Comprehensive Exercise Report

Team **Alpha** of Section **007**

Team members: Daniel Hudhra, Denisa Kercanaj, Glears Canaj, Ebube Mbakogu

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

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.

  - After reading the client's brief, the project involves developing software for a digital version of Connect

  Known requirements include:

- Ability to display the game board and pieces
- Support for two players to take turns placing their pieces
- Winning condition detection
- User interface for player interactions

- 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.

  - Questions for the client:

- Are there any specific rules or variations of Connect 4 we should consider?
- Should the game have options for different board sizes or difficulty levels?
- Will there be any additional features like a replay option or AI opponents?

Answers from instructor:

1. Standard Connect 4 rules apply.
2. Stick to the standard board size.
3. No additional features required for now.

- Does the project cover topics you are unfamiliar with? If so, look up the topics and list your references.

  AI algorithms for game playing, graphical user interface development.

- Describe the users of this software (e.g., small child, high school teacher who is taking attendance).

  Users will be players interested in playing Connect 4 against each other.

- Describe how each user would interact with the software

  Users will interact through a graphical user interface where they can select columns to drop their pieces, observe the game board, and receive notifications about game outcomes.

- What features must the software have? What should the users be able to do?

  Game board display
  Player turn management
  Winning condition detection

Requirements/Analysis
Week 2

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 Connect 4 digital version project aims to develop software that provides an engaging and intuitive platform for players to enjoy the classic game of Connect 4 in a digital format. The software will feature a graphical user interface (GUI) where players can interact seamlessly, taking turns placing their colored discs on the game board. The primary goal of the software is to faithfully replicate the rules and mechanics of the traditional Connect 4 game while offering a user-friendly experience.

Software Requirements:

Game Board Display:

- The software shall display a visually appealing game board consisting of vertical columns and rows where players can drop their discs.
- The game board shall accurately represent the current state of the game, including the positions of the discs placed by each player.

Player Turn Management:

- The software shall support ***two players***, allowing them to take turns placing their colored discs on the game board.
- The system shall enforce turn-based gameplay, ensuring that each player has the opportunity to make a move before switching to the next player.

Winning Condition Detection:

- The software shall detect and identify winning conditions, such as when a player achieves four of their colored discs in a row ***vertically, horizontally, or diagonally***.
- Upon detecting a winning condition, the software shall declare the corresponding player as the winner and display a winning animation or message.

User Interface:

- The software shall provide a user-friendly graphical user interface (GUI) for player interactions.
- The GUI shall allow players to select columns to drop their discs, observe the game board, and receive notifications about game outcomes such as wins, ties, or invalid moves.

User Stories:

- As a player, I want to see a visually appealing game board displayed on the screen, so I can easily understand the game state.
- As a player, I want to take turns with my opponent, so we can compete fairly in the game.
- As a player, I want the software to detect when I achieve four discs in a row, so I can be declared the winner promptly.
- As a player, I want the GUI to be intuitive and easy to navigate, so I can focus on playing the game without confusion.
- As a player, I want the interface to be interactive.
- Also, I do not want the game to be monotonous, but the game should, for example, train the AI to make good decisions, ***preferably using Minimax and Alpha-Beta Pruning algorithms***.

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)?
    - The input for our software consists of the player's selection of a column to drop a disc.
    - This input can be represented as an **integer value** corresponding to the column number.
    - This input will be provided by the player clicking on a column in the graphical user interface.
- What does output for the software look like (e.g., what type of data, how many pieces of data)?
    - **In the context of our "Connect 4" game,** the output for the software includes the updated grid displaying the discs after each player's move, along with notifications of game events such as a player achieving four in a row, a tie game, or prompts for invalid moves.
- What equivalence classes can the input be broken into?
    - In general, the input can be classified into the equivalence classes of valid column numbers (1 to 7) and invalid column numbers (less than 1 or greater than 7).
    - Therefore, in our GUI-based game, the input can be classified into the equivalence classes of valid column selections (clicks within the range of columns displayed on the GUI) and invalid column selections (clicks outside the range of columns).
- What boundary values exist for the input?
    - Boundary values include the minimum valid column number (1), maximum valid column number (7), and values outside this range (0 and 8).
    - In our GUI-based game, the boundary values include the minimum valid column number (the leftmost column displayed on the GUI) and the maximum valid column number (the rightmost column displayed on the GUI).
- Are there other cases that must be tested to test all requirements?
    - Apart from testing valid and invalid column numbers, testing for winning conditions (**horizontal, vertical, diagonal**) and tie game scenarios is necessary to cover all requirements.
- Other notes:
    - It is essential for us to ensure that the game interface functions smoothly, allowing players to *select columns* and *displaying the updated grid accurately after each move*.
    - Moreover, it is fundamental to ensure that the GUI interface functions smoothly.

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	Testing a valid column selection within the range (e.g., column 3).	The disc should be visually placed in the lowest available position in the selected column on the grid.	N/A (since no program currently exists).
T2	Testing an invalid column selection (e.g., clicking on a space outside of the grid).	The GUI should not register the click as a valid move, and the player should not be able to place a disc.	N/A (since no program currently exists).
T3	Testing the winning condition with four discs aligned horizontally.	The game should declare the player who achieved four in a row as the winner, and a winning animation or message should be displayed.	N/A (since no program currently exists).
T4	Testing the winning condition with four discs aligned vertically.	The game should declare the player who achieved four in a row as the winner, and a winning animation or message should be displayed.	N/A (since no program currently exists).
T5	Testing the winning condition with four discs aligned diagonally.	The game should declare the player who achieved four in a row as the winner, and a winning animation or message should be displayed.	N/A (since no program currently exists).
T6	Testing for a tie game scenario where the grid is filled without any player achieving four in a row.	The game should declare a tie, and a tie game message should be displayed.	N/A (since no program currently exists).

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.>>