Comprehensive Exercise Report

Team LogBaiters of Section <<ADB>>

Eltaj Abbasov 211ADB072  
Majid Aghayev 211ADB088

[**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.
  + A digital version of the classic Connect Four game, allowing players to compete against each other.
    - Accurate replication of traditional Connect Four rules and mechanics.
    - Clean user interface with customizable appearance options.
    - Implementation of local multiplayer mode.
    - Digital Platform compatibility across desktop
* 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 specific visual customization options for the game board? - Traditional color schemes and themes for the game board and discs
  + Any preferences regarding the design or layout of the user interface? - Minimalist design for the user interface, focusing on clarity and ease of use.
* Does the project cover topics you are unfamiliar with? If so, look up the topics and list your references.
  + No unfamiliar topics.
* Describe the users of this software (e.g., small child, high school teacher who is taking attendance).
  + Users can vary from casual gamers, friends, and family members looking for a fun or just a gaming experience together.
* Describe how each user would interact with the software
  + Users interact by selecting side, making moves on the grid, and enjoying the game with friends or family in a local multiplayer setting.
* What features must the software have? What should the users be able to do?
  + Local Multiplayer Mode
  + Accurate Gameplay Replication
  + Intuitive User Interface
  + Restart Game or Rematch
  + Users should be able to :
    - Start a new game.
    - Play against a friend locally.
    - Enjoy smooth and immersive gameplay.
* Other notes:
  + Make sure simplicity and user-friendly design is implemented for players of all ages can enjoy the game.
  + Testing to find and resolve any bugs or issues, in order to provide a smooth gaming experience.

## 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.>>

# 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)?
  + Player moves: Column number (1-7)
  + Game start/reset commands
  + User interface interactions (e.g., selecting side, initiating a new game)
* What does output for the software look like (e.g., what type of data, how many pieces of data)?
  + Updated game board after each move
  + Game status messages (e.g., "Player 1 wins")
  + Visual updates on the user interface
* What equivalence classes can the input be broken into?
  + Valid player moves (columns 1-7)
  + Invalid player moves (columns < 1 or > 7)
  + Full columns (attempting to place a disc in a column that is already full)
  + Game status (player 1 wins, player 2 wins)
* What boundary values exist for the input?
  + Minimum and maximum column values (1 and 7)
  + Edge cases for column full condition
  + Winning move scenarios (four discs in a row horizontally, vertically, and diagonally)
* Are there other cases that must be tested to test all requirements?
  + Validating that the game maintains accurate state transitions and updates the user interface consistently with the game logic.
* Other notes:
  + Not Any

## 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** |
| T1 | Verify that a disc is added to the correct column when a valid move is made. | Disc appears in the chosen column by player | Disc appears in the chosen column by player |
| T2 | Verify game status message when player 1 wins with a horizontal line. | Game status updates to "Player 1 wins". | Game status updates to winner player |
| T3 | Verify the game restart functionality. | Game board resets to the initial state, ready for a new game. | Game board resets to the initial state, ready for a new game. |

# 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.>>