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

Connect Four is a two-player board game where the objective is to be the first to connect four of your discs in a row, either vertically, horizontally, or diagonally. The game is played on a vertical grid with six rows and seven columns. Players take turns dropping their colored discs into any of the columns, with the discs falling to the lowest available space. The game ends when a player connects four discs in a row or the grid is filled without a winner, resulting in a draw.

The software requirements will be :

* Game Interface:
  + Allow players to drop discs into columns.
  + Provide a real-time view of the game state.
  + Implement the rules of Connect Four.
  + Ensure the design is responsive and adapts to different screen sizes.
* Game State Management:
  + Manage and update the game state continuously.
  + Store the positions of the discs on the grid.
  + Handle turns between players.
* Additional Features:
  + Option to restart the game.
  + Disk color selection for players.

# 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. |
| T4 | Verify the game status when the game ends in a draw. | Game status updates to "Draw" | Game status updates to "Draw", restart button appears. |
| T5 | Verify that a disc cannot be placed in a full column. | Error message displayed "Column full". | Error message displayed "Column full". |
| T6 | Verify visual updates on the user interface after each move. | The user interface correctly highlights the changes, including the latest disc placed, the winning line, or full columns. | The user interface correctly highlights the changes, including the latest disc placed, the winning line, or full columns. |

# 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.
  + Player
  + Color
  + Column
  + Row
  + Game board
  + Menu
  + Button
* Which nouns potentially may represent a class in your design?
  + Player
  + Game Board
* Which nouns potentially may represent attributes/fields in your design? Also list the class each attribute/field would be a part of.
  + Player
  + Color
  + Game Board
  + Button
* 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.

Option 1: Single Class for ConnectFour

* + Pros:
    - Simplified structure.
    - Easier to implement and maintain.
  + Cons:
    - Limited modularity.
    - Harder to extend functionalities.

Option 2: Separate Classes for Game Logic and UI

* + Pros:
    - Better separation of concerns.
    - Easier to test and extend.
  + Cons:
    - Increased complexity.
    - Requires more coordination between classes.
* Which design do you plan to use? Explain why you have chosen this design.
  + Option 1: Using a single ConnectFour class for simplicity and ease of implementation.
* List the verbs from your requirements/analysis documentation.
  + Drop
  + Make Move
  + Check Winner
  + Restart
  + Update
  + Disable
  + Enable
* Which verbs potentially may represent a method in your design? Also list the class each method would be part of.
  + make\_move
  + check\_winner
  + restart\_game
  + update\_board\_colors
* Other notes:
  + Not any

## Software Design

The ConnectFour class represents the main logic and user interface for the Connect Four game. It manages the game state, handles user interactions, and updates the display.



# 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.
  + Classes and Objects: Used for creating the ConnectFour game.
  + Event Handling: Used for button clicks and user interactions.
  + GUI Design: Using tkinter for creating the user interface.
  + Loops and Conditionals: Used for game logic and checking winning conditions.
  + Animation: Used for simulating the disc drop.
* Other notes:
  + Not Any

## Implementation Details

README:

How to Play Connect Four:

Start the Game: Launch the game by running the script. A window with a grid representing the game board and buttons for each column will appear.

Make a Move: Click on the button corresponding to the column where you want to drop your disc. The disc will fall to the lowest available position in that column.

Winning the Game: The game will check for a winning condition (four discs in a row) after each move. If a player wins, a message will be displayed, and you will have the option to restart the game.

Restart the Game: You can restart the game at any time by selecting "Restart Game" from the "Settings" menu.

Change Disc Colors: Customize the disc colors by selecting "Player Color Setup" from the "Settings" menu. You can choose colors via text input or a color picker.

System Requirements: Python 3.x , tkinter library (comes with standard Python distribution)

# 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.
  + None of the requirements changed.
* List the classes of your implementation. For each class, list equivalence classes, boundary values, and paths through code that you should test.
  + ConnectFour
    - Equivalence Classes:
      * Player Moves:
        + Valid moves (columns 0-6)
        + Invalid moves (columns < 0 or > 6)
        + Moves in a full column
      * Game State:
        + Ongoing game
        + Player 1 wins
        + Player 2 wins
        + Draw
      * User Interface Interactions:
        + Button clicks for columns
        + Menu selections for restart and color setup
    - Boundary Values:
      * Column Numbers:
        + Minimum: 0
        + Maximum: 6
      * Row Numbers (for disc drop):
        + Minimum: 0
        + Maximum: 5
    - Paths Through Code:
      * Path 1: Player makes a valid move.
      * Path 2: Player attempts to make a move in a full column.
      * Path 3: Player makes a winning move.
      * Path 4: Player attempts to make an invalid move (column < 0 or > 6).
      * Path 5: Game ends in a draw.
      * Path 6: User restarts the game.
      * Path 7: User sets up player colors using text input.
      * Path 8: User sets up player colors using color picker.
* Other notes:
  + Unittest in python used

## 

## 

## Testing Details

We used the unittest framework to verify the functionality of the ConnectFour game implemented . The tests ensure that the game initializes correctly, handles player moves properly, identifies winning conditions, and correctly handles attempts to place a disc in a full column. The necessary modules and classes are imported, including unittest for testing, Tk from tkinter for creating the GUI, and game-related constants and classes from connect\_four.py (python file of main program) . We used this approach to cover critical aspects of the game's functionality, ensuring the game works as intended under various scenarios. The code for the testing will be int the github repository.

# 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
  + Implementation of the classic connect four game in python using tkinter library
* Describe your requirement assumptions/additions.
  + Color palette added for the color selection part for the players.
* Describe your design options and decision. How did you weigh the pros and cons of the different designs to make your decision?
  + As the connect four game have traditional and simple design, we sticked to it and had no problems implementing it in the application.
* How did the extension affect your design?
  + We had time to fix some simple errors, such as when user rapidly clicked in the same column game used to crash, but we fixed it.
* Describe your tests (e.g., what you tested, equivalence classes).
  + We tested the necessary aspects of the basic gameplay of the connect four game.
* What lessons did you learn from the comprehensive exercise (i.e., programming concepts, software process)?
  + We had some knowledge in tkinter, which we definitely improved making this application. Also it helped us to improve basic programming concepts as loops and error handling.
* What functionalities are you going to demo?
  + Traditional gameplay of the connect four game.
* 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.)
  + Eltaj will start the presentation by briefly explaining the key concepts of the connect four game and requirements. Majid will finish the presentation, by explaining the programs used and game features. Also we both will try to answer any questions if any, about the game and its implementation.
* Other notes:
  + GitHub link : <https://github.com/whoseltaj/DIP392-LogBaiters>