Connect 4 Report

Team Cloud of Section

Mahmud Yusifli 201ADB064, UMUT KANDEMIR 201ADB101, Mustafa Kemal VURAL 201ADB076

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

[**Requirements/Analysis**](#_heading=h.3znysh7) **[2](#_heading=h.3znysh7)**

[Journal](#_heading=h.2et92p0) [2](#_heading=h.2et92p0)

[Software Requirements](#_heading=h.tyjcwt) [3](#_heading=h.tyjcwt)

[**Black-Box Testing**](#_heading=h.3dy6vkm) **[4](#_heading=h.3dy6vkm)**

[Journal](#_heading=h.1t3h5sf) [4](#_heading=h.1t3h5sf)

[Black-box Test Cases](#_heading=h.4d34og8) [5](#_heading=h.4d34og8)

[**Design**](#_heading=h.2s8eyo1) **[6](#_heading=h.2s8eyo1)**

[Journal](#_heading=h.17dp8vu) [6](#_heading=h.17dp8vu)

[Software Design](#_heading=h.3rdcrjn) [7](#_heading=h.3rdcrjn)

[**Implementation**](#_heading=h.26in1rg) **[8](#_heading=h.26in1rg)**

[Journal](#_heading=h.lnxbz9) [8](#_heading=h.lnxbz9)

[Implementation Details](#_heading=h.35nkun2) [9](#_heading=h.35nkun2)

[**Testing**](#_heading=h.1ksv4uv) **[10](#_heading=h.1ksv4uv)**

[Journal](#_heading=h.44sinio) [10](#_heading=h.44sinio)

[Testing Details](#_heading=h.3j2qqm3) [11](#_heading=h.3j2qqm3)

[**Presentation**](#_heading=h.1y810tw) **[12](#_heading=h.1y810tw)**

[Preparation](#_heading=h.4i7ojhp) [12](#_heading=h.4i7ojhp)

[**Grading Rubric**](#_heading=h.2xcytpi) **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.
  + Connect Four is a two-player connection rack game, in which the players choose a color and then take turns dropping colored tokens into a seven-column, six-row vertically suspended grid.
* Functional Requirements:
  + - The software should allow users to set up the game by choosing their colors (red or yellow) and placing the discs in the grid.
    - The software should enable turn-based gameplay, where each player takes turns dropping their discs into the grid.
    - The software should allow players to place their discs in the grid by clicking on the desired location on the grid.
    - The software should check for a win condition after each disc placement, and declare a winner if a player has achieved four discs of the same color in a row (horizontally, vertically, or diagonally).
    - The software should check for a tie condition if the grid is completely filled with discs and no winner has been declared.
    - The software should allow players to restart the game from the beginning.
* Non-functional Requirements:
  + - The software should be optimized for performance to ensure smooth and seamless gameplay.
    - User Interface: The software should have a user-friendly interface that is easy to navigate and understand.
    - Cross-Platform Compatibility: The software should be designed to work on a variety of platforms and devices.
    - Graphics and Design: The graphics and design of the software should be visually appealing and enhance the user experience.
    - The software should be accessible to a wide range of users, including those with disabilities.
    - The software should adhere to data privacy regulations and protect the user's personal information. This could include features such as secure login, encryption, and data anonymization.
    - The software should be available in multiple languages to accommodate users who speak different languages.
* Constraints:

Hardware Constraints: The software should be designed to work on a variety of hardware platforms, including desktop and mobile devices.

Time Constraints: The software should be delivered within a specific timeframe to meet project deadlines.

Budget Constraints: The development of the software should adhere to a specific budget to ensure cost-effectiveness.

* 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.
  + <<Insert your questions and your instructor’s answers>>
* Does the project cover topics you are unfamiliar with? If so, look up the topics and list your references.
  + We are almost familiar with project cover topics.
* Describe the users of this software (e.g., small child, high school teacher who is taking attendance).
  + The users of the classic game of Connect 4 software can vary but the general target are children and teens.

Children: Connect 4 is a game that is easy to learn and can be enjoyed by children of different ages. Children can play the game against their friends or a computer opponent in single-player mode.

Teens: Connect 4 can also be enjoyed by teenagers who are looking for a quick and fun game to play with their friends or family members. The game's simplicity and short play time can make it an ideal option for social gatherings or parties.

* Describe how each user would interact with the software
  + Users would interact with the software by launching the game and selecting the single-player or two-player mode. In single-player mode, they would select the difficulty level and start playing against the computer player. In two-player mode, they would take turns selecting a column and dropping their color of disc into the game board.
* What features must the software have? What should the users be able to do?
  + Game Mode
  + Difficulty level
  + Game Board
  + Sound effects
  + Player turn indicator
* Other notes:
  + <<Insert notes>>
  + —-

## 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 software requirements for implementing the classic game of Connect 4 can include:

Programming Language—We will use Python

Integrated Development Environment (IDE)--- An IDE such as Visual Studio Code to write, run, and test the code.

Graphical User Interface (GUI) Framework—A GUI framework such as JavaFX, PyQt, or Tkinter to create the game interface.

Data Structures— We need to implement data structures such as a 2D array to represent the game board, an array or list to keep track of the player moves, and a stack or list to implement the undo/redo feature.

Artificial Intelligence: We need to use algorithms such as minimax or alpha-beta pruning to create an intelligent opponent.

Version Control System: A version control system such as Git to manage the codebase, collaborate with other developers, and maintain the project history.

Documentation: You need to document the code and the project to make it easier for others to understand and maintain it.

Testing Framework: A testing framework such as JUnit, pytest, or unittest to create unit tests and integration tests to ensure that the game functions as expected.

Deployment: If we plan to deploy the game on a website or as a mobile app, we need to use technologies such as HTML, CSS, JavaScript, React, or React Native to create the user interface and deploy the code on a server or app store.

# 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)?
  + <<Insert answer>>
* What does output for the software look like (e.g., what type of data, how many pieces of data)?
  + <<Insert answer>>
* What equivalence classes can the input be broken into?
  + <<Insert answer>>
* What boundary values exist for the input?
  + <<Insert answer>>
* Are there other cases that must be tested to test all requirements?
  + <<Insert answer>>
* Other notes:
  + <<Insert notes>>

## 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.>>