Comprehensive Exercise Report

Team Team-21 of Section ADB

Ibrahim Velizade 211ADB065

Farid Aghazada 211ADB092

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.
    - Standard Connect 4 game with GUI that is played with 2 people locally and follow the traditional Connect 4 rules
        - Graphical User Interface
        - Game rules
        - Played locally
        - Ability to start a new game or exit the application
        - 2-player
- 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.
    - Q: Are there any specific design preferences for the graphical user interface (GUI)? A: It should resemble the physical game set, i.e., the color of the disks
    - Q: Are there any specific design preferences for the graphical user interface (GUI)? A: No, you can choose what you wish
    - Q: On which devices should a user be able to play this game? A: It should be a desktop application
    - Q: Are there any additional features or functionalities you would like to include other than the basic requirements listed? A: It is optional
- Does the project cover topics you are unfamiliar with? If so, look up the topics and list your references.
    - GUI implementation
    - Desktop game development
    - Game algorithms
- Describe the users of this software (e.g., small child, high school teacher who is taking attendance).
    - Everyone who is familiar with game rules
- Describe how each user would interact with the software
    - Players will be able interact with the game by using mouse/touchpad to click relevant buttons in the game
- What features must the software have? What should the users be able to do?
    - To restart the game
    - To see the game outcomes (win/lose)
    - Users should be able to make moves by dropping tokens into columns.
- Other notes:
    - Overall it should give the impression of the playing with the physical game set

Software Requirements

The software is the web application of the classic Connect 4 game. Connect 4 is a two-player strategy game where the goal is to be the first to form a line of four of your colored discs in a row, horizontally, vertically, or diagonally on a grid. The game is played on a vertical grid with 6 rows and 7 columns, totaling 42 spaces. It is played with red and yellow disks. Players take turns dropping one of their colored discs from the top into any of the columns. The disc then falls to the lowest available space in that column. To win the game players need to build their own line of four while also strategically blocking their opponent from forming their own line. The game ends when one player successfully connects four of their discs in a row, and that player is declared the winner. If the board fills up without a player connecting four discs, the game is a draw.

In the main page where the gaming takes place, there is a board and disks inside the spaces like in the physical game. During the game the players take turns clicking on the column where they want to drop the disk (red or yellow depending on whose turn it is) and it continues till the end of the game. When the game ends users have a choice to exit the game or replay.

The software requirements are the following:

- Game Interface: Develop a user-friendly interface that allows players to interact with the game, i.e., drop discs into columns, and view the game state in real-time.
- Game Logic: Implement the rules of Connect Four
- Responsive Design: Ensure the web application is responsive and works across various devices and screen sizes
- Game State Management: Implement a system to manage and update the game state, including storing the position of discs on the grid and handling player turns.
- User Experience Enhancements: Provide visual feedback when hovering over columns to indicate where the disc will drop. Animations (such as disc dropping and winning sequences) can enhance the user experience.
- Additional Features: Ability to restart game at any given moment.

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)?
    - Mouse click events: player actions, such as clicking on a column to drop a disc or to restart the game. The number of mouse clicks depends on the progress and the state of the game.
- What does output for the software look like (e.g., what type of data, how many pieces of data)?
    - Visual feedback on the game interface: the updated game grid after each player's turn, indicating whose turn it is, and displaying messages for game outcomes (win, draw).
- What equivalence classes can the input be broken into?
    - Valid input: actions that follow the game rules, such as clicking on a valid column to drop a disc.
    - Invalid input: actions that violate the game rules, such as attempting to drop a disc in a column that is already full or clicking outside of the game area.
- What boundary values exist for the input?
    - Position of the mouse click: it should be on the game grid that consists of 6 rows and 7 columns or on the other available buttons
- Are there other cases that must be tested to test all requirements?
    - Testing if restarting the game works.
    - Testing if the user experience enhancements (e.g., hovering on columns to indicate where the disc will drop) work properly
    - Testing the responsiveness of the web application across various devices and screen sizes to ensure consistent gameplay
- Other notes:
    - Tests can be done in different browsers to see if there is any difference in the results

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**
1	Valid input: Clicking on a valid column to drop a disc.	Disc is dropped in the selected column. The Game grid is updated accordingly.	Disc is dropped in the selected column. The Game grid is updated accordingly.
2	Invalid input: Clicking on a column that is already full.	No disc is dropped. Player receives a notification indicating the invalid move.	No disc is dropped. Player receives a notification indicating the invalid move.
3	Invalid input: Clicking outside of the game area and buttons	No action is taken. The game state remains unchanged.	No action is taken. The game state remains unchanged.
3	Testing the response when the grid is full	Game ends in a draw. Players are notified of the draw outcome.	Restart button appears
4	Testing if restarting the game works.	Game state is reset to the initial state. Players can start a new game.	Game state is reset to the initial state. Players can start a new game.
5	Testing whether cell color can be changed.	User can change cell color by clicking on it.	User can change cell color by clicking on it in the settings page.
6	Testing responsiveness across various devices and screen sizes	Game interface adjusts smoothly to different screen sizes. Gameplay remains consistent and enjoyable across devices.	Game interface adjusts smoothly to different screen sizes. Gameplay remains consistent and enjoyable across devices.

Design

Instructions: Week 6

Journal

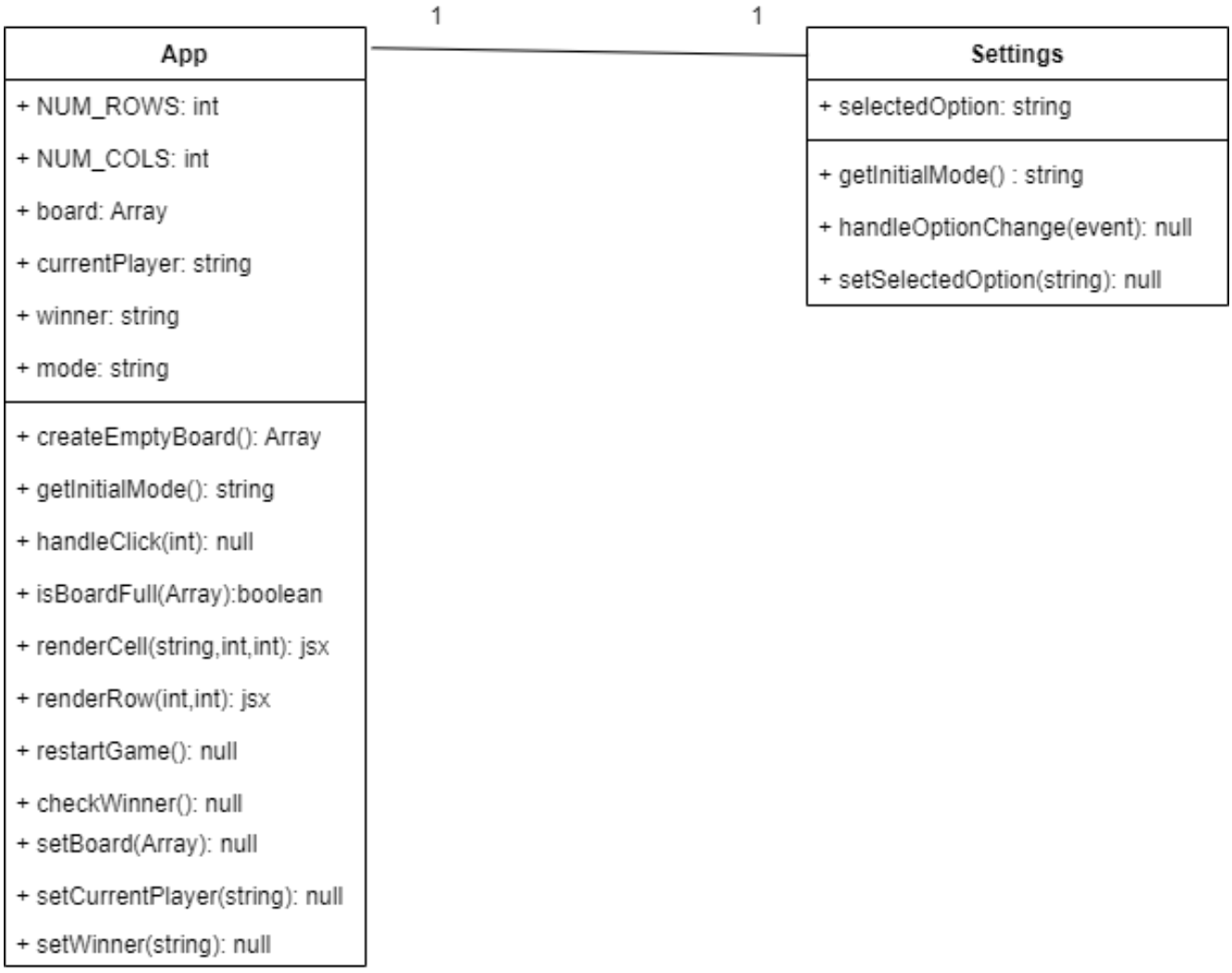
Remember: You still will not be writing code at this point in the process.

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
    - disc color
    - cell
    - board
    - game results
- Which nouns potentially may represent a class in your design?
    - cell
    - board
- Which nouns potentially may represent attributes/fields in your design? Also list the class each attribute/field would be a part of.
    - disc color: cell
- 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.
    - Two basic approaches are the Monolithic approach and Separation approach.
    - Monolithic design is simple, just uses 1 class for all the logic and functions. Pros include simplicity and great for small projects like connect 4. Cons include testing problems and harder management when the project becomes big.
    - On the other hand Separation of Concerns there are multiple classes based on the page or functionality that page performs. It is like lego pieces that developers can connect at the end. Pros of this design include following single responsibility principle and easier testing. Cons of this design include more planning beforehand.
- Which design do you plan to use? Explain why you have chosen this design.
    - We chose the design “Separation of Concerns” because it is much more reliable and adaptable than the Monolithic approach, additionally we have experience working with this model.
- List the verbs from your requirements/analysis documentation.
    - Click
    - Drop
    - Update
    - Display
    - Restart
- Which verbs potentially may represent a method in your design? Also list the class each method would be part of.
    - Restart: App
    - Click: App
    - Display: App

Software Design

As described above design pattern “Separation” will be used. Since we are planning to have 2 separate pages, 1 for settings and second for the game, there will be 2 classes responsible for the game. Classes are called “Settings” and “App”(which is a default name of the react app project). Functions and attributes are listed in the UML diagram down below. While setting class is responsible for the changing option of background image, App class is doing main logic of the game as well as retrieving information that can be changes from local Storage.



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.
    - We will mostly utilize OOP and Agile development. Since we are working on javascript, utilization of OOP is lethal for quality coding. Additionally utilization of Agile development tactics will be used to keep track of the development flow. Lastly, usage of user interface evaluation will be used.
- Other notes:

Implementation Details

The system has a very simple design. In the main page there is a virtual Connect 4 board.

Here is how the game is played:

- The users can start a game instantly just by clicking on one of the columns of the board.
- The default disks are red and yellow, however before starting the game users can click on the “Settings” button where they can choose to play with another set of discs.
- The player that makes the first move is assigned to play with the red discs, and the other player plays with the yellow discs.
- The players make moves one by one by clicking on one of the columns where they want the disc to drop.
- The players can only drop the discs on the columns that are not full.
- The game ends with a win of a player who manages to form a horizontal, vertical, or diagonal line of four of one’s discs or with a draw when the board is full but none of the players formed such a pattern.
- When the game ends, users can click on the “Restart Game” button that will start a new game.

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.
    - Yes we removed cell hovering animation from our requirements.
- List the classes of your implementation. For each class, list equivalence classes, boundary values, and paths through code that you should test.
    - Equivalence classes include all the functions inside of the App and Settings classes like : createEmptyBoard, resetGame, handleOptionChange and so on. All the functions are mentioned in the UML diagram above.
    - Boundary values for these function are not actually present since our application is more UI oriented. Boundary values include boundary of the defined board, meaning outside of the board is non clickable.
    - Since our application have both Settings and App page it uses React pages for navigation thus there are 2 paths. App is rendered at “/main” and the settings page is rendered at “/settings” path.
        - All the necessary tests are present in the git repository inside of “connect-4/src/__tests__”.
        - Visual tests also was done by determining boundaries of the game with the help of google dev tools which is built into browsers.
    - In the end testing phase was done both in programming and in visual level to ensure stability of the application.
- Other notes:
    - Vitest and Jest was used as a testing tools.

Testing Details

There are 3 tests for App.jsx class and 1 for Settings.jsx class.

First test in App class is checking whether the page is loading correctly.

Second test in App class is checking whether the title of the App page is displayed by checking rendering.

And the third test in the App class is checking whether the Settings button is functional.

The test in the Settings class is similar to the 1st test in the App class. It performs the same action by checking if the Settings page is rendered correctly.

Instructions for these tests can be found in our github repository.

All the other test like whether the navigation and setting new background image was done manually since Jest and Vitest have no access to the files with .css extensions.
Responsiveness of the application is also been tested with the “device toolbar” with the help of element inspection. Web application is fully responsible and can be used in mobile and tablet applications.

All the tests done were successful.

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
    - Web application of the classic Connect-4 game
- Describe your requirement assumptions/additions.
    - Hovering animation when hovering over a cell was a requirement that was removed from requirements.
    - “Ability to select different colored cells ” were added as a additional requirement,
- Describe your design options and decision. How did you weigh the pros and cons of the different designs to make your decision?
    - Design-wise there were no problems since we were familiar with working “Separation of Concerns” design that fits React’s Component structure perfectly. Each of us worked on separate components while keeping each other updated on everything. There were no cons in our opinion since React offers multiple libraries that enhance web development like react-dom, react-pages.
- How did the extension affect your design?
    - We got a more clear picture of how our design will be
- Describe your tests (e.g., what you tested, equivalence classes).
    - All the tests are described in the “Testing” section of the document and can also be accessed through our github repository.
- What lessons did you learn from the comprehensive exercise (i.e., programming concepts, software process)?
    - Learned lot about testing libraries used in javascript like Jest and Vitest. Also we learned how to work with different design approaches and making sure that our code stays clean. Learned that neat documentation and requirements are the key to successful application development and working on a good team is much more efficient if you understand your teammate.
- What functionalities are you going to demo?
    - We are going to share a working link to our web application. It is functional both for pc and for smartphones.
- 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.)
    - Ibrahim will speak about the overall system and its requirements while Farid will speak about technologies used.