prompts below and feel free to add additional notes.

* List the nouns from your requirements/analysis documentation.
  + <<Insert answer>>
* Which nouns potentially may represent a class in your design?
  + <<Insert answer>>
* Which nouns potentially may represent attributes/fields in your design? Also list the class each attribute/field would be a part of.
  + <<Insert answer>>
* Now that you have a list of possible classes, consider different design options (***lists of classes and attributes***) along with the pros and cons of each. We often do not come up with the best design on our first attempt. Also consider whether any needed classes are missing. These two design options should not be GUI vs. non-GUI; instead you need to include the classes and attributes for each design. Reminder: Each design must include at least two classes that define object types.
  + <<List at least two design options with pros and cons of each>>
* Which design do you plan to use? Explain why you have chosen this design.
* List the verbs from your requirements/analysis documentation.
  + <<Insert answer>>
* Which verbs potentially may represent a method in your design? Also list the class each method would be part of.
  + <<Insert answer>>
* Other notes:
  + <<Insert notes>>

## Software Design

<<Use your notes from above to complete this section of the formal documentation by planning the classes, methods, and fields that will used in the software. Your design should include UML class diagrams along with method headers. ***Prior to starting the formal documentation, you should show your answers to the above prompts to your instructor.****>>*

# Implementation

Instructions: Week 8

## Journal

The following prompts are meant to aid your thought process as you complete the implementation portion of this exercise. Please respond to each of the prompt below and feel free to add additional notes.

* What programming concepts from the course will you need to implement your design? Briefly explain how each will be used during implementation.
  + <<Insert answer>>
* Other notes:
  + <<Insert notes>>

## Implementation Details

<<Use your notes from above to write code and complete this section of the formal documentation with a README for the user that explains how he/she will interact with the system.>>

# Testing

Instructions: Week 10

## Journal

The following prompts are meant to aid your thought process as you complete the testing portion of this exercise. Please respond to each of the prompts below and feel free to add additional notes.

* Have you changed any requirements since you completed the black box test plan? If so, list changes below and update your black-box test plan appropriately.
  + <<Insert answer>>
* List the classes of your implementation. For each class, list equivalence classes, boundary values, and paths through code that you should test.
  + <<Insert class>>
    - <<Insert needed tests>>
  + <<Insert class and tests for each class>>
* Other notes:
  + <<Insert notes>>

## 

## 

## Testing Details

<<Use your notes from above to write your test programs and complete this section of the formal documentation by creating a list of your test programs along with descriptions of what they are testing. You will also complete the black-box test plan by running the program and filling in the Actual Results column.>>

# Presentation

Instructions:Week 12

## Preparation

The following prompts are meant to aid your thought process as you complete the presentation portion of this exercise. It is recommended that you examine the previous sections of the journal and your reflections as you work on the presentation as it is likely that you have already answered some of the following prompts elsewhere. Please respond to each of the prompts below and feel free to add additional notes.

* Give a brief description of your final project
  + <<Insert answer>>
* Describe your requirement assumptions/additions.
  + <<Insert answer>>
* Describe your design options and decision. How did you weigh the pros and cons of the different designs to make your decision?
  + <<Insert answer>>
* How did the extension affect your design?
  + <<Insert answer>>
* Describe your tests (e.g., what you tested, equivalence classes).
  + <<Insert answer>>
* What lessons did you learn from the comprehensive exercise (i.e., programming concepts, software process)?
  + <<Insert answer>>
* What functionalities are you going to demo?
  + <<Insert answer>>
* Who is going to speak about each portion of your presentation? (Recall: Each group will have ten minutes to present their work; minimum length of group presentation is seven minutes. Each student must present for at least two minutes of the presentation.)
  + <<Insert answer>>
* Other notes:
  + <<Insert notes>>

<<Use your notes from above to complete create your slides and plan your presentation and demo.>>