

UU Game Platform

Project Completion

Prepared for

Feby Rose

Prepared by

Group A

Shiqi Shu

Efdal Erdem

Elsa Grönbeck

Isak Hvarfner

Thom Kunkeler

David Carlsson

Sabereh Hassanyazdi



Project Manager / Main contact

Sabereh

Hassanyazdi

Sabereh.hassanyazdi.9809@student.uu.se

1 Introduction

In the context of our collaboration with a company affiliated with Uppsala University, our team embarked on a challenging yet exciting project to develop UU-GAME, a two-player chess game that can be played via a terminal interface. The initiation of this project was spurred by the explicit requirements and a comprehensive feasibility study conducted by our client, providing a clear path for our developmental efforts.

The essence of this document is to meticulously outline the entirety of our project's life-cycle concerning the UU-GAME application. This includes a thorough delineation of the requirements elicitation process, which entails gathering and specifying the needs as outlined by our client. Additionally, we delve into the risk assessment plan, identifying potential pitfalls and devising strategies to mitigate these risks effectively. Our integration plan also forms a critical part of this document, highlighting our collaborative efforts with other groups to ensure a seamless and efficient development process.

Our approach to the project was underpinned by a series of client meetings, group discussions, and scrum retrospectives. These interactions were instrumental in dividing the workload among team members, fostering a sense of collective responsibility, and ultimately steering us toward the successful completion of the project. Each stage of the project, from the initial design concepts to the final implementation, was guided by the principle of adhering to the specified requirements while managing potential risks and ensuring effective integration with existing systems.

Furthermore, this document aims to provide a comprehensive analysis of the lessons learned throughout the project's duration. It reflects on our methodologies, the challenges encountered, and the strategies employed to overcome these obstacles. By documenting our experiences, we aspire to offer insights that could benefit future projects of a similar nature.

The UU-GAME project represents a confluence of theoretical knowledge and practical application. Through diligent planning, continuous collaboration, and adaptive problem-solving, we have endeavored to deliver a product that not only meets the client's specifications but also enriches our collective experience as developers.

Table of contents

1 Introduction.....	2
Table of contents.....	3
2 Design.....	6
2.1 User classes.....	6
2.2 User requirements.....	7
2.3 Definitions.....	9
2.4 Components.....	14
2.5 Functional requirements.....	17
REQ-COOP1.1 Multiplayer gaming experience.....	18
REQ-I1.1 Switching to keyboard input and keyboard UI.....	19
REQ-I3.1 Terminal game.....	21
REQ-I3 Layout of views.....	23
REQ-R2 The transfer of stones within the board.....	24
REQ-R3 Building a road with flat stones.....	26
REQ-R4 Draw condition.....	28
REQ-R6 Opponent's color initial stone placement.....	29
REQ-R7 Surrender feature.....	30
REQ-R8 Draw feature.....	31
REQ-R9: Flat win.....	32
REQ-R10 Game Over Detection.....	34
REQ-R11: Valid move validation.....	35
REQ-A1: Accessing instructions during the game.....	37
2.6 Non-functional requirements.....	39
Performance.....	41
Usability.....	41
Dependability.....	42
Environment.....	43
Security.....	43
Operational.....	43
Legislative.....	44
Development.....	44
3 Risk Assessment Plan.....	46

3.1 List of potential risks.....	47
3.2 Risk analysis and Planning.....	49
4 Integration Plan and Implementation.....	58
4.1 Integration Plan.....	58
4.2 Activity tasks.....	59
Tabel 1 Activity Table.....	61
Table 2 Activity bar chart.....	62
Justification.....	62
Implementation of the Integration.....	63
6 Team reflection / Lessons learned.....	65
Annex 1: Deliverable D1.....	68
Introductions and Expertise.....	68
Individual interest.....	70
Grade ambitions.....	71
Roles and Responsibilities.....	71
Agile and Scrum workflow.....	76
Collaboration Tools.....	78
Annex 2: Scrum Retrospectives and Kanban Board Snapshots.....	79
Annex 3: Code Project.....	90
Annex 4: Additional Artifacts.....	91
Appendix.....	111
Who have done what for individual contributions:.....	111
Figures.....	113
Refined user requirements.....	126
Group Agreements.....	127
Agreement honoring.....	129
Changelog.....	132

2 Design

In this section we describe the design of our game platform.

2.1 User classes

Student class: Player 1, Student persona: IT student

John 22 is a sophomore student at Uppsala University, studying a Master's Programme in Computer and Information Engineering. John has extensive experience using different computer systems. In John's spare time, he likes to play computer games, and he has even developed a few simpler games himself. John is not afraid to tackle a challenge when it comes to learning new things; instead, he rather enjoys being challenged. John is also the type of person to skip the tutorial when playing a new game, believing that he will learn as he goes. This often causes John to have to look up information about the game's rules and other hints on how to play a certain game. Since John is a sophomore student, he does not have as much time to play games as he wishes. It is therefore important for John that the game is quick to load and that it will not crash, wiping out his progress. A game that lasts for less than 15 minutes would be ideal for John since he can play during breaks. The multiplayer aspect is very important for John, due to his being very competitive while also enjoying the social aspects of playing together. John also has a friend, Sara, that he wants to spend more time with. By playing a computer game with her, John hopes that they can keep in touch even though she has finished school and now has to work.

Student class: Player 2, Student persona: non-IT friend

Sara is a 26-year-old professional working in marketing with a keen interest in the creative arts. She graduated with a degree in graphic design and has been working in the marketing field for the past three years. Sarah enjoys exploring various forms of art, attending music festivals, and experimenting with new recipes in her free time. Despite not having a technical background, she is open-minded and enjoys trying out new experiences. She has recently been asked by her friend, John, to try out a new game with him. Sara has limited experience with video games and is more inclined towards casual, user-friendly games. She enjoys games that are easy to pick up, visually appealing, and have a social aspect to them. Sara might feel overwhelmed if the game involves too many technical details or complex rules, and she would prefer a straightforward and intuitive user interface. While Sara is more likely to prefer mobile games, she is willing to try out keyboard and mouse gaming, as long as the controls of the game are intuitive.

In order to engage Sara, it is fundamental that there are social aspects to the game, incorporating elements for collaboration or friendly competition.

2.2 User requirements

2.2.1 Theme: Cooperation (REQ-COOPx)

1. As a user I want the game to be played between 2 players via terminal.
2. As someone who likes to think before making a move, I want a game that has a turn-based system, allowing me to develop my strategic ability

2.2.2 Theme: Rules (REQ-Rx)

1. As a player, I want to place my standing stones directly on the board or on a flat stone so that I prevent any stones from being placed on top of it.
2. As a player, I want to pick up at most 5 stones from a stack of stones, except the bottom one, so that I can move them.
3. As a player, I want to build a road of my flat stones, visible from above, connecting one edge to the one opposite it, so that I win immediately, unless it's a draw.
4. As a player, I want the game to be a draw if we both win the same turn, so that I won't completely lose if I move stones to complete one of my roads but accidentally complete my opponent's at the same time.
5. As the black player, I want to go first so that it's unlike that famous other board game.
6. As a player, I want the first stone I place to be my opponent's color so that the game is more interesting.
7. As a player, I want to declare defeat when I know I'm playing a losing game so that I can display sportsmanship and get to play the game again sooner.
8. As a player, I want to suggest a draw, and be able to possibly accept defeat if it's rejected, so that I can leave the game early without leaving my opponent hanging, and if the game is even, to not lose because I had to leave.
9. As a player, I want a stalemate from no more empty cells on the board to be decided by counting who has the most stones that count as part of a road, so that fewer games lead to a boring draw.
10. When I, as a player, pick up stones from a stack, starting from the cell I picked them up from, I want to walk cell to cell through their edges and drop at least one of the stones I picked up into each cell I enter, and optionally right where and when I picked them up. I want it to be like if I split a stack of coins with a single hand and move my fingers up to let gravity drop them so that it feels intuitive.

11. As a player, I want to be able to see the rules, whose turn it is and who is winning at all times so that I know how the game works and I make fewer mistakes.
12. As a player, I want to take turns with my opponent so that it's not about reflexes.
13. As a player, I want to either take one of my 16 flat and 5 standing stones and place it on the board, or move already placed stones.
14. As a player, I want to place my stones in the horizontal 5x5 grid of the board.

2.2.3 Theme: Interaction (REQ-Ix)

1. As a player, I want to control the game using the mouse and keyboard so that I only need to learn shortcuts for the common actions to do them quickly and use the mouse where it feels natural to do so.
2. As a player, I want a quick loading time and game to be stable, without any crashes, so that my breaks can be spent playing the game and not troubleshooting.
3. As a player, I want to play in the terminal so that it's easy to control.
4. As a player, I want to control the game using only a keyboard so that the controls I learn can be used even if I play on terminals without mouse support and so that I can keep my hands in one place.
5. As a player, I want to control the game using my touch screen or mouse so that it feels natural.

2.2.4 Theme: Accessibility/Intelligibility (REQ-Ax)

1. As a player or observer, I want the rules to be intuitive and clear, so that I can understand how to play the game without too much difficulty regardless if I play a lot of games in my free time.
2. As a player, I want to understand where stones have been placed and their color, and if they are standing or lying flat, so that I can plan my move accordingly.
3. As a colorblind person, I want to be able to play the game like anyone else, so that I am not at a disadvantage.
4. As a player, I want the game to indicate when it's my turn to play, so I know exactly when to make my move and can stay engaged even when I have a short attention span.
5. As a player, I want all text and instructions in the game to be in a simple and understandable level of English, so that they can be easily comprehended by a wide range of players.
6. As a non-IT student, I want the game to have a simple interface, clear instructions without technical jargon, and easy-to-understand gameplay.

2.3 Definitions

Following

2.3.1 Board grid

The *grid* of the game *board* is a rectangle divided into squares or grid cells. The game takes place on the grid, and the stones (def. 2.3) are placed in the grid cells. The dimensions of the grid are 5x5, but if game design decisions require it, they can be changed at any time.

1. A grid cell is a subdivision part of the grid. The grid cell is a square whose four sides connect either to a different grid cell or the edge of the board.

2.3.2 Player

A *player* refers to a user of the game who is taking part in it. The two players of the game own some amount of stones. The terms *player one*, *player two*, or *opponent* all refer to players.

The player is a user interacting with the game by playing.

1. Player One, or the black player, is the player that generally places black-colored stones. They're the opponent of Player Two.
2. Player Two, or the white player, is the opponent of player one and generally places white-colored stones.
3. Opponent, either player one or player two depending on the context. The opponent of player one is player two and the opponent of player two is player one.
4. Active player, the player that can currently take actions on the board.

2.3.3 Stone

A *stone* is the game piece, used to play the game. The stone can be of two types and be placed in a stack (def. 2.5).

1. Standing stone, the stone is placed perpendicular to the grid.
2. Flat stone, the stone is placed lying horizontally aligned to the grid.

2.3.4 Terminal

A *terminal* refers to an emulator of a physical computer terminal, which consists of a screen that displays text and a keyboard. It can be used by a user to execute applications and to send input to and receive output from certain applications.

2.3.5 Stack

A *stack* is defined as zero or more flat stones placed on top of each other on the grid. Stacks are sometimes labeled as the source, hand, or destination.

1. Source stack, where stones are being moved from.
2. Hand stack, initially populated with stones taken from the source stack, and where the bottom can be dropped on the board.
3. Destination stack, any stack of stones which has had stones placed in it from the hand.

2.3.6 Road

A *road* is one or more of the topmost flat stones in stacks that when viewed from above, visually connect grid cells. A *complete* road connects one edge of the board to the opposite edge.

2.3.7 Box

The *box* is where all stones originally come from.

2.3.8 Flat win

A flat win occurs when both players are out of new stones to place on the board and the board is fully covered, meaning that all of the squares on the board have at least one or more stones of either type on them.

2.3.9 Game Rules

- Players take turns placing one of their 16 flat or 5 standing stones on a 5x5 grid to strategize and win.
- The black player places the first stone.
- The first stone placed must be of the opponent's color to enhance game intrigue.
- A player can make a move by either placing a new stone from their box or by moving a stone or stones on the board.

- Standing stones can be placed on the board or on top of a flat stone. No other stone can be placed on top of a standing stone.
- Flat stones can be placed on the board and on top of other flat stones.
- When picking up a stone or stones from the board a player has to leave the bottom stone. In other words, a player can only pick up a stone if it belongs to a stack of at least two stones. When picking up stones from a stack a player can pick up a maximum of five stones.
- When placing stones a player has picked up from a stack the player can only place the stones in adjacent squares to where they first picked up the stones and then adjacent to where they placed the last stone. Adjacent squares are squares that share a common side with the square in question, meaning the square to the left, right, above, and below.
- To win the game, players must build a visible road of flat stones across the board. The road has to connect two sides of the board and the stones making up the road can't have either flat or standing stones on top of them.
- Players can suggest and/or accept a draw, meaning to end the game session without winners or losers, after making their first move.

In-game instructions

These are the short version of the rules that will be displayed in-game. They are shortened due to the context from the visible elements of the game, temporary messages displayed to each player the first round, and so they can be displayed together with the rest of the game while still fitting in a normal-size terminal window in an editor such as VS Code.

Short instructions:

This is a flat stone ■ and this is a standing stone ●.

To win, visually connect opposite borders with ■.

Alt. have most visible ■s when the board is covered.

Win with the same turn as your opponent to draw.

Place 1 ■/● a turn or transfer some from a stack.

Transfer ≤ 5 stones adjacent to/back into a stack.

The bottom stone of a stack can't be moved.

Can't balance stones on ● ∴ it prevents stacking.

To make a move use the mouse/keyboard.

Mouse: click to select type, pick up and drop

WASD/↑←↓→: choose where to place stone on board.

Select type Standing: 1. Flat: 2. Space: pick up/drop.

To suggest a draw: Click draw-button/press U: black or T: white.

Answer Yes: Click yes-button/press Y.

Answer No: Click no-button/press N.

To admit a defeat: Click defeat-button/press M

Press Q: quit

Full instructions:

This game has a white player and a black player.

The game pieces are called stones and have two types: flat and standing.

This is a flat stone ■ and this is a standing stone ●.

A player wins when they have visually connected opposite borders with a road of flat stones.

To count towards a road the stones can't have another stone on top of them.

When the board is fully covered and both players have placed all their stones on a flat win occurs. This means that the player with the most visible flat stones wins.

If both players build a full road at the same turn the game comes to a draw.

The black player places the first stone.

For the first turn you place a stone of your opponent's color from your box.

For the rest of the game you only place stones of your own color.

When it's your turn you can either place a new stone from your box or pick up a stack of stones from one of the squares on the board.

The bottom stones of a stack can't be moved. This means that you can only pick up stones from the board if there are more than one stone in the square.

When picking up a stack of stones from the board you can pick up a maximum of 5 stones. When placing these stones you have to place them in an adjacent square. First in an adjacent square to the square you picked up the stack from. After that an adjacent square to where you last placed a stone until you have placed all the stones you picked up.

You can't balance stones on a standing stone • ∴ it prevents stacking.

When making a move you can either use the mouse/keyboard.

With the mouse you can click to select the stone type, pick up stones and drop stones on the board.

Using WASD/↑←↓→ you can choose where to place stones on board.

To select a stone type to place on the board you can press 1 for flat stone and 2 for standing stone.

To drop or pick up a stone press space.

You can suggest a draw at any time after beginning the game.

You can admit defeat after the opponent has denied your draw.

To suggest a draw: Click draw-button/press U: black or T: white.

Answer Yes: Click yes-button/press Y.

Answer No: Click no-button/press N.

To admit a defeat: Click defeat-button/press M

To quit press: Q

2.4 Components

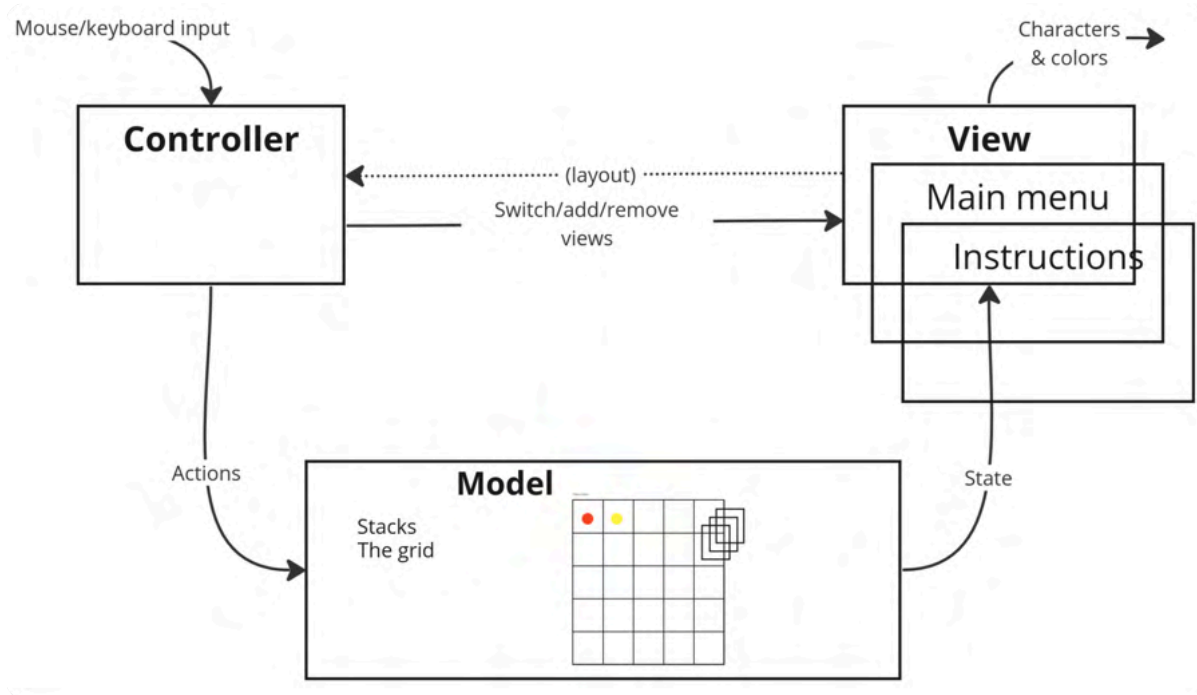


Figure 1.1 MVC framework of UU game

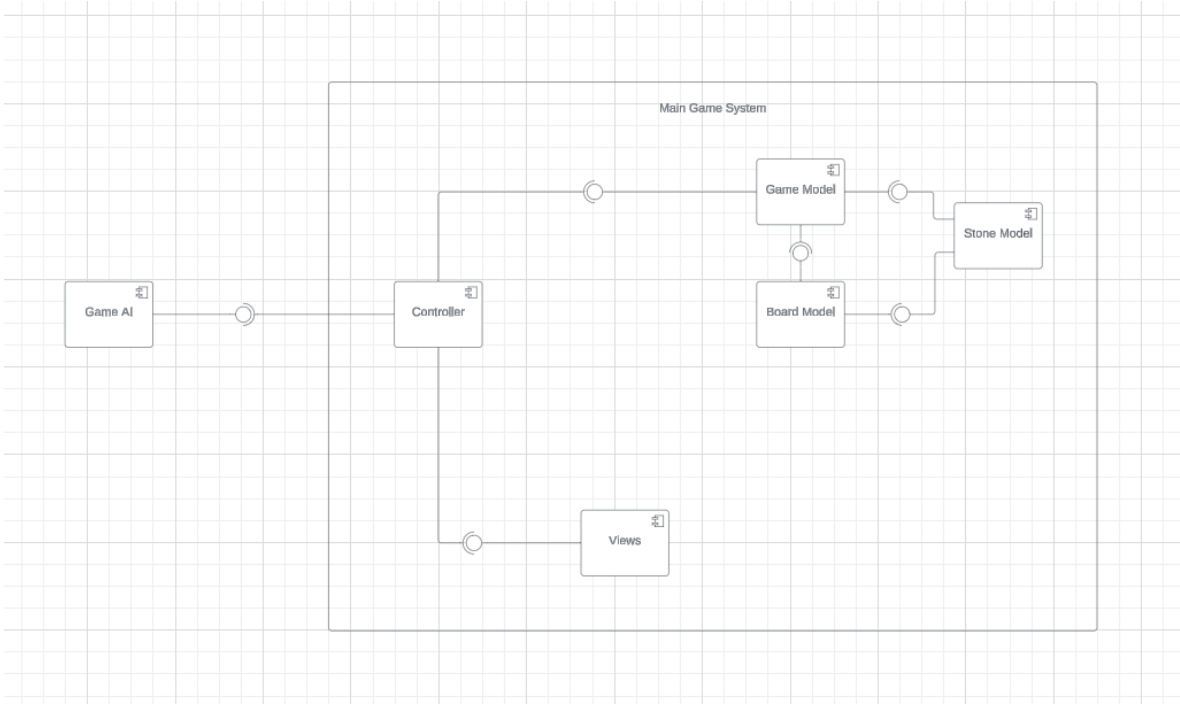


Figure 1.2 Component diagram of UU game

These diagrams illustrate the system architecture for UU Game, figure 1.1 by employing the Model-View-Controller (MVC) framework and figure 1.2 as a component diagram. In the MVC framework, the **Model** component encompasses the application's data and rules, such as Stacks and The Grid. These elements represent the game's state, including the grid and its contents. The **View** components represent the application's user interface, and each is responsible for displaying one part of it, such as the instructions or the board. This is where the entire visualization of the game occurs. The **Controller** component is responsible for handling user inputs, received through a mouse or keyboard, and converting them into actions within the model. It serves as a conduit between the View and the Model, ensuring that the user's interactions are reflected in the game's state and that updates from the model are accurately rendered in the view.

The same architecture is displayed in the component diagram, one exception is that the **Model** component is instead displayed as the three models **Game Model**, **Board Model**, and **Stone Model**.

The following two screenshots show the board view with and without stones.



Figure 2. Screenshot of a game prototype

The game will use the built-in (on macOS and Linux) curses Python library (and the corresponding Windows-curses on Windows) to display the game on a separate terminal “screen”, change character colors and receive mouse and key events.

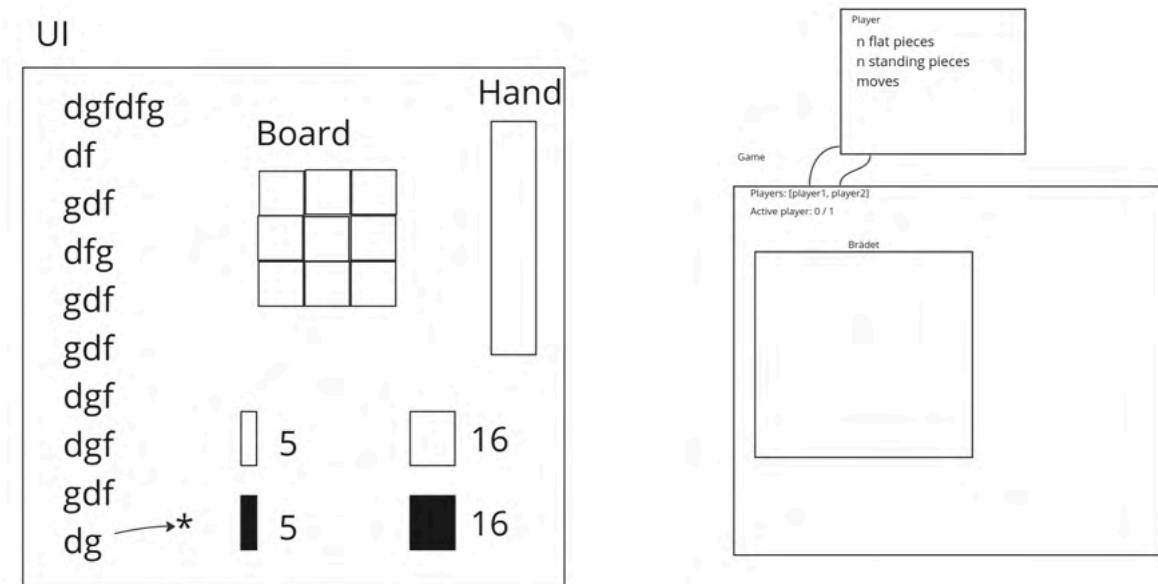


Figure 3. (Left) Initial design of the user interface. (Right) How data is planned to be grouped.

Game

The responsibilities of the **Game** class include managing turns, handling win, tie, or stalemate conditions, and maintaining intermediate states between moves. It also contains the **Box** and the intermediate states between turns created when the hand is used. It also contains player-related data in a data-only class called **Player**.

Board

The first responsibility of the **Board** class is to ensure stacks on it are always in a valid state. The second responsibility is to provide operations and queries directly related to the stones on it, such as if something is part of a road and if it is complete.

Stone

The **Stone** class is a smaller class that handles the color and the stone position, *standing* or *flat*, of a stone.

2.5 Functional requirements

The following sections contain an incomplete description of the system for developers. They reference the user requirements, which can be found in [2.2 User requirements](#).

REQ-COOP1.1 Multiplayer gaming experience.....	15
REQ-I1.1 Switching to keyboard input and keyboard UI.....	16
REQ-I3.1 Terminal game.....	18
REQ-I3 Layout of views.....	20
REQ-R2 The transfer of stones within the board.....	21
REQ-R3 Building a road with flat stones.....	23
REQ-R4 Draw condition.....	25
REQ-R6 Opponent's color initial stone placement.....	26
REQ-R7 Surrender feature.....	27
REQ-R8 Draw feature.....	28
REQ-R9: Flat win.....	29
REQ-R10 Game Over Detection.....	31
REQ-R11: Valid move validation.....	32
REQ-A1: Accessing instructions during the game.....	34

REQ-COOP1.1 Multiplayer gaming experience

Created by	Sabereh Hassanyazdi
Date created	2024-02-06
Primary actor	Player One
Secondary actor	Player Two
Diagrams	Figure 5
User requirement	REQ-COOP1 2.2.1 Theme: Cooperation (REQ-COOPx)

The game allows two players to establish a multiplayer session. From the beginning of the game, both players engage with the game and each other through the session, ensuring an interactive multiplayer gaming experience.

Preconditions

- The game is launched and Both players have access to the game through a shared keyboard device.

Postconditions

- A multiplayer session is successfully established between Player One and Player Two, allowing both players to interact within the game environment.

Normal flow

1. One of the players starts the game from the game menu.
2. Player One makes the first move.
3. Player Two makes the second move.
4. The players take turns playing the game until one player wins or they choose to terminate the game.

Alternative flows

<none>

Exceptions

If the keyboard does not respond or if input from one player is not recognized, the game provides feedback

REQ-I1.1 Switching to keyboard input and keyboard UI

Created by	Thom Kunkeler
Date created	2024-02-24
Primary actor	Active player
Secondary actor	<none>
Diagrams	Figure 5
User requirement	REQ-I1 2.2.3 Theme: Interaction (REQ-Ix) As someone who mostly uses a personal computer in my free time, I want to control the game using either a keyboard or a mouse, so that I can choose the input method I'm most comfortable with.

The system shall detect whether the player is using keyboard or mouse and present a user interface accordingly. The standard user interface shall be presented when the player is using a mouse, and an alternate user interface shall be presented when the player switches to keyboard input. The user interface for the keyboard includes arrows which helps the player navigate the board.

Trigger

The player provides a valid keyboard input to the system

Preconditions

- PRE-I 1.1-1 The system displays the user interface for mouse

Postconditions

- POST-I 1.1-1 The system displays the user interface for keyboard

Normal flow

REQ-I 1.1.a Switch user interface for keyboard input

1. The system displays the user interface for mouse
2. The player provides a valid keyboard input
3. The system recognizes the keyboard input and displays the user interface for keyboard

Alternative flows

<none>

Exceptions

REQ-I 1.1/EXCEPTION 1 Keyboard input not recognized

- The player is using the keyboard but not providing the correct input as defined in the instructions
- The system continues to display the user interface for mouse input

REQ-I 1.1/EXCEPTION 2 Keyboard not connected

- The player provides valid keyboard input
- The keyboard input is not recognized by the computer
- The system continues to display the user interface for mouse input
- The player connects the keyboard
- Restart the functional requirement

REQ-I3.1 Terminal game

Created by	Thom Kunkeler
Date created	2024-02-24
Primary actor	Player
Secondary actor	Terminal
Diagrams	Figure 4
User requirement	REQ-I3 As a player, I want to play in the terminal, so that I do not need to

install any game files [2.2.3 Theme: Interaction \(REQ-Ix\)](#)

A player shall be able to launch the game using the terminal by writing the command to execute the Python script in the terminal. The terminal shall execute the Python script and launch the game.

Trigger

The player commands the terminal to execute the provided Python script

Preconditions

- PRE-I 3.1-1 The player launches the terminal
- PRE-I 3.1-1 The player writes a command to execute the Python script in the terminal

Postconditions

- POST-I 3.1-1 The Python script is executed by the terminal and the game is launched

Normal flow

REQ-I 3.1.a Launch a new game from the terminal

- The player launches the terminal
- The player writes a command to execute the Python script in the terminal
- The Python script is executed by the terminal and the game is launched

Alternative flows

<none>

Exceptions

REQ-I 3.1/EXCEPTION 1 Python not installed

1. The player writes the Python command to execute the Python script in terminal
2. The terminal displays the error message: "Python is not recognized as an internal or external command, operable program, or batch file"

3. The player installs Python
4. Restart the functional requirement

REQ-I3 Layout of views

Created by Isak Hvarfner
Date created 2024-02-11
Primary actor Terminal
Secondary actor <none>
Diagrams [Figure 13](#), [Figure 15](#)
User requirement REQ-I3 [2.2.3 Theme: Interaction \(REQ-Ix\)](#)

It is important that the board stays centered because it is aesthetically pleasing and communicates its central role in the game, and especially that it does not render outside the window where it cannot be seen. To accommodate even exceptionally small windows, it should gracefully remove less important elements and sacrifice centering before that.

Trigger

A terminal resize event is received.

Preconditions

<none>

Postconditions

- The board grid is centered in the terminal.
- The box, board, hand and instructions are rendered.

Normal flow

1. The controller retrieves the width and height of the terminal window.
2. The controller uses the sizes of the views report to update their new positions.

Alternative flows

<none>

Exceptions

- The terminal is too small.
 1. Try to create a layout where the entire layout is centered instead of being based off of the board in the center.
 2. If that fails, try a layout with only the board.
 3. If that fails, display an error message.

REQ-R2 The transfer of stones within the board

Created by	Isak Hvarfner
Date created	2024-02-06
Primary actor	Active player
Secondary actor	Controller
Diagrams	Figure 12 , Figure 11
User requirement	REQ-R2 2.2.2 Theme: Rules (REQ-Rx)

The system should react to a pattern of interaction which leads to the transfer of stones from a single source stack to several destination stacks through a cell-based walk through the edges of cells. It can only go on as long as dropping is performed in each cell entered. During the process, the picked up stones are all displayed.

Trigger

A grid cell containing a stack of stones is interacted with, either through the left mouse button, or through a keyboard-controlled cursor for board cells.

Preconditions

- The full instructions are not opened fullscreen.

Postconditions

- Picked up stones have been placed on top of the grid cells the active player chose.
- It's the opponent's turn.

Normal flow

1. A stack of stones in the grid cell interacted with.
2. A means of selecting how many are picked up is used. A view separate from the board displays the picked up stones.
3. Another grid cell is interacted with and the bottom stone of the hand stack is then transferred to the top of the destination stack.
4. If the hand stack is not empty, the flow continues with step 3.
5. The opponent becomes the active player.
6. The now active player starts to make a valid move and all stones are unmarked as recently placed.

Alternative flows

<none>

Exceptions

- The source stack contains only one stone.
 1. A message explaining this is displayed on screen.
 2. When a successful action is taken the message disappears, if a different unsuccessful action is taken, the message is replaced.
- A destination stack has a standing stone on top of it.
 1. A message explaining this is displayed underneath the board and the relevant rule is highlighted in the instructions.
 2. When a successful action is taken the message disappears, if a different unsuccessful action is taken, the message is replaced

REQ-R3 Building a road with flat stones

Created by	Shiqi Shu
Date created	2024-02-06
Primary actor	System
Secondary actor	Active player
Diagrams	Figure 8
User requirement	REQ-R3 As a player, I want to build a road of my flat stones, visible from

above, connecting one edge to the one opposite it, so that I win immediately, unless it's a draw. [2.2.2 Theme: Rules \(REQ-Rx\)](#)

The System shall enable an active player to construct a road using their flat stones. This road must be clearly visible from an aerial perspective, spanning directly from one edge or corner of the game board to the opposite edge or corner. The completion of such a road by a player immediately results in a win for them, unless the game is declared a draw due to conditions specified in other requirements.

Trigger

The active player decides to build a road using their flat stones during their turn.

Preconditions

- PRE-REQ4.1 The game is in progress, and it is the active player's turn.
- PRE-REQ4.2 The active player has sufficient flat stones to attempt building a road.

Postconditions

- POST-REQ4.1 The active player has successfully placed a flat stone on the board in alignment with their intended road.
- POST-REQ4.2 The game checks if the road built by the active player extends from one edge or corner to the opposite edge or corner. If so, the player wins immediately, barring a draw condition.

Normal flow

1. PRE-REQ4.1 and PRE-REQ4.2 are met.
2. The active player selects a flat stone from their reserve.
3. The player places the stone on the game board aiming to build a road from one edge/corner to the opposite.
4. POST-REQ4.1 and POST-REQ4.2 conditions are checked and fulfilled if applicable.

Alternative flows

<none>

Exceptions

<none>

REQ-R4 Draw condition

Created by	Shiqi Shu
Date created	2024-02-06
Primary actor	System
Secondary actor	Active player
Diagrams	Figure 14
User requirement	REQ-R4 2.2.2 Theme: Rules (REQ-Rx)

The System shall declare a draw if both players achieve winning conditions simultaneously within the same turn. This mechanism is designed to minimize the impact of turn order on the game's outcome, ensuring fairness and recognizing simultaneous victories.

Trigger

Both players complete winning moves in the same turn.

Preconditions

- PRE-REQ5.1 Both players are positioned to win the game in their current turns.
- PRE-REQ5.2 One player completes a winning move.

Postconditions

- POST-REQ5.1 The system checks for a winning move by the second player within the same turn.
- POST-REQ5.2 If both players achieve the winning conditions simultaneously, the game is declared a draw.

Normal flow

1. Preconditions are met.
2. Player One completes a move that would normally result in a win.
3. The system pauses to allow Player Two to complete their turn.
4. If Player Two also completes a winning move, postconditions are checked and fulfilled.
5. The game is declared a draw if both players achieve victory in the same turn.

Alternative flows

<none>

Exceptions

<none>

REQ-R6 Opponent's color initial stone placement

Created by	Sabereh Hassanyazdi
Date created	2024-02-06
Primary actor	Player One
Secondary actor	Player Two
Diagrams	Figure 16
User requirement	REQ-R6 2.2.2 Theme: Rules (REQ-Rx)

The game requires that the first stone each player places on the board at the start of the game is of their opponent's color. This introduces an early strategic element to the game, as players must consider not only how to build their path to victory but also how to potentially disrupt their opponent's strategy from the very first move..

Preconditions

- A new game has started.
- Both players are aware of the rule requiring the first stone placed to be of the opponent's color.

Postconditions

- Each player has placed one stone of their opponent's color on the board.
- The game proceeds with players placing stones of their own color.

Normal flow

1. Player One places their first stone, which is of Player Two's color on the board.
2. Player Two places their first stone, which is of Player One's color, on the board.
3. Players continue the game, now placing stones of their own color, strategizing both offense and defense based on the initial placements.

Alternative flows

<none>

Exceptions

- If a player attempts to place a stone of their own color as their first move, the game prompts them to select a stone of the opponent's color instead.

REQ-R7 Surrender feature

Created by	David Carlsson
Date created	2024-02-09
Primary actor	Player
Secondary actor	Opponent
Diagrams	Figure 6.a , Figure 6.b
User requirement	REQ-R7 2.2.2 Theme: Rules (REQ-Rx)

The system should give the player an option to surrender the game. The system shall register a request to trigger REQ-R10 before the normal game flow ends.

Trigger

- A request to end the game with information about the winner of the game is received by the system.

Preconditions

- PRE-REQ 8-1 An active game must be in progress.
- PRE-REQ 8-2 A draw request has been made.
- PRE-REQ 8-3 The draw request has been denied.
- PRE-REQ 8-4 A request to forfeit the game has been input.

Postconditions

- POST-REQ 8-1 REQ-R10 is executed.
- POST-REQ 8-2 The game state transitions to “new game” if a request to restart the game has been received.
- POST-REQ 8-3 The game terminates if an exit command has been received.

Normal flow

1. The system prompts a confirmation that the game should end.
2. Confirmation to end the game is received.
3. Information about the result of the game is gathered.
4. REQ-R10 is executed with the information about the game result.

Alternative flows

- The system does not receive a confirmation before a preset timer ends.
 1. The game continues from its previous state.

Exceptions

REQ-R8/Exception 1 The game is neither restarted nor exited.

1. A timer starts.
2. The timer reaches its limit and the game automatically restarts.

REQ-R8 Draw feature

Created by	David Carlsson
Date created	2024-02-09
Primary actor	Active player
Secondary actor	Opponent
Diagrams	Figure 7.a , Figure 7.b
User requirement	REQ-R8 2.2.2 Theme: Rules (REQ-Rx)

The system should give the active player and the opponent an option to draw the game. The system shall register a request to trigger REQ-R10 before the normal game flow ends.

Trigger

A request to end the game with the result set to draw is received by the system.

Preconditions

- PRE-REQ 9-1 An active game must be in progress.
- PRE-REQ 9-2 The draw command has been received.

Postconditions

- POST-REQ 9-1 REQ-R10 is executed.
- POST-REQ 9-2 The game state transitions to "new game" if a new game command has been received.
- POST-REQ 9-3 The game terminates if an exit command has been received.

Normal flow

1. The system receives a command to draw the game.
2. REQ-R10 is executed.
3. The system receives a command to restart the game.
4. The game state is set to "new game".

Alternative flows

1. The system receives a command to draw the game.
2. REQ-R10 is executed.
3. The system receives a command to exit the game.
4. The game exits normally

Exceptions

REQ-R9/Exception 1 The game is neither restarted nor exited.

1. A timer starts.
2. The timer reaches its limit and the game automatically restarts.

REQ-R9: Flat win

Created by	Elsa Grönbeck
Date created	2024-02-11
Primary actor	Players
Secondary actor	System
Diagrams	Figure 9
User requirement	REQ-R9 2.2.2 Theme: Rules (REQ-Rx)

When the board is fully covered, meaning in all of the squares there is at least one stone placed, and neither player has built a path from one end of the board to the other instead of automatically assuming a tie the system shall count the number of same colored flat stones only including the stones that are top stones. In other words excluding all flat stones that have another stone, no matter if it is a flat or standing stone, on top of them from the count. If one player's color is the majority of the stones counted they win. After counting the stones the results will be promptly displayed on the terminal.

Trigger

The board is fully covered and both players have placed all of their stones on the board

Preconditions

- PRE-REQ 12-1 An active game must be in progress.
- PRE-REQ 12-2 The board is full covered
- PRE-REQ 12-3 Both players have placed all of their stones on the board
- PRE-REQ 12-4 Neither player has built a road of stones

Postconditions

- POST-REQ 12-1 The game ends with a win for the player with the highest number of flat top stones
- POST-REQ 12-2 The game ends with a draw if the players have equal numbers of stones

Normal flow

1. Player places stone and ends turn
2. The flat top stones on the board are counted
3. Each players number of top flat stones are displayed
4. The player with the highest amount of flat top stones win
5. The winner is displayed

6. The board and stones reset to the initial state, and the game restarts.

Exceptions

REQ-R12/Exception 1. The players have the same amount of flat top stones.

1. Player places stone and ends turn
2. The flat top stones on the board are counted
3. Each players number of top flat stones are displayed
4. The game results in a draw and not a flat win
5. The board and stones reset to the initial state, and the game restarts.

REQ-R10 Game Over Detection

Created by Efdal Erdem
Date created 2024-02-20
Primary actor System
Diagrams [figure 17](#)
User requirement REQ-R10 [2.2.2 Theme: Rules \(REQ-Rx\)](#)

The system shall perform a check to see if the player won or lost the game and accordingly output a prompt in the system's terminal. The check will determine if a win condition or a draw condition has been met during a game session. Based on the outcome, the system shall appropriately conclude the game session and update the game state.

Trigger

The game is continuously monitored for win or draw conditions.

Preconditions

- PRE-REQ 10-1: An active game session is in progress.

Postconditions

- POST-REQ 10-1: The game session concludes, and the appropriate outcome (win or draw) is determined.

Normal flow

- The system continuously monitors the game state for win or draw conditions.
- If a win or draw condition is detected, the system concludes the game session.

Exceptions

- REQ-R10/Exception 1: If the system fails to detect the win or draw condition, implement additional checks or refine the detection algorithm to ensure accurate detection of game outcomes.

REQ-R11: Valid move validation

Created by	Efdal Erdem
Date created	2024-02-20
Primary actor	Player
Secondary actor	<none>
Diagrams	Figure 18
User requirement	REQ-R11

The game must enforce the rule that standing and flat stones are two separate things. Attempting to convert flat pieces to standing pieces or convert standing pieces to flat pieces are forbidden. The game shall not permit the player to perform this action.

Trigger

The player selects a stone on the game board to interact with.

Preconditions

- An active game session must be ongoing.
- It must be the player's turn to make a move.

Normal flow

1. The player selects a stone on the game board to interact with.
2. The game verifies if the selected stone is in a valid state to perform the desired action.
3. If the action is valid (e.g., not attempting to move a standing stone or not attempting to convert a flat stone), the game proceeds with the action.
4. If the action is invalid (e.g., attempting to convert a standing stone to flat or vice versa), the game interface displays an error message notifying the player of the forbidden action, and prevents the action from being executed.
5. The player rectifies the action by selecting a valid stone or performing a valid action.
6. The game verifies the corrected action.
7. If the corrected action is valid, the game proceeds with the action.
8. The game continues with the next player's turn.

Exceptions

- If a player attempts to convert a standing stone to flat or vice versa, the game will prevent the player from executing the forbidden action.

Postconditions

- The player successfully performs valid actions without attempting to convert between standing and flat stones.
- The game interface updates to reflect the new game state after each valid action.

REQ-A1: Accessing instructions during the game

Created by	Elsa Grönbeck
Date created	2024-02-11
Primary actor	Players
Secondary actor	System
Diagrams	Figure 10
User requirement	REQ-A1 2.2.4 Theme: Accessibility/Intelligibility (REQ-Ax) REQ-R11 2.2.2 Theme: Rules (REQ-Rx) (part of)

The game shall always show a short version of the game rules, a summary of the most essential parts of the game instruction and a player should always be able to access the full length of the instructions when they want. The short version of the game rules shall be displayed in a sidebar next to the game board. When a player wants to access the full instructions the instructions will promptly be presented within the terminal.

Trigger

A player wants to read the instructions

Preconditions

- PRE-REQ 13-1 An active game must be in progress.

Postconditions

- POST-REQ 13-1 The game continues when the player is done reading the instructions

Normal flow

1. A player wants to read the instructions for the game
2. The player can read the short summary of the instructions displayed in the sidebar next to the game board
3. When the player has reviewed the instructions they can continue playing the game

Alternative flows

1. The player is not satisfied with the summary of the instructions and wants to review the full instructions
2. The player clicks the button "Full instructions" on the bottom of the sidebar
3. The full instructions will be displayed in the terminal
4. The player reviews the instructions
5. The player uses the keyboard/clicks to exit the full instructions

6. The game resumes when the player makes their move

Exceptions

REQ-R12/Exception 1. The screen is too small to fit the sidebar

1. If the screen is too small to fit the sidebar with a short summary of the instructions only the button "Full instructions" will be available on the screen to click.
2. When the button is clicked it the full instructions will pop up for the player to review.
3. The full instructions will be displayed in the terminal.
4. The player reviews the instructions.
5. The player uses the keyboard/clicks to exit the full instructions.
6. The game resumes when the player makes their move.

2.6 Non-functional requirements

The **non-functional requirements** section of the requirements will briefly describe the limitations of the solution, imposed by some externality. Unlike the system constraints, these **are enforced and must be followed**, and most of them are aligned with our architecture.

Performance.....	38
2.6.1 Input switching speed.....	38
2.6.2 User Interface responsiveness.....	38
2.6.3 Optimized Response Time.....	38
Usability.....	38
2.6.4 Ease of use.....	38
2.6.5 Language option.....	38
2.6.6 Visual Distinction.....	38
2.6.7 Simple documentation.....	39
2.6.8 Comprehensive Documentation.....	39
2.6.9 Wrong move requirements.....	39
2.6.10 Always accessible game rules.....	39
Dependability.....	39
2.6.11 Reliability.....	39
2.6.12 Availability.....	39
2.6.13 Future Python compatibility.....	39
Environment.....	40
2.6.14 Supported platform.....	40
2.6.15 Dependencies.....	40
Security.....	40
2.6.16 Security and data protection.....	40
Operational.....	40
2.6.17 Scalability.....	40
2.6.18 Maintainability.....	41
2.6.19 Performance Monitoring and Optimisation.....	41
Legislative.....	41
2.6.20 Compliance.....	41
Development.....	41
2.6.21 Version Control.....	41
2.6.22 Software Engineering Methodology.....	42

2.6.23 Developers.....	42
2.6.24 Python Features and Minimum Version.....	42

Performance

2.6.1 Input switching speed

The system shall be able to seamlessly and rapidly transition between keyboard and mouse input, with the system being able to switch between these input methods at a speed of no more than 0.5 seconds.

2.6.2 User Interface responsiveness

The game's user interface shall demonstrate high responsiveness, ensuring that all interactions, such as user input and UI update, occur within 0.2 seconds.

2.6.3 Optimized Response Time

The game maintains a response time of less than 2 seconds for all player actions during the session, ensuring a seamless and engaging experience. This includes the processing and display of moves, score updates, and game state changes to both players without noticeable delays and crashes.

Usability

2.6.4 Ease of use

The game shall be intuitive and the user shall be able to learn all of the game features after using the game for 1 hour.

After one hour of game time, the user shall be able to input a correct command at least 95% of the time.

2.6.5 Language option

The game should be available only in English to streamline accessibility and ensure uniformity across all users. Given the widespread use and understanding of English globally, opting for a single language reduces complexity and ensures seamless gameplay for all players. Measurably, players should be able to navigate through key game menus and understand basic gameplay mechanics within a maximum of 5 minutes.

2.6.6 Visual Distinction

The game must visually differentiate standing and flat stones to ensure players can easily identify the state of each stone at a glance.

2.6.7 Simple documentation

The documentation shall be thorough and contain all of the game rules in natural language. The number of unintentional illegal moves after reading the game rules once shall be no more than 10% on average.

2.6.8 Comprehensive Documentation

The game needs to include thorough documentation as an accessible in-game help feature. This thorough documentation should cover game rules and instructions. All documentation in the game's interface is in simple and basic level english.

2.6.9 Wrong move requirements

A clear message in natural language shall be displayed if a player attempts to make an illegal move. The message shall contain enough information so that an English speaker shall understand why the move was incorrect at least 99% of the time.

2.6.10 Always accessible game rules

The game should ensure that players can always access the game rules on their screens. This can be achieved by continuously displaying the rules in the user interface or making them easily accessible in a prominent location. Continuous visibility of the game rules allows players to consult and understand them at any time while playing, thus making the game experience more consistent and enjoyable.

Dependability

2.6.11 Reliability

The game shall not crash more than 1% of the time. That is, for every 100 games played, at least 99 should have terminated correctly.

2.6.12 Availability

The game should be accessible to its intended end users at all times. In other words the game needs to be available for download at all times. An acceptable downtime for the game download is nearly nine hours per year which will equate to approximately 99.9% uptime. This can be measured by service availability which can be calculated by dividing the systems uptime with the total sum of the systems uptime and downtime.

2.6.13 Future Python compatibility

The game should remain compatible with future updates of Python to ensure its continued functionality. To measure this, the development team should conduct regular

assessments to verify compatibility with the latest versions of Python. This can be achieved by implementing automated tests that evaluate the game's performance and behavior with each new Python update. Additionally, tracking the time taken to adapt the game to new Python versions can serve as a metric to ensure timely updates and compliance with this requirement.

Environment

2.6.14 Supported platform

The game should be compatible with at least the Windows, macOS, and Linux operating systems. This will increase accessibility by catering to a wider range of players.

2.6.15 Dependencies

The game should not require any dependencies on macOS and Linux and only require the package windows-curses on Windows.

Security

2.6.16 Security and data protection

The game shall implement robust security measures to protect user data and prevent unauthorized access. This includes encrypting player data, especially if the game involves online multiplayer interactions or stores user preferences and progress. Regular security audits and updates are required to address new vulnerabilities and ensure compliance with relevant data protection regulations.

Operational

2.6.17 Scalability

The game shall be designed to efficiently handle an increase in user load, especially for features that may involve multiplayer or online components. The backend infrastructure should support scaling to accommodate a growing number of simultaneous players without degradation in performance or user experience. This includes optimizing server resources, ensuring efficient data handling, and minimizing latency in multiplayer sessions. For example: the system is capable of supporting at least a 50 per cent increase in the number of simultaneous players over the current peak user level without increasing response time.

2.6.18 Maintainability

The game should be easy to maintain with a low-cost during its expected lifetime. This for example can be done by having decoupled code. One way to describe this is that components and modules are less interdependent on each other. Basically if a component needs fixing or changing the fewer lines of code that needs to be fixed the better. Measuring true maintainability is hard to do before you actually have to maintain the game. One way to approach maintainability in the planning stage is to evaluate which parts or components build the system making sure that a change in a component doesn't trickle into a need to change the whole system. There are many ways to measure the maintainability of a system. One way is to try to assess the effort needed to maintain the system. This can be calculated by measuring the time it takes to fix a piece of code (mean time to repair).

2.6.19 Performance Monitoring and Optimisation

The game should implement a comprehensive performance monitoring system for real-time tracking and recording of the game's performance metrics, such as load time, frame rate and memory usage. The system should be able to automatically detect performance bottlenecks and anomalies and provide relevant data to support performance optimisation decisions. Performance monitoring data should be reviewed on a regular basis (e.g., monthly) to ensure that the game continues to meet established performance standards. For example: the game takes no more than 5 seconds to load.

Legislative

2.6.20 Compliance

The game needs to follow the legal constraints of the countries the game will be available in. For example making sure the game does not store any personal information in such a way that it breaks the GDPR laws. If any future features need storage of personal information then this needs to be clearly communicated to the user.

Development

2.6.21 Version Control

The game will be developed using version control provided by Git and hosted on GitHub, so that changes can be rolled back easily and who has changed what and when can be tracked for each piece of code.

2.6.22 Software Engineering Methodology

Scrum and agile methodologies will need to be applied during the development of the game due to the format of the course. Tasks will for the same reason need to be tracked on a Trello Kanban board and assigned points on a relative scale of effort according to guidelines available to read as a card in the Kanban board.

2.6.23 Developers

The game will need to be developed by a team of at most 7 people with programming experience, including one with professional embedded experience. They will be working less than one-third of full-time due to the pace of the course and time spent on lectures and deliverable assignments.

2.6.24 Python Features and Minimum Version

The match statement needs to be available for use because of developer preference and as such the game needs to be developed for Python 3.10 and later.

3 Risk Assessment Plan

In this risk management section, we aim to identify and address the most critical risks that could potentially jeopardize the successful completion of our project. After a comprehensive analysis of various project management, technical, operational, and human factors risks, we have identified the top four risks that pose the greatest threat to our project's success.

Failure to Meet Project Timelines: One of the most critical risks is the possibility of failing to complete the project on time due to poor planning or unforeseen complications. This risk could impact project approval and future support, leading to significant setbacks in project delivery.

Data Loss or Breach: Another critical risk is the potential loss or breach of important project data. Such an event could undermine project reliability and erode customer trust, posing a severe threat to project success.

Software Compatibility Issues: Incompatibilities or integration issues with the technologies used in the project represent a significant risk. Failure to meet performance expectations or requirements could lead to dissatisfaction among users, impacting project outcomes.

Operational Disruptions: Operational hazards such as failures in underlying hardware or wasteful processes during software integration could disrupt project development and progress, potentially derailing the project if not adequately addressed.

To minimize the probability of these critical risks and ensure project success, we have devised proactive mitigation strategies:

Rigorous Planning and Contingency Measures: Implementing rigorous planning sessions, including contingency planning and regular project reviews, to address potential delays and scope creep.

Data Security Measures: Strengthening cybersecurity measures and conducting regular security audits to prevent data breaches and protect sensitive project information.

Continuous Monitoring and Dependency Management: Monitoring dependencies closely and having contingency plans in place to address any

issues that arise, mitigating risks related to software compatibility and technical dependencies.

Operational Resilience: Implementing redundancy measures for critical hardware components and establishing backup plans for hardware failures to minimize operational disruptions.

By identifying these critical risks and implementing proactive mitigation strategies, we aim to safeguard the successful completion of our project. Through careful planning, continuous monitoring, and proactive risk management, we are confident in our ability to overcome challenges and deliver a high-quality product that meets clients' expectations.

3.1 List of potential risks

Project Management Hazards

1. Failing to meet the expectations or requirements of stakeholders will impact project approval and future support.
2. Lack of proper documentation leads to confusion and inefficiencies in project execution and handover.
3. Failure to complete the project on time due to poor planning or unforeseen complications.
4. Inadequate project planning makes late adjustments incredibly difficult.

Technical Hazards

1. Losing important project data due to a lack of backups or technical failures.
2. Software is not meeting performance expectations or requirements and affecting user satisfaction.
3. New language versions may lead to broken code or functionality in the future.
4. Introducing a new language version could inadvertently introduce new bugs or performance issues that were not present in the older version.
5. Libraries, frameworks, or other dependencies used in your project may not yet support the newest version of the programming language, causing parts of your application to fail.

Operational Hazards

1. Wasteful or redundant processes as a result of integrating the software of another team.

2. Failures in the underlying hardware supporting the development process.
3. Cybersecurity threats expanded analysis and planning.

Human Factors

1. Disagreements or conflicts within the team lead to a toxic work environment and reduced productivity.
2. Team culture and management clashes when working with another team for software integration.
3. Poor communication leads to misunderstandings about project requirements.
4. Insufficient skills or knowledge among team members to complete project tasks effectively.
5. High levels of stress or overwork lead to reduced productivity for team members.
6. The key person decided that the project could not be carried out properly.
7. Difficulty in finding suitable time slots for collaboration with the other team during integration could disrupt cooperation and integration processes.
8. The other team not completing their work within the specified timeframe could hinder the progress of the project and meet the deadlines.

Project and Product Risks

1. Specification delays, the interface specification of the AI module to be integrated are not available on schedule.
2. Requirement change, the client might introduce new changes not accounted for in previous planning.
3. User engagement lower than expected.
4. Run out of budget or time limitation.
5. Deficiencies or inadequacies in the project's content could result in the final product failing to meet expected performance or user satisfaction.
6. Overreliance on external resources in the project increases the risk of uncontrolled changes from external factors adversely affecting the project.
7. Unexpected changes in requirements from clients could complicate.
8. Loss of key developer personnel.

3.2 Risk analysis and Planning

Project Management Hazards

- **Failure to complete the project on time due to poor planning or unforeseen complications.**

At the moment of writing, the team is on schedule to deliver the product and corresponding deliverables. In the case of unforeseen circumstances, the team is ready to deliver a base product, and there is a possibility not to deliver D6. As for the integration component of sprint 4, the team shall try to collaborate with another team. If, for some reason, this does not work as planned, the team will develop the component themselves. In the rare case that one of the team members cannot continue their work, the collaboration is set up in such a way that other team members can still deliver the product and the deliverables. Therefore, while this hazard has severe consequences, namely failure, the probability of the consequences occurring are low.

- **Specification delays, the interface specification of the AI module to be integrated are not available on schedule.**

Late delivery of the interface specification to integrate the modules used by the project can cause blockage of implementation. This can also lead to tighter time constraints regarding interface verification. The probability of a late interface specification is moderate, due to the high workload of the other teams. The effects of a late delivery are tolerable.

- **Requirement change, the client might introduce new changes not accounted for in previous planning.**

The risk of requirement creep is to be considered low at this stage of development. Effects on the project depend on what kind of changes are required but range from insignificant to tolerable.

- **Inadequate project planning makes late adjustments incredibly difficult.**

Poor planning may lead to unrealistic timelines and scope creep, affecting project delivery.

Implement rigorous planning sessions, including contingency planning and regular project reviews.

Technical Hazards

- **Introducing a new language version could introduce new bugs or performance issues that were not present in the older version.**

Introducing a newer version of Python may become an obstacle to the UU game or might even introduce new bugs or performance issues. These could also compromise the game's quality and the user experience.

To mitigate this potential problem, it is important to have continuous performance monitoring and bug tracking after updates or integrations which would headress any emerging issues.

- **New language versions may, leading to broken code or functionality in the future.**

The development of our terminal UU game involves leveraging the latest version of Python (right?), which introduces a medium risk of facing non-backward compatible changes. This risk could result in broken code, functionalities and the integration between the AI components.

To mitigate this, we should stay informed with potential language updates, and include a comprehensive test suite to quickly identify and address any issues.

- **Software not meeting performance expectations or requirements and affecting user satisfaction.**

The risk of the software not meeting performance expectations or requirements is crucial and might have the potential to negatively affect user satisfaction. The game performance must ensure smooth sessions and responsive interactions which is a critical factor for engaging players. Performance issues could lead to frustration, reduced interest among players, and ultimately, affect the success of the game.

To mitigate this problem, it is important to prioritize performance testing throughout the development process and adopt a test-driven approach.

- **Losing (an important) parts of the code due to lack of backups, technical failures.**

The risk of losing important project data due to lack of backups or technical failures is highly probable and carries severe consequences within the UU game, especially since it is a collaborative project with multiple people working on different parts of the project. This would also lead to project delays and the

potential need to redo some portions of work. And this doesn't only apply to the code but also to the documentation, etc.

To effectively mitigate this risk, a key strategy involves implementing regular saving practices within GitHub.

Operational Hazards

- **Cybersecurity threats expanded analysis and planning.**
Data breaches can compromise user data and trust.
Strengthen cybersecurity measures and conduct regular security audits.
- **Wasteful or redundant processes as a result of integrating the software of another team.**
All individual team members will read the D3 of other teams, and decide accordingly if it is worth the time and effort to integrate a component from another team. In the case of collaboration with another team, the product owners of either team shall decide on the integration and division of labor. Thus, while there is a risk of wasteful and redundant processes, this risk is mitigated by careful preparation and decision making leading up to the integration. The severity of the consequences of this risk are medium, as at most, the team will have to work more on the wasteful and redundant processes.

Human Factors

- **Difficulty in finding suitable time slots for collaboration with the other team during integration could disrupt cooperation and integration processes.**
Limited availability for collaboration sessions may lead to delays in resolving integration issues and hinder the smooth coordination between teams. This can result in miscommunications, misunderstandings, and inefficiencies in the integration process.
- **The other team not completing their work within the specified timeframe could hinder the progress of the project and meet the deadlines.**
Risk Analysis: If the other team fails to deliver their work on time, it may lead to dependencies not being met, causing delays in subsequent tasks and jeopardizing the overall project timeline. This could result in increased pressure on resources and compromise the quality of deliverables.
- **Key person departure that the project could not be carried out properly.**

Loss of key team members can derail project progress.

Develop a cross-training program to mitigate the impact of personnel changes.

- **Disagreements or conflicts within the team, leading to a toxic work environment and reduced productivity.**

The team filled in a pre-course questionnaire which assessed their compatibility to work with one another. During the first sprints, it became evident that the team has a positive work culture and high productivity. The risk of disagreement or conflict is therefore considered low based on previous performance. The severity of the consequences are also low, as the scrum master is responsible for solving conflicts between team members before they get worse.

- **Team culture and management clashes when working with another team for software integration.**

Once the team has decided that they would like to work with another team, the team will have to assess the group agreement of the other team, and assess their work culture in a first meeting. In doing so, the risk of team culture and management clashes is mitigated. In the case that the teams cannot continue their collaboration, the teams can develop the software components themselves as a plan B. The consequences of the worst case would be that the team has to work more than initially planned.

Project and Product Risks

- **Loss of key developer personnel.**

Losing key developers as the project nears completion will most likely contribute to major delays in the release. The probability of losing a key developer this close to release is low but the effects would be serious.

- **Deficiencies or inadequacies in the project's content could result in the final product failing to meet expected performance or user satisfaction.**

Insufficient project content may lead to gaps in functionality, usability issues, or deviations from stakeholder expectations. This could result in dissatisfaction among users, lower adoption rates, and potential reputational damage to the project.

- **Overreliance on external resources in the project increases the risk of uncontrolled changes from external factors adversely affecting the project.**

Dependence on external resources may expose the project to risks such as delays, cost overruns, or changes in external factors beyond the team's control. This can lead to disruptions in project planning and execution, impacting its success and stability.

- **Unexpected changes in requirements from the client could complicate.**

Sudden changes in client requirements may lead to scope creep, resource reallocation, and potential conflicts with existing project plans. This could result in increased project complexity, extended timelines, and challenges in managing stakeholder expectations.

- **User engagement lower than expected.**

The game might not meet user expectations, leading to low engagement.

Engage with the community early on for feedback and adjust development accordingly.

- **Run out of budget or time limitation.**

Unforeseen expenses can push the project beyond its budget.

Maintain a reserve budget and monitor expenses closely.

Risk assessment table

Risk	Prob.	Effects	Strategy/Mitigation
Failure to complete project on time due to poor planning or unforeseen complications.	Low	Failure	The team is to focus on a base product and report in the first instance, even with the risk of achieving a lower grade. D6 (for grade 5) is only prioritized when all the other deliverables have been successfully handed in.
Wasteful or redundant processes as a result of integrating the software of another team.	Low	Extra work	All team members to read D3 of other team and decide on team to work with The product owner is to decide

			how many processes are to be added when working with another team
Disagreements or conflicts within the team, leading to a toxic work environment and reduced productivity.	Low	Conflict resolution by scrum master	<p>Based on pre-course questionnaire and collaboration in first sprints, the team has positive work culture and high productivity</p> <p>In the case of conflict, the scrum master is responsible for conflict resolution between the team members</p>
Team culture and management clashes when working with another team for software integration.	Low	Extra work	<p>The team assesses the group agreement of the other team, and their work culture during a first meeting</p> <p>Plan B is to withdraw from the collaboration and to develop the software independently.</p>
Losing (an important) parts of the code due to lack of backups, technical failures.	Low	Extra Work	<p>The risk of losing important project data due to lack of backups or technical failures is highly probable and carries severe consequences within the UU game, especially since it is a collaborative project with multiple people working on different parts of the project. This would also lead to project delays and the potential need to redo some portions of work.</p>

			<p>And this doesn't only apply to the code but also to the documentation, etc. To effectively mitigate this risk, a key strategy involves implementing regular saving practices within GitHub</p>
<p>Software not meeting performance expectations or requirements and affecting user satisfaction.</p>	Medium	Failure	<p>The risk of the software not meeting performance expectations or requirements is crucial and might have the potential to negatively affect user satisfaction. The game performance must ensure smooth sessions and responsive interactions which is a critical factor for engaging players. Performance issues could lead to frustration, reduced interest among players, and ultimately, affect the success of the game.. To mitigate this problem, it is important to prioritize performance testing throughout the development process and adopt a test-driven approach.</p>
<p>New language versions may, leading to broken code or functionality in the future.</p>	Low	Extra work	<p>The development of our terminal UU game involves leveraging the latest version of Python (right?), which introduces a medium risk of facing non-backward compatible changes. This risk could result in broken code, functionalities and the integration between the AI</p>

			<p>components.</p> <p>To mitigate this, we should stay informed with potential language updates, and include a comprehensive test suite to quickly identify and address any issues.</p>
Introducing a new language version could introduce new bugs or performance issues that were not present in the older version.	Low	Extra Work	<p>Introducing a newer version of Python may become an obstacle to the UU game or might even introduce new bugs or performance issues. These could also compromise the game's quality and the user experience.</p> <p>To mitigate this potential problem, it is important to have continuous performance monitoring and bug tracking after updates or integrations which would headress any emerging issues.</p>
Difficulty in finding suitable time slots for collaboration with the other team during integration could disrupt cooperation and integration processes.	Medium	High	Schedule regular meetings with the other team to discuss integration plans and prioritize collaborative efforts.
The other team not completing their work within the specified timeframe could hinder the progress of the project and meet the deadlines.	Medium	High	Establish clear milestones and deadlines for the other team, and regularly monitor their progress to identify any delays early on.
Deficiencies or inadequacies in the project's content could result in the final product failing to meet expected performance or user satisfaction.	High	Very High	Conduct thorough reviews and testing throughout the project lifecycle to identify and address any deficiencies or inadequacies.

Overreliance on external resources in the project increases the risk of uncontrolled changes from external factors adversely affecting the project.	High	High	Diversify resources where possible and have contingency plans in place to mitigate the impact of external factors on the project.
Unexpected changes in requirements from clients could complicate.	Medium	Very High	Establish a robust change management process to evaluate and incorporate client requirements efficiently while minimizing disruptions.
Specification delays, the interface specification of the AI module to be integrated are not available on schedule.	Medium	Tolerable	Contact the developers of the other team to get a status update about their implementation and specification of the interface to the AI module.
Requirement change, the client might introduce new changes not accounted for in previous planning.	Low	Insignificant - Tolerable	Requirement changes to the product are halted until coming development iterations.
Loss of key developer personnel.	Low	Serious	Utilize pair programming to encourage different developers to have a broader understanding of the system. Increase overlap of work to allow people to increase understanding of the different tasks.
Inadequate project planning makes late adjustments incredibly difficult.	High	Extra Work	Implement rigorous planning sessions, including contingency planning and regular project reviews.
Cybersecurity threats expanded analysis and planning.	Medium	Serious	Strengthen cybersecurity measures and conduct regular security audits.
Key person departure that the project could not be carried out properly.	Medium	Extra Work	Develop a cross-training program to mitigate the impact

			of personnel changes.
User engagement lower than expected.	Low	Serious	Engage with the community early on for feedback and adjust development accordingly.
Run out of budget or time limitation.	Medium	Extra Work	Maintain a reserve budget and monitor expenses closely.

4 Integration Plan and Implementation

This integration plan outlines the collaborative efforts between Group A and Group C for the integration of the UU Game Platform with the UU Game AI. Our teams embark on a journey to merge our respective components into a cohesive and functional whole.

At the core of this endeavor lies the vision of creating a seamless experience for players, where the dynamic capabilities of the UU Game Platform converge harmoniously with the strategic prowess of the UU Game AI. Through careful planning and meticulous execution, we aim to deliver an integrated software solution that not only meets the diverse needs of our clients but also exceeds their expectations.

Drawing inspiration from the Reuse-oriented Software Engineering model, we adopt a structured approach towards integration, delineating clear tasks and milestones to guide our progress. Central to our strategy is the establishment of open communication channels between our teams, ensuring transparency and facilitating the exchange of knowledge and expertise.

As we navigate through the integration process, we remain cognizant of the unique requirements and specifications outlined by both groups. Through collaborative refinement and adaptation, we aim to seamlessly incorporate these elements into our collective framework, thereby enhancing the overall functionality and user experience of the integrated system.

4.1 Integration Plan

We are using a simplified version of the Reuse-oriented Software engineering model (p.52).

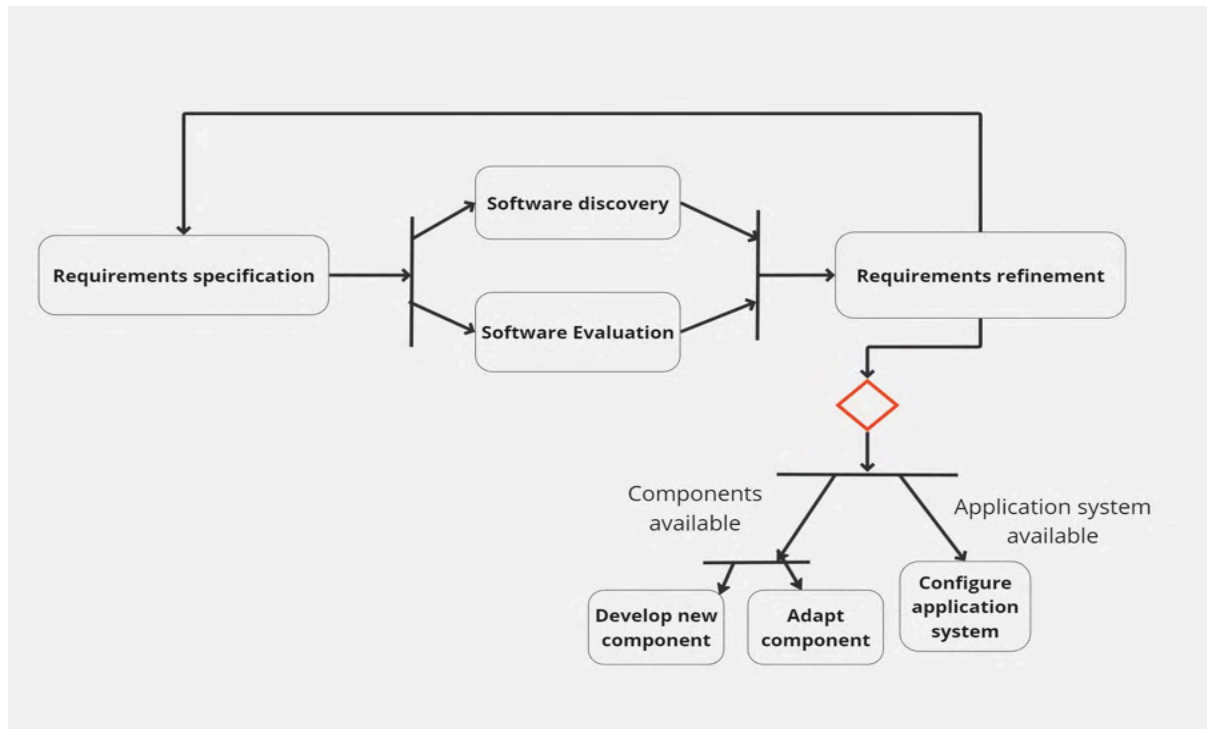


Fig. 19: Reuse-oriented Software engineering model

Our game platform component will be provided to Group C as a standalone component compatible with their AI. According to the [group agreement](#), both groups grant each other irrevocable permission to deal with the current version of their software and any related documentation or files without restriction. Both groups agree to maintain an open line of communication throughout the integration process. Group A agrees to make some effort to support and answer at least some questions from Group C regarding their software for the purpose of facilitating the successful integration of the platform and the AI. Group C agrees to make a reasonable effort to answer any question regarding their project.

We expect to deliver the integrated software within 7 days, leaving the remaining days for other tasks. Members of the group who are not working on integration are scheduled to spend more time writing the DF.

4.2 Activity tasks

Integration tasks

The main integration tasks are:

Software discovery

I-SD1. After reading the D3 we will, as a team, discuss the project specifications of the other teams to find which teams are worth making further agreements with.

I-SD2 Our project manager will write and approve the agreement with Group C (see [Appendix](#)).

Software evaluation

I-SE1 Our software engineers will evaluate the code of the other teams to determine the amount of effort that would be required to make them compatible.

Requirements refinement

I-RF1. As a team, we will refine the requirements using information about the reusable components and applications that have been discovered (see [Appendix](#)).

Adapt components

I-AC1 Our software engineers will identify the relevant code for the new user requirements.

I-AC2 Our software engineers will adapt the code from the relevant user requirements so that it can be used within our system.

Integrate system

I-I1 Our software engineers will integrate the adapted component to our system.

Concurrent tasks

The main other tasks for the sprint are:

O1 Write D5 deliverable in order to plan better how to use our resources.

O2 Copy-edit the D5 and submit.

O3 Writing the DF.

O4 Copy-edit the DF and submit.

O5 Review and integrate feedback D4.

O6 Copy-edit D4 and submit.

Tabel 1 Activity Table

Task	Effort (%)	Duration (days)	Milestone	Dependencies
I-SD1	25%	1		
I-SD2	50%	1	M1	
I-SE1	25%	1		
I-RF1	25%	1	M2	I-SD2
I-AC1	25%	1		
I-AC2	100%	2		I-AC2
I-I1	100%	2	M3	
O1	150%	2	M4	
O2	25%	1		O1
O3	600%	5	M5	
O4	50%	1		O3
O5	100%	1	M6	
O6	25%	1		O5

M3 = Complete source file for UU Game

M4 = Integration plan

M5 = DF

Table 2 Activity bar chart

Day	1	2	3	4	5	6	7	8	9	10	11
Thom	I-SD1 25%		O1 / 100%	I-RF1 25%				O3 50%	O3 50%	O5 25%	
Isak	I-SD1 25%	I-SD2 50%	O1 / 25%	I-SE1 25%	I-AC1 25%	I-AC2 100%	I-I1 100%				
Efdal	I-SD1 25%		O1 / 50%					O3 50%	O3 50%	O5 25%	
Elsa	I-SD1 25%							O3 50%	O3 50%		
David	I-SD1 25%							O3 50%	O3 50%	O5 25%	
Shiqi	I-SD1 25%		O2 / 25%					O3 50%	O3 50%	O6 25%	O4 50%
Sabereh	I-SD1 25%							O3 50%	O3 50%	O5 25%	

Justification

The main goal of this project was to bring the two groups together and create a single integrated work. Since the other group's clients had different needs, we chose to contact both of the other two groups to find the product that would be most compatible with

our product. After reviewing both groups' code, we concluded that Group B was the best choice for us. However, before we managed to write the contract with Group B, they reconsidered their decision. Consequently, we shifted our focus and finalized a contract with Group C, which showed a strong interest in our product. After a couple of days, this change in direction necessitated a strategic adjustment, and we managed to secure another agreement with Group B, where we only use their product, but they don't use ours.

Looking back at our journey, it would have been easier if we had been able to make just one integrated piece of work. If we had combined our efforts into one integrated work, the process would have been more streamlined and possibly quicker, eliminating duplicate work and effectively achieving our project goals.

Implementation of the Integration

In this section we describe some of the problems we encountered when integrating the AI with our game and how we tried to solve them.

To implement selecting difficulty, command line arguments are used. At one point there was a plan to add a whole menu, but we figured someone who wants to play a terminal game might as well learn how command line arguments work.

To get a working prototype, all that was needed was translating back and forth between our data structures, and manually counting the stones to see which one had been placed and which ones were recent. The only change other than that was disabling all the print statements and changing the board size to be generic so it could match our board.

Making the board size generic unfortunately was more work than originally anticipated, as unlike our group, the board size was defined in many places, and sometimes not even as constants but as the amount of elements in a list, making it harder to do a find and replace.

If this was a real project with plans of launching, we might have considered a complete rewrite with performance in mind, as with only a search depth of two, the game was struggling. One factor, but not the biggest, is that many functions repeatedly querying the number of pieces on the board, when a separate counter, only incremented/decremented as stones are placed, as well as not using nested (linked) lists. Having received something in exchange for sharing our code with the other groups,

access to their versions, would still have allowed us to see potential problems with designs if we would have redesigned it instead.

Our game has more possible moves than what the AI was prepared for because their rules for moves were more restrictive. This means that in the end we had to remove some rules like time limits for the AI because it needed more time to solve the next move when applied to our game version.

When it came to scoring, we did not have time to experiment with how much value to give each type of piece, as a difference between our games was that our stone poses were not interchangeable before being placed.

Honoring agreements

We were asked questions by? abou group C and answered.

6 Team reflection / Lessons learned

After individually reflecting how we worked as a group and lessons learned during the project. We have made a summary of the team's thoughts and lessons learned. Overall we are satisfied with how we worked as a group and our teamwork. We believe we worked well together and all of us contributed to the work. Our teamwork was based on having a good tone when communicating with each other. All of the team members were very flexible, did their best and worked together, even when things did not go to plan. We made most of our decisions during team meetings. Although we are content with our work as a team there are always things that can be improved. Below we will reflect on things we would have done differently if we could redo the project.

Even though we often kept a good tone and saw each other as nice people, and we were mostly able to work together and allocate work, there were times where we could have been more constructive. For example, while we were discussing the complete design of the game beyond the prototype, David suggested that we should have a Player class. Isak asked why and David responded that not having a player class would mean there is no organization, that functions would be spread all over. In Isak's mental image that was still developing, objects had only top-down references and the public API, just like the UI, always assumed the active player was the one taking an action, which meant that having behavior associated with player data did not make sense. But instead of clearly expressing this, he made a quick remark that organization can also be done as modules or structs, and made what came off as a snide remark about not assuming the design has to be object oriented. A reason he did not take the time to explain was that the suggestion, to Isak, came off as only a distracting low-effort idea, because he thinks grouping things together is a fundamental part of the coding activity that need not be said, when there are particulars of the design that need to be worked out. This could have contributed to resentment, and this was not brought up until this reflection.

In a professional, full-time environment, as opposed to students with busy schedules assigned to each other, people that do the designing will probably be more on the same page, and there can be time to fully explore disagreements. But when scheduling meetings is so hard, this may have to be postponed to a later time. But bringing it up later can make it even harder to overcome any aversion to expressing one's feelings and the idea that it is a waste of time or something that resolves on its own. Here, gender societal expectations can play a role.

Overall, it can be said that communication is always a challenge, but that moving in the direction of more communication and not less is probably the safest bet. In addition,

communicating effectively and with empathy, and speaking up where that falls short can be another helpful principle.

To expand further on part of the context the project was taking place in – the course – we look to the fact that we were still learning what we were doing and the theory. Due to assurances that we could submit and receive feedback multiple times, and our busy schedules with other courses, it was natural that we did not not all study the material at the pace and in a way where we were always ready to contribute equally in each meeting. Perhaps here we should have done some more extensive planning in addition to mostly individually adding to the agenda for each weekly meeting a couple of days beforehand. One example of a meeting where this came into play was our knowledge of scrum. In one of our first meetings we assigned roles like scrum master and such. In the next meeting, as we were expected to individually write the tasks in Trello, we discovered we knew different amounts on how scrum works. This is an example of a problem that could have been solved by more extensive planning but at the time we felt like we had no time to make a more extensive and thorough plan of how to work together.

Then there was the uneven distribution of labor, despite everyone's commitment and efforts to contribute, the workload wasn't always shared equally. This situation arose not from a lack of dedication, but rather from a shortfall in our communication about decision-making and task allocation. Occasionally, we found ourselves overcomplicating matters, which led to inefficiencies and a sense of misdirection within the team. At times, our lack of a shared understanding led us to overthink solutions or methodologies, ultimately pulling us in different directions. If we had done more extensive planning and worked more to have a shared vision for the project we could have solved some of these problems.

This imbalance in work distribution sometimes created an undercurrent of tension in the team. Therefore a more structured approach to task allocation and clearer communication about responsibilities could have mitigated this issue. Moreover, the challenge of being able to be accommodative of one another's schedules for meetings to effectively discuss work made collaboration even harder. Conflicting schedules, in reality, mean that one cannot, at all times, be able to contribute at a go during discussions. Hence, decision-making was at times done without full participation.

During our project we tested a few ways to have meetings and apply scrum. Some of the things we tested were having daily-scrum updates on Discord, mid-week updates, having only one in-person meeting every week. In the end we decided to have two meetings every week, one in-person at the end of the week and one meeting that could be an online meeting mid-week.

Using Discord as a communication device was a great decision. It provides a quick and diverse way for us to exchange ideas which definitely helped in keeping our project on track. The truth is that our experience with Trello was the complete opposite. It's meant to keep our efficiency high by organized tracking of things to do, but its potential was realized only at the most basic level due to wrong application and lack of knowledge. We also had different knowledge of and experience working in Trello which hindered our ability to work according to scrum in the beginning.

The most significant communication challenges came from our inability to fully listen to each other's input. We were often preoccupied with our own concepts and plans for the project. This habit led to frequent misunderstandings within our team and even extended to our interactions with the client. In addition to that we switched note-takers between meetings, having the same note-taker as originally planned would have been easier as well assuring the same quality of notes. Not having the same note-taker and an agreed form of note-taking contributed to unnecessary conflicts and the overthinking of decisions that were already made. Having more strict note-taking would have been invaluable for resolving disagreements or confusions by offering a definitive account of what was agreed upon, thereby preventing misinterpretation and miscommunication. One thing we should have done from the beginning was having an agenda where everyone could add their point for their main responsibilities to be discussed in the next meeting. This would have resulted in more productive meetings and less misunderstandings.

To summarize, we had fun working together and have learned a lot about how to efficiently work as a group when applying a somewhat agile way of working. In the end we feel like the course is a good preparation for working in industry, highlighting the different ways of working and the struggles that come with it.

Annex 1: Deliverable D1

Introduction

This annex provides information about the initial project delivery D1, which was a contract defining the members of the team and the processes that the group would use to specify, design, implement, and validate the software.

At the start of the project, the group met to form an agreement on the team's collaboration and project execution. After an introduction of the group members where each member presented their strengths and previous experiences, the group assigned the different roles and responsibilities that best corresponded to each member's expertise.

The group then outlined a contract containing rules and responsibilities to improve the process of developing the project. The contract includes limited work hours, inspired by XP practices and weekly meetings to further adopt an agile approach.

To improve the quality of the work done, the contract also contained information on how to implement and refine the work to be done. This includes reviewing the backlog and utilizing a set code standard to make the code more readable and allow for code evolution while reducing the risk of code degradation.

Since the group all agreed that communication is a major factor in succeeding in any project implemented by a larger number of team members a set of collaboration tools were decided on.

Deliverable D1 Group Agreement

Members: Thom Kunkeler, Isak Hvarfner, Sabereh Hassanyazdi, David Carlsson,
Efdal Erdem, Shiqi Shu, Elsa Grönbeck

Introductions and Expertise

Thom Kunkeler:

I am a PhD student in Computing Education with a particular interest in why and how students engage with computing. Because of this, I have a good understanding of how to engage people in learning while creating a positive work environment for others. Before starting my PhD I worked at the Raspberry Pi Foundation for two years, which has given me extensive experience working within a team, and my research there allowed me to develop analytical and writing skills. In both my work and personal life, I

am people-oriented, with a particular emphasis on communication and wellbeing, and I believe that this will be beneficial to the team as well.

Isak Hvarfner:

I'm a third-year CS bachelor with a decade of programming experience, both as a hobby and in university group projects. Game projects include custom chess with a web UI, Connect Four with AI using Haskell, and a tower defence game in Rust. My favourite language right now is Rust, but TypeScript is a close second because of its ecosystem. When it comes to natural languages, I've essentially come to replace my native Swedish understanding with English, and now I know more English idioms and words than Swedish ones. No matter the language, I get high on dopamine when I find the best way to communicate an idea.

Sabereh Hassanyazdi:

During my three years in computer science, I've frequently stepped into the roles of Scrum Master and Project Manager across different projects. This experience has enriched me with a special mix of skills and knowledge. My strengths are primarily in communication and planning, making it easy for me to connect with team members and keep projects on track. Secondly, when it comes to programming, Python and Java are the languages that I am most comfortable with.

David Carlsson:

I studied the "Master's Programme in Computer and Information Engineering" between 2015 and 2020. Currently, I work for a medtech company as an embedded engineer tasked with maintaining older products and developing new products. Working for this company has given me experience in handling customer requests, writing technical documents, and integrating code developed by different companies. I have experience in many programming languages, but C is the language that I use most frequently. Besides embedded programming, I have also developed a few desktop utility applications using either C++ or C#.

Efdal Erdem:

I bring my expertise in programming and writing to the team. In the realm of programming, I have extensive experience with languages such as C++, and while my proficiency in Python is at an intermediate level, I am keen on utilizing it for this project. My familiarity with various technologies allows me to navigate the technical aspects seamlessly. Moreover, my writing skills are not only focused on clear communication but

also on creating comprehensive documentation, catering to the team's communication and documentation needs effectively.

Shiqi Shu:

I'm a Master student in Computer Science with a Bachelor of Software Engineering and a Bachelor of Informatics. Basically I'm familiar with Java and Python, but I'm willing to learn more. For now my role is Copy-editor which is kind of my favorite part. I am able to put the files in a tidy order. I'm always able to fit right in with the gang. One of my proudest skills is being able to provide engaging speeches and concise presentations.

Elsa Grönbeck:

I'm a Master student in Sociotechnical Systems Engineering. When it comes to programming languages, I'm familiar with a few, but my two favorites are Python and Java. During my spare time, I'm a project manager for Engineers Without Borders. One of my strengths, both as a leader and as a teammate, is that I'm a good listener. I like to tackle challenges, and I have a lot of patience when it comes to solving problems.

Individual interest

Thom Kunkeler:

I am involved in the teaching of two project-based courses, namely Information Management Systems and Global Software Development. However, both of the courses lack a theoretical foundation in software development, and I believe that this course can fill that gap. I would also like to experience a project-based course from the student perspective, so I can improve my teaching.

Isak Hvarfner:

Initially, it was because I was told I'd look good on the transcript, but I of course recognize that there is immense value in being able to develop software in a team efficiently and that knowing the methodology and gaining experience doing so will help.

Sabereh Hassanyazdi:

I am particularly interested in this course as it offers valuable knowledge in Agile development methodologies and real-life teamwork experiences, which are crucial for preparing me with the skills and expertise needed for a successful career in software project management.

David Carlsson:

Since I work for a company whose main priority is not to develop software and new products the processes have not yet been standardised. Our major customer is a

company with extremely strict processes to test new products before integrating them into their products. My main interest in this course is to learn more about formalising the development process to increase the quality of work done. As a consultant, I also believe that I can add extra value to my customers by making the process of developing new products more standardised and cost-efficient. It is also a good way to justify fee increases.

Efdal Erdem:

I am unsure about which specific field of computer science I want to work in the future. While I have been involved in artificial intelligence and data science projects before, I haven't explored software engineering. I believe that taking this course will not only allow me to experience a simulation of the software side but also help me decide whether I would like to pursue it in the future.

Shiqi Shu:

I'm attracted by so-called 'management' and agile development. I have learned a lot about coding stuff, now I want to do a realistic simulation of real workplace scenarios. And I believe that this course will eventually help me a lot in my future work.

Elsa Grönbeck:

I have always liked teamwork and been interested in how to work best and most efficiently together as a group. I already have some experience in project management in general but since I want to work with software development I want to know how project management is done in this field.

Grade ambitions

Everyone is aspiring to get a 5 in the course.

Roles and Responsibilities

Scrum Master - Thom Kunkeler

As the Scrum Master, I play a crucial role in facilitating and supporting the Agile software development process in our team. I will implement a Daily Update system which will be used by the team to keep each other updated on work and wellbeing.

My responsibilities are:

- Converting the product requirements to a product backlog
- Sprint planning
- Designing the sprint backlog

- Stay informed on team member's daily update and act where needed
- Coordinate sprint review meetings with scrum manager

In addition, my organization skills will ensure that team members know what to do and who to collaborate with. I will also remove obstacles that hinder the team's progress, for instance by approaching individuals who are not following their job description or are not performing up to the team standards. I am also tasked with identifying and implementing improvements to the Scrum process. And, through communication I will develop a team culture where people feel like they can continuously learn and improve themselves. I will work closely with the product owner to manage the product backlog.

Project Manager - Isak Hvarfner

I have the ultimate responsibility of getting the project done in time while meeting external quality standards, using the team members' time and skills efficiently.

- I act as the main contact for stakeholders.
- I define the scope, deliverables and a high-level plan for the project where the course does not do so clearly.
- I identify and take into account risks, such as the project becoming too big, or death.

I always consult with the **scrum master** on how to best incorporate agile principles when defining the high-level plan with deliverables, and to ensure the plan is feasible. To meet external quality standards, I coordinate with the **QA engineer**.

QA Engineer - Sabereh Hassanyazdi

As a Quality Assurance Engineer, my primary responsibility is to ensure the highest quality of our product by effectively utilizing various testing methodologies and tools. I will thoroughly understand the product's specifications and requirements to design and implement appropriate testing strategies. My role involves carefully planning and conducting a range of tests, from functional to performance, to identify any defects or areas for improvement. I will document and report these findings to the development team.

DevOps Engineer - David Carlsson

My job is to provide tools to simplify the team's development and integration. This will be done by using different tools, such as Git for version control. I will also help the team to collaborate to be able to address project needs effectively. Creating documentation for the team on how to use the provided tools and processes will be handled by me.

Communicator - Efdal Erdem

As the communicator, my responsibilities include scheduling time slots when everyone in the group is available, maintaining communication with the client and course instructors, coordinating and organising meetings with the client, and conveying any unresolved issues or questions within the group to the course instructors. I am accountable for facilitating effective communication, ensuring smooth interaction among team members.

Copy Editor - Shiqi Shu

As a copy editor, your role is vital in ensuring the clarity, correctness, consistency, and coherence of written content.

- Thoroughly check and correct any grammatical and spelling errors. Pay attention to commonly confused words and homonyms.
- Rewrite sentences or paragraphs that are unclear, awkward, or clunky. Ensure the text flows logically and smoothly.
- Correctly use commas, periods, colons, semicolons, dashes, and quotation marks. Pay attention to the punctuation style preferred by the publication or the style guide being used.

Group Note-taker & Product manager - Elsa Grönbeck

As the Product manager my responsibility is to identify the clients needs and outline what a successfully delivered product looks like. I will also deliver a vision for the product and make sure the rest of the team understands it. In other words, it's my responsibility to make sure the team is aligned under a shared vision. Throughout the project it is also my responsibility to ensure that the product we deliver meets the requirements given by the client. As the group note-taker my responsibility is to take notes during our meetings and post the meeting notes so team members who could not attend the meeting can read them.

Everyone

All team members are expected to contribute to the tasks listed below.

Requirements engineering

- Engage with the client to identify, gather, and understand their needs and expectations.
- Conduct interview with client to elicit product requirements.
- Analyze and document functional and non-functional requirements.

- Organize and prioritize product requirements.
- Ensure that requirements are written in a way that is testable, and that specifications can be understood by development teams.
- Verify that the implemented solution corresponds to the specified requirements.

Software development

- Translating the product requirements to high-quality Python code.
- Collaborate with the Quality Assurance Engineer to conduct tests and fix identified issues.
- Identify and fix bugs or errors through debugging and troubleshooting.
- Work closely with other members of the team, for instance through pair programming.
- Use a version control system as defined by the Dev Ops engineer.

Technical writing

- Writing clear, concise, and user-friendly documentation.
- Organize complex technical information into understandable content.
- Tailor documentation to audience needs, and considering the user perspective, create documentation that is accessible and relevant.

Extra Information

Working hours

Team meetings are to take place during office working hours. Office working hours are defined to be between 08:00 to 17:00 (8 a.m. to 5 p.m.) except for 12:00 to 13:00 (12 p.m. to 1 p.m.) which is reserved for personal use i.e. lunch. The group's communicator shall inform the group about the day and time the group agreed upon.

The time for the meeting shall be decided by the group at least 24 hours before the time for the meeting. The members of the group shall attend group meetings in person if possible. If the member cannot attend in person, a hybrid solution will be used. If the member is still not able to attend the meeting, the group shall be informed before the meeting starts.

Weekly meeting

An in-person group meeting is to be planned by the Scrum Master each week to facilitate group collaboration. The meetings will take place in a meeting room booked in advance. The Scrum Master is responsible for setting the agenda. Occasionally this responsibility will be passed onto another group member depending on the work that has been done the week prior. At the start of a sprint this weekly meeting will be used for sprint planning.

Midweek Update

Starting Wednesday 31st of January, a midweek update system will be in place using Discord. Every Wednesday before 10:30 AM, team members are to answer the following questions in the assigned Discord channel:

- How are you feeling?
- What are you working on?
- Do you need help with something?
- Are you unable to continue until someone else completes their task?

Using this information, the Scrum master is responsible for assisting group members who fall behind with their specific tasks in a constructive way. Assisting the member is done by inquiring about what is holding the group member back and what type of assistance the group member might need to succeed with the specific task. If a solution cannot be found, and, or the group member does not show signs of wanting to improve the situation, the Scrum master shall discuss a more strict solution with the group.

Client and Scrum review meetings

Client and Scrum review meetings are less structured than the weekly meeting and more flexible. That is, depending on the availability of the TA, these meetings will be scheduled when needed, and these meetings can take place either online or in person.

Other forms of collaboration

To increase the amount of collaboration within the group, each member shall try pair programming at least once. Collaboration between people is allowed; however, the person assigned to the task shall make an effort to find a solution before asking for help.

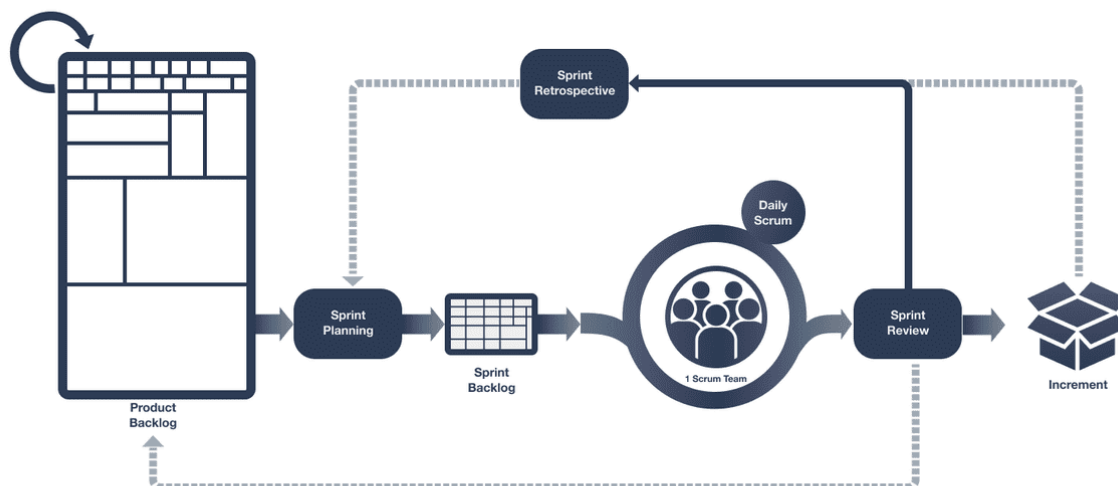
Role Reversal

Individual roles are static, general roles are dynamic. Feb 12 we will evaluate the group agreement and individual performance within the roles. Changes within roles might occur based on this evaluation. When a person needs help in a role, other people in the team can provide support.

Agile and Scrum workflow

The teamwork and project management in this project follows an Agile methodology. Agile is a set of principles and values that promote flexibility, collaboration, and customer satisfaction in software development and project management. Within the Agile umbrella, we are working with the Scrum framework, which provides a structured yet flexible approach to product development. It defines roles, events, and artifacts to help teams deliver valuable increments of a product in short, time-boxed iterations called sprints.

Scrum framework



Based on the Scrum framework illustration by Scrum.org

Figure 1: The Scrum workflow by Scrum.org

The official Scrum guide is available [online](#) and provides insight into how we defined our roles and workflow. At the same time, we acknowledge that Scrum works best in full-time software development teams, and some adjustments had to be made to fit our part-time project. As outlined in this document, we moved away from a daily scrum to a weekly meeting and a midweek update.

In our Scrum workflow, we have a product backlog, and a non-product backlog, including tasks such as writing the group agreement. Among other things, our product backlog consists of both user stories and requirements which allow us to distribute and prioritize work based on complexity of the task. In our project management board Trello, user requirements and stories are stored in a separate list from our product backlog for clarity.

The increment for each sprint is defined by the product backlog, and includes the updated product at each end of the sprint. The product is then reviewed by the client which then allows us to define the increment for the next sprint. An increment should meet a Definition of Done. These are defined for each sprint as follows:

Sprint 1: Requirements Elicitation

User Stories Refinement

All user stories related to requirements elicitation are refined and have acceptance criteria.

Acceptance Criteria Defined

Each user requirement has clear and agreed-upon acceptance criteria.

Backlog Grooming

Product Backlog is reviewed and refined.

Client Approval

Stakeholders have reviewed and approved the refined requirements.

Ready for Sprint Planning

All items in the Sprint Backlog related to requirements are well-defined and ready for implementation in the next sprint.

Sprint 2: Product design and implementation

Product Design

Design documents, including architecture and interface designs, are completed and reviewed (in deliverable D3).

Coding Standards

Code adheres to the team's coding standards and best practices (in GitHub).

Code Review

Code has been reviewed by team members, and necessary adjustments have been made (during prototype review meeting).

Integration Testing

Features have been integrated and tested as a cohesive unit.

Demo Ready

The developed product is ready for a demo to the client.

Review with Stakeholders

A review with stakeholders is conducted to gather feedback and ensure alignment with expectations.

Sprint 3: Outsourcing

To be defined

Collaboration Tools

Trello / Discord / Email / Google Drive

Annex 2: Scrum Retrospectives and Kanban Board Snapshots

Introduction

Annex 2 explains how the group utilized a subset of the tools in the agile toolkit during the project.

Scrum Retrospectives

The scrum retrospective was a way for the group to effectively communicate the parts of the processes that work well, or poorly. The retrospectives established a keen and responsive discussion of the progress and hardships during the previous sprint. In the scrum retrospectives below, the keep, toss, and change from the retrospectives can be found.

1. [Retrospective 220126](#)

- Keep
 - Friendly, clear communication, efficient meetings
 - Project management, i.e. Kanban
 - Good teamwork and effect
 - People are active during meetings
- Toss
 - Daily updates in discord channel, reduce frequency
 - Plan on low fidelity prototype
- Change
 - Use Kanban more consistently
 - Less communication after the meeting and more Trello
 - Focus more on tasks in discord channel
 - Check studium regularly and send emails for any clarifications
 - Finish tasks earlier
 - Use Trello more for task updates
 - Use Trello as a productivity tool

2. [Retrospective 220212](#)

- Keep
 - Actively using the Kanban board
 - Use of Discord and Trello

- Good communication
- Use of channels on Discord
- Weekly meetings
- Friendliness, collaboration efficient meetings
- Toss
 - Midweek update
 - Clean up the drive
 - Unclear version handling of documents
 - Weekend/evening work
- Change
 - Two weekly meetings instead of one longer
 - More collaborative work during the meetings
 - Better preparation before meetings
 - Work as teams in Trello
 - Soft deadline and hard deadline
 - Document version handling with changelog
 - More user requirements on Trello and distribute work according to them

3. [Retrospective 220227](#)

- Keep
 - Reliability and teamwork
 - Accommodating
 - Thoughtful
 - Average the level of group work
 - Nice people
- Toss
 - Crisis in project
 - PM disappeared
- Change
 - Working on weekends and after 17.00
 - Reduce delayed work
 - Workload
 - Time management
 - People don't read what was previously written
 - Be more clear when dividing the tasks
 - More clarity of who does what

Keep	Toss	Change
<ul style="list-style-type: none"> • friendly - clear communication good, efficient meetings project management -- e.g. kanban board • Good teamwork and effect communication. • People are active during meetings. 	<ul style="list-style-type: none"> • Daily updates in Discord channel -Frequency of the updates • Plan on Low fidelity prototype • Shared vision/what tasks need to be done 	<ul style="list-style-type: none"> • Use Kanban more consistently – Person doing the planning can take the initiative. • Less communication after the meeting and more Trello Unclear instructions and must make assumptions. – Focus more on the tasks during the conversations in discord channel. Divide topics in discord Check stadium regularly and send emails for ay clarification • we can do tasks earlier <ul style="list-style-type: none"> - Start using Trello more for the task updates. • Do use Trello as Productivity tools

Figure A2.1 Retrospective 1 on 220126

Keep	Toss	Change
Actively using the Kanban board	Midweek update	Two weekly meetings instead of one longer one (e.g. one meeting on Monday and one on Wednesday instead of midweek update). Potentially Wednesday's meeting could be online.
Use of Discord and Trello	Clean up the drive	More collaborative work during the meetings, requiring that everyone is better prepared for the meeting.
Good communication		Trello is used nicely, but we can also work as teams in Trello – e.g. work closer with people who have similar tasks.
Discord – use of channels	<p>1) There are two versions of files in Drive – finished and draft. This makes it confusing which file to edit after deadlines.</p> <p>2) Coordinate the individual work with the group work (E.g. individual work d3 -> group work d2)</p>	Soft deadline (individual work) and hard deadline (copy edit and submit)
Weekly meetings		<p>Document version handling – one version for handing in, and then for future edits copy the file.</p> <p>Keep a changelog of documents.</p>
Friendliness, collaboration, efficient meetings	Weekend / evening work	User requirements on Trello and distribute work according to them

Figure A2.2 Retrospective 2 on 220212

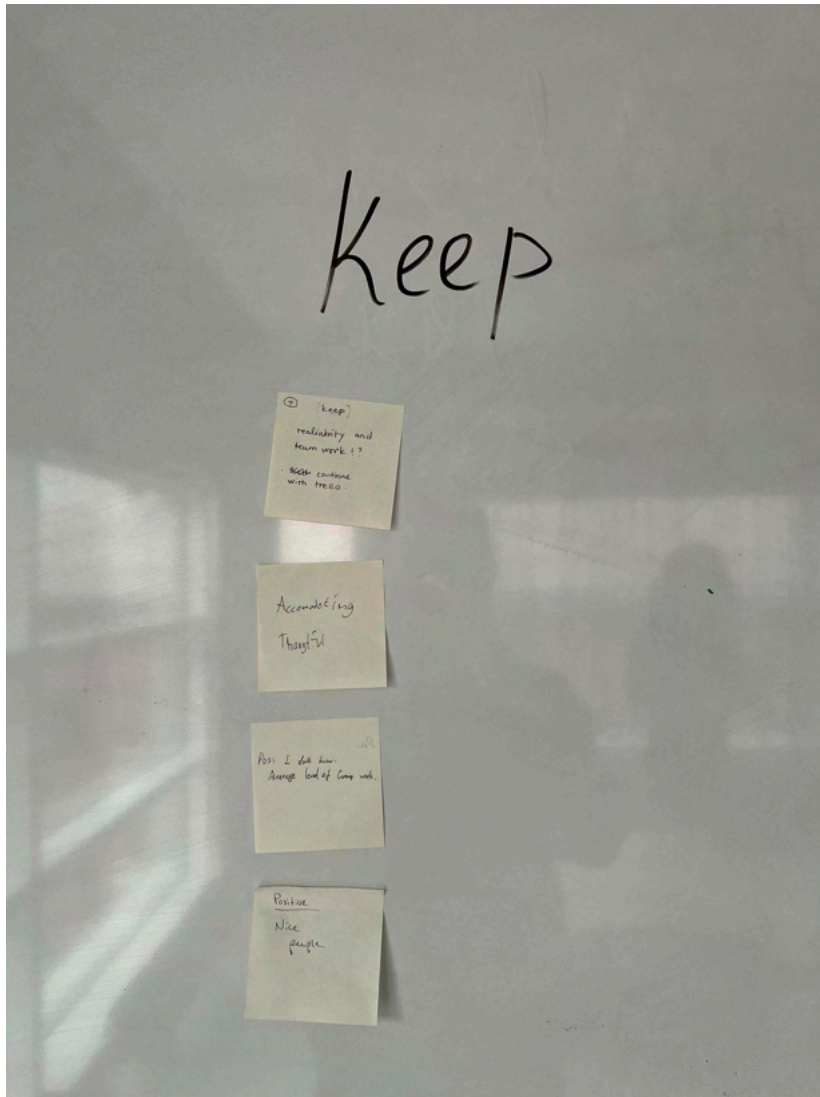


Figure A2.3.a Retrospective 3 keep on 220227

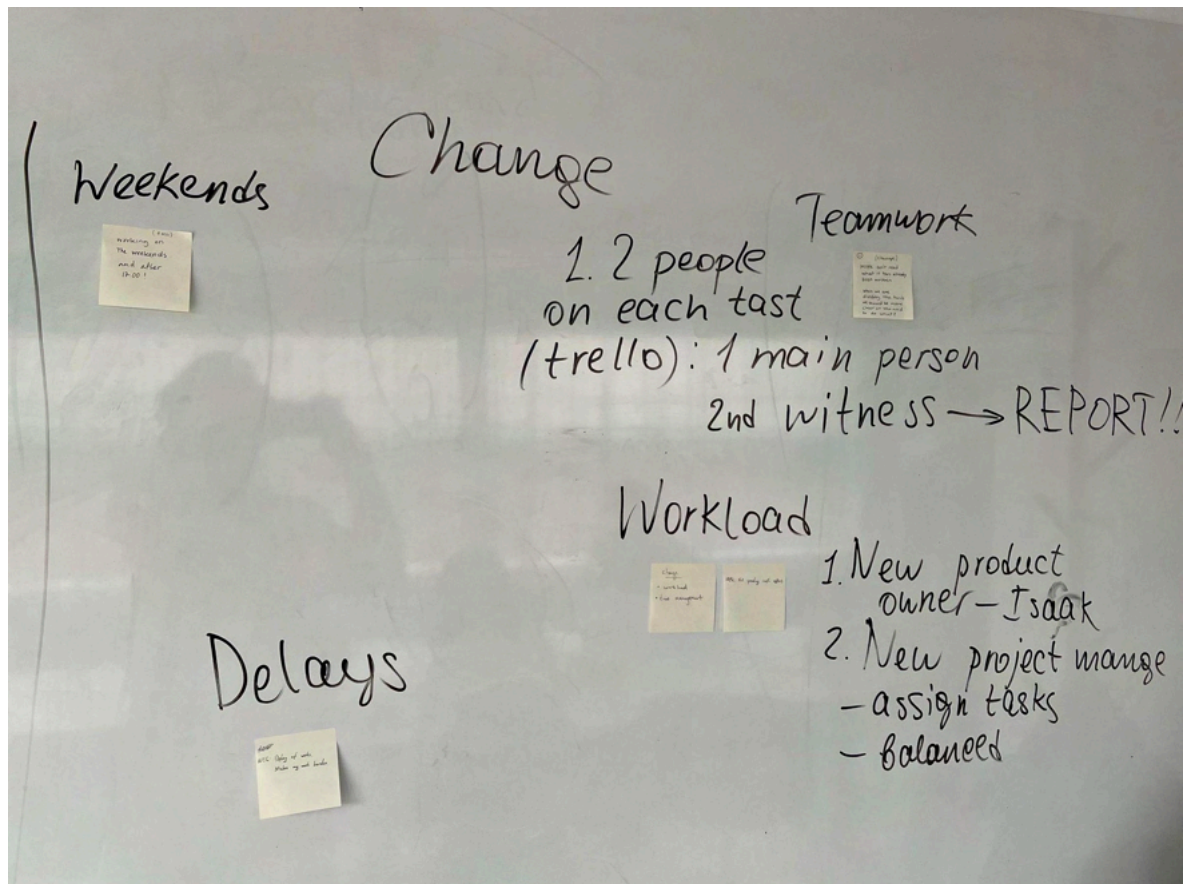


Figure A2.3.b Retrospective 3 change on 220227

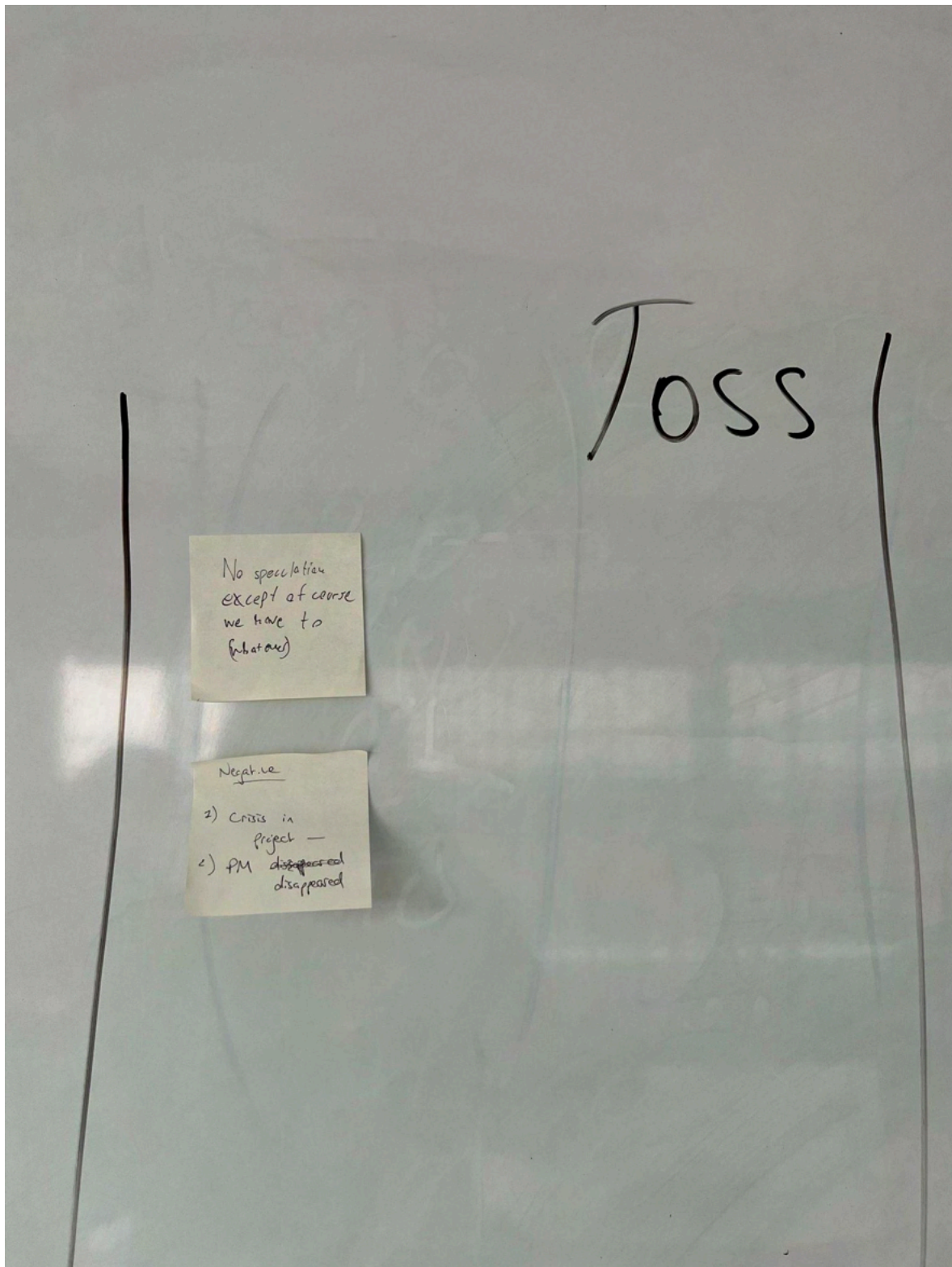


Figure A2.3.a Retrospective 3 toss on 220227

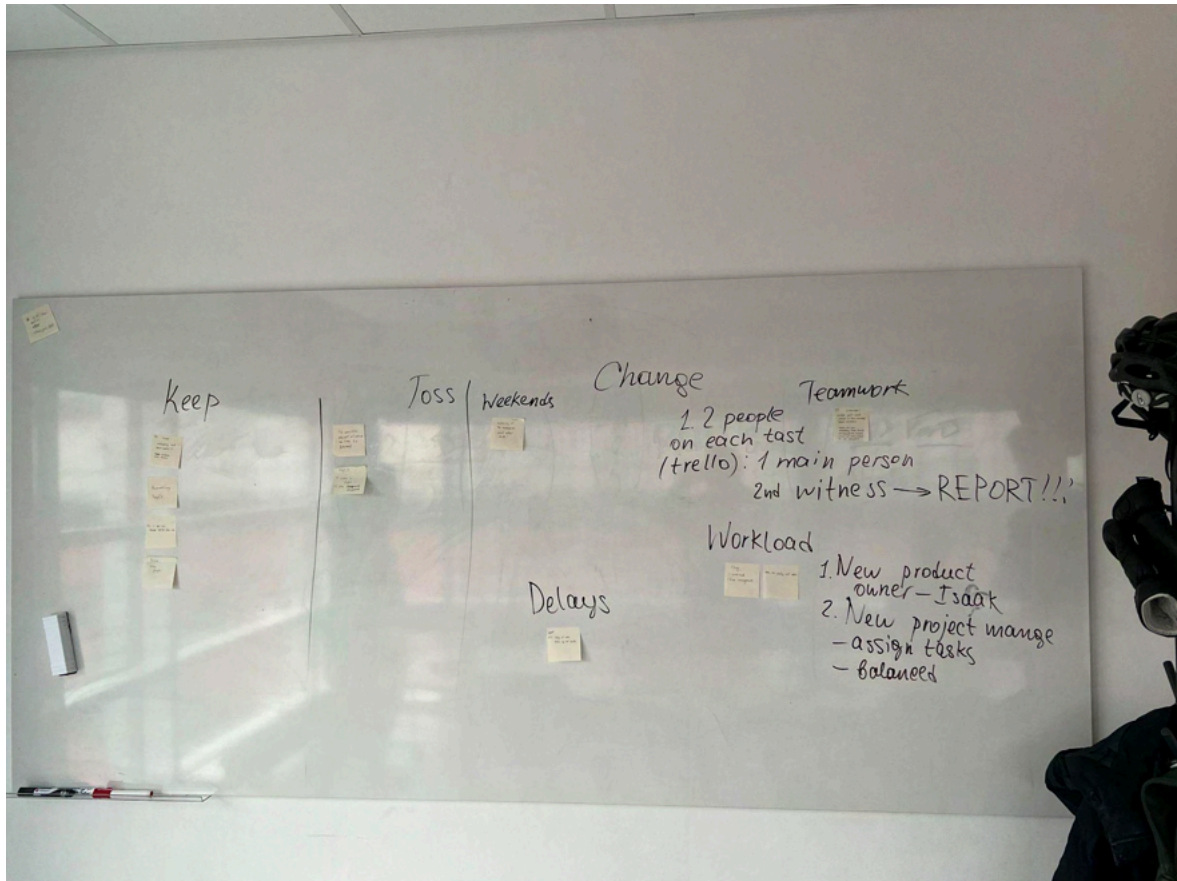


Figure A2.3.a Retrospective 3 whole board on 220227

Kanban Board Snapshots

To enhance the organization and efficiency of work processes, the group utilized a web-based Kanban board owned by Atlassian called Trello. Trello allowed the group members to follow each other's progress, get an overview of the project status, and help the group resolve blockage caused by pre-required tasks.

As the group saw the advantages of using a kanban board, the board started to evolve as illustrated in the figures below. More functionality was added to the board, such as task prioritizing and approximated workload. Besides the functionality, a set of standardized rules was implemented to facilitate a more homogeneous workflow. More information on the rules can be found in [Annex 4: Additional Artifacts, Trello Guidelines](#).

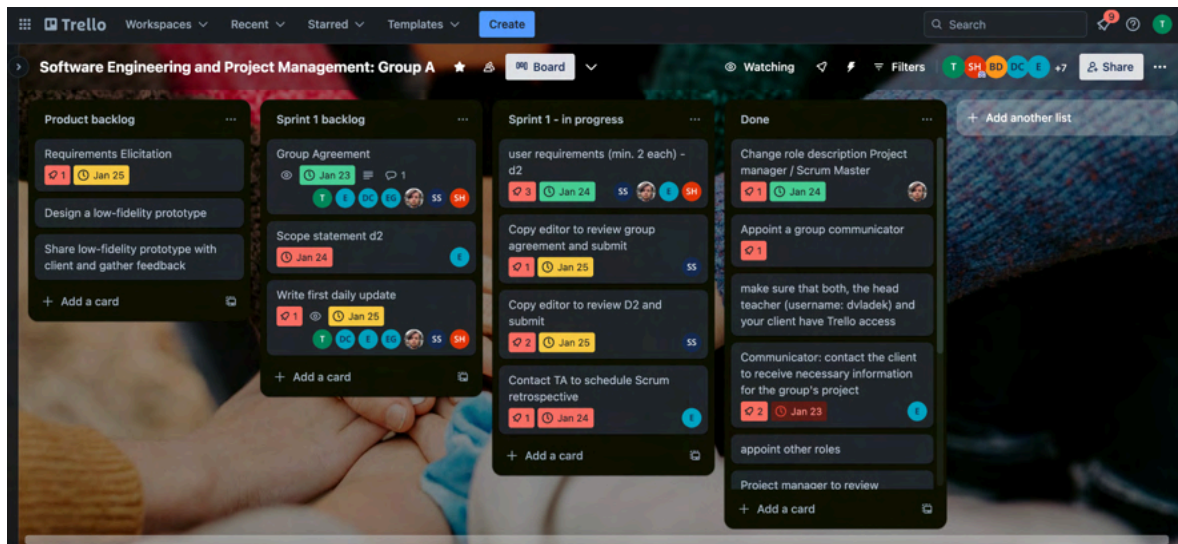


Figure A2.4. Kanban board state 240125.

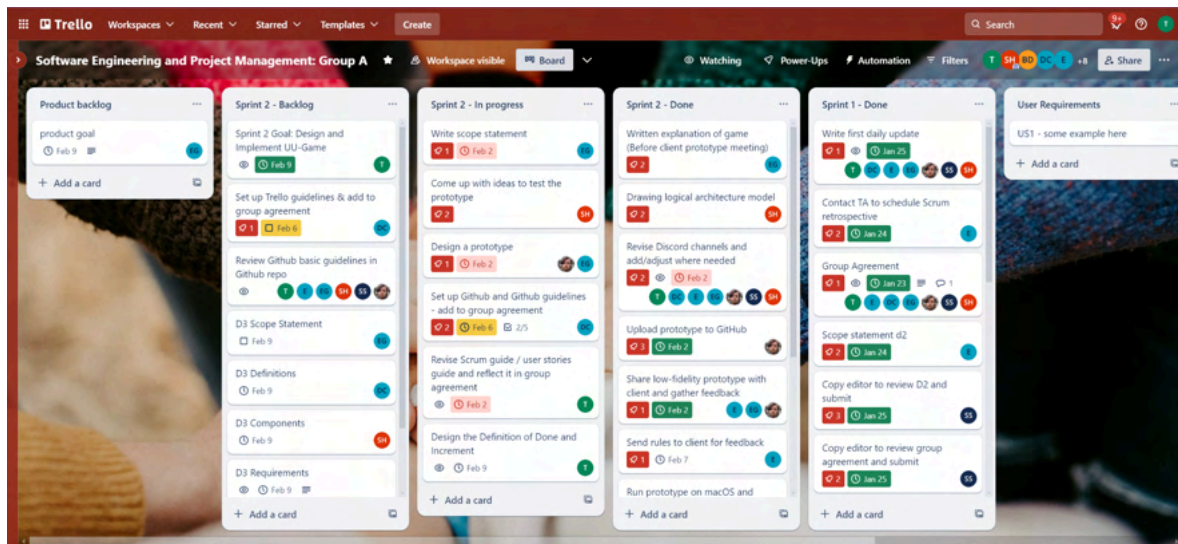


Figure A2.5. Kanban board state 240206.

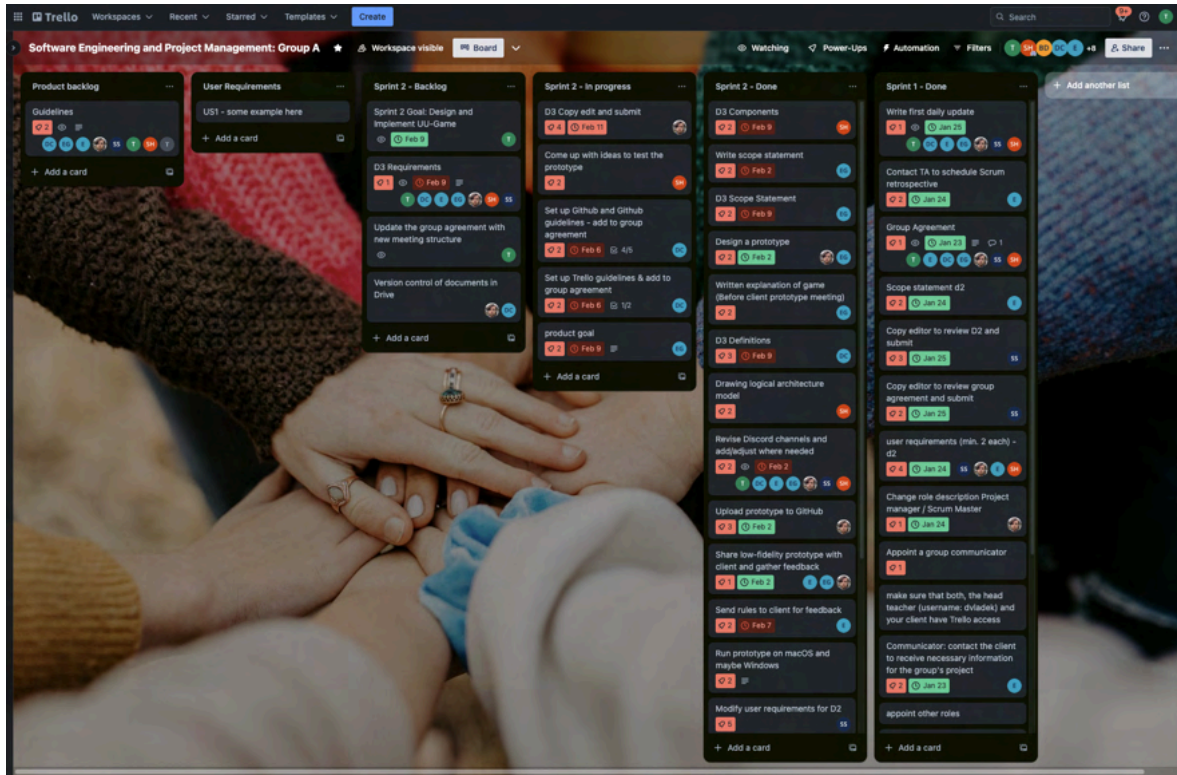


Figure A2.6. Kanban board state 240212.

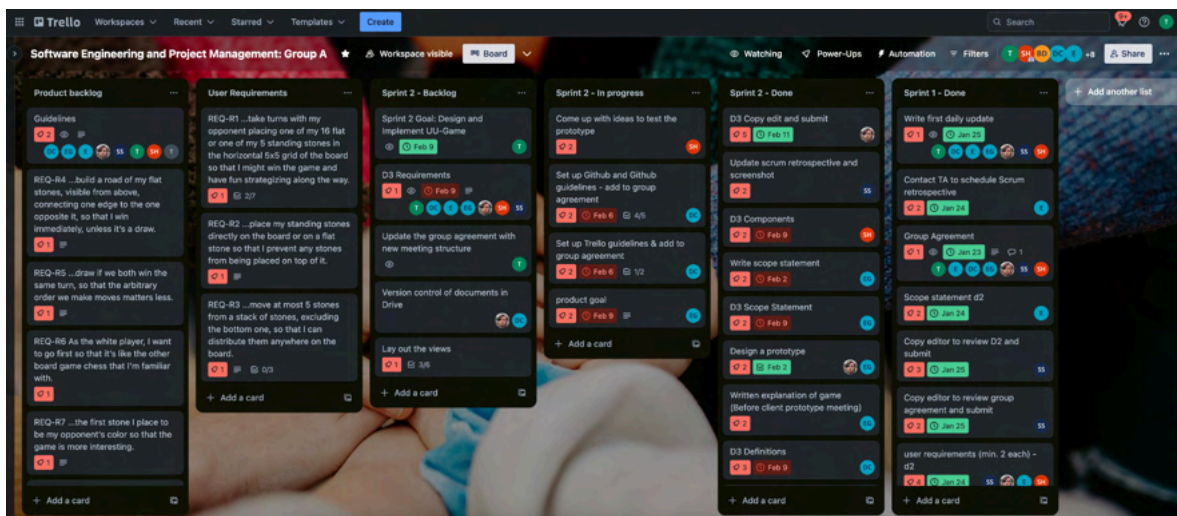


Figure A2.7. Kanban board state 240214.

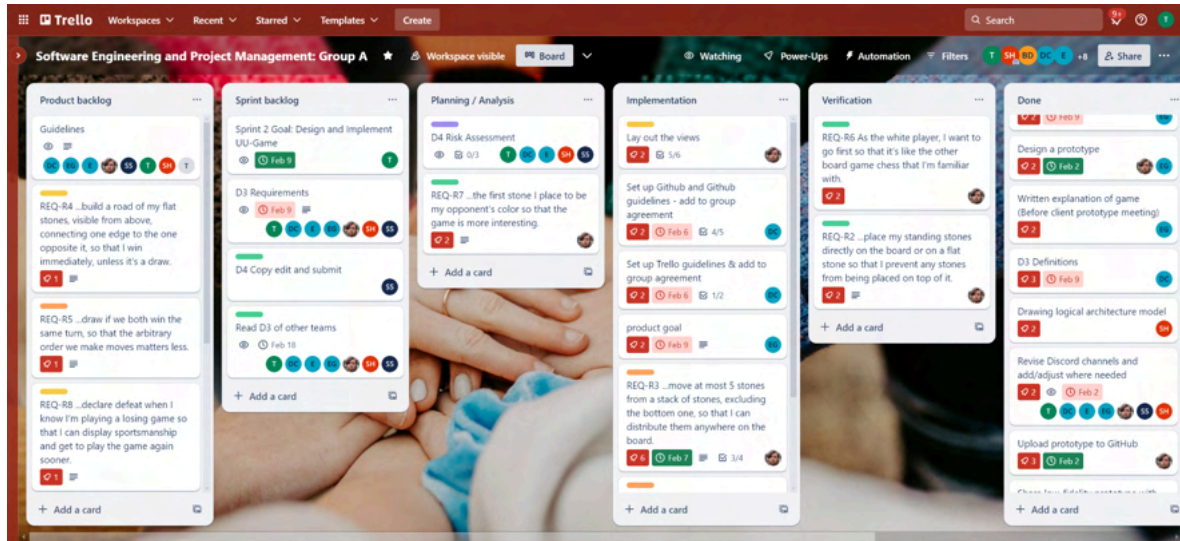


Figure A2.8. Kanban board state 240216.

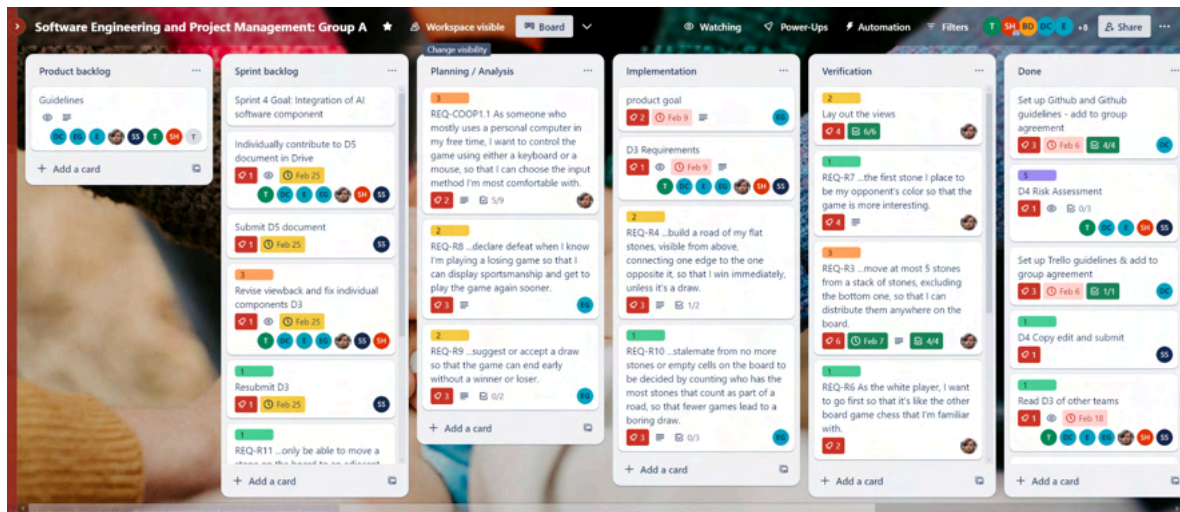


Figure A2.9. Kanban board state 240224.

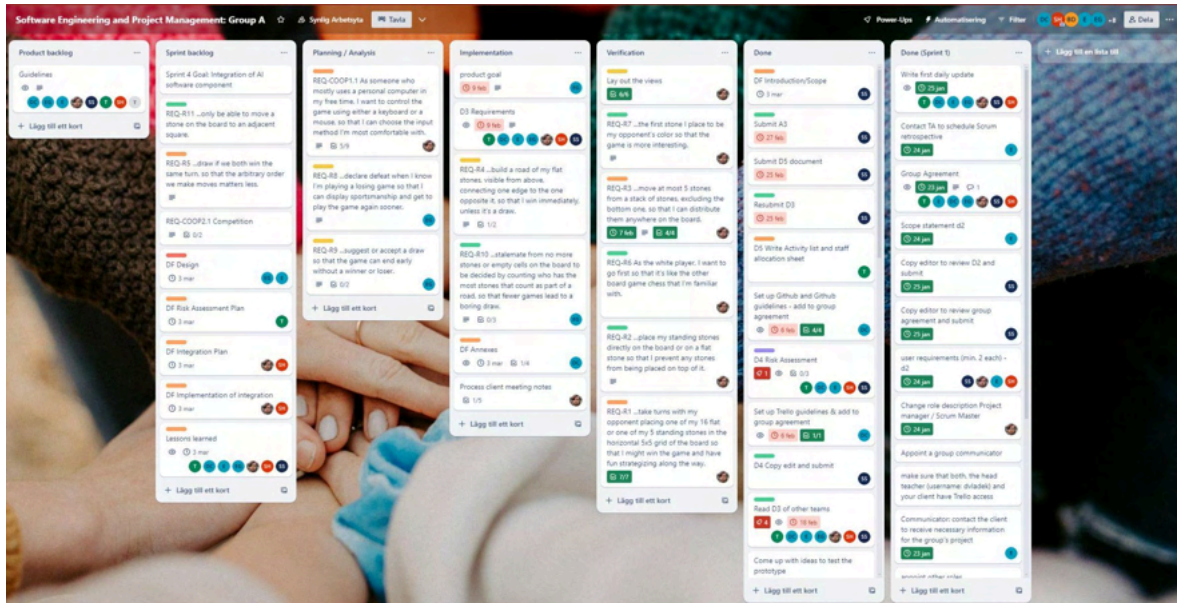


Figure A2.10. Kanban board state 240229.

Annex 3: Code Project

Introduction

To simplify the parallel development of the project, GitHub was used to allow code sharing and version handling. Due to the extent of the code project, the code is not available in this annex. To access the code, see the part below.

Accessing the Code

The code can be downloaded from GitHub. To gain access to the repository, send an e-mail to david.carlsson.6652@student.uu.se containing your GitHub username. The full readme file can also be found in the repository. Below are instructions on how to run the software using Python and a terminal.

Running the code

If you are using Windows, you must first follow the steps further down.

Install Python 3.10 or later. Using the terminal, navigate to root folder of the project and run the command:

```
python src/run.py
```

If you are running the software on a Windows platform the following steps must be made to work around limitations and bugs:

- Use the Windows Command Prompt (or the official [Windows Terminal from the MS store](#)) because the Windows VSCode terminal doesn't have mouse or proper color support
- Use Python 3.11 or earlier due to a bug in windows-curses that prevents a new terminal screen from ever being created. (<https://github.com/zephyrproject-rtos/windows-curses/issues/50>)
- Install the windows-curses package for the version of Python you installed:
`C:/Users/User/AppData/Local/Programs/Python/Python311/python.exe -m pip install windows-curses`

If you are running the game on a platform other than Windows, ensure that Python 3.10 or later is used.

Annex 4: Additional Artifacts

Introduction

Annex 4 covers the artifacts not included in [Annex 2](#).

Contracts

To reduce the integration risks, contracts with groups B and C were established separately to use their implementation of the game engine. This allowed for redundancy if one of the projects failed to deliver the expected quality. The development team could also freely select the game engine interface that would best suit the implemented platform. Below are the contracts agreed upon.

Contract for Code and D3 Documentation

This agreement is entered into on the 29th of February 2024, between group B, hereinafter referred to as the "Provider," and group A, hereinafter referred to as the "Recipient", collectively referred to as the "Parties".

1. Scope of Work:

1.1 The Provider agrees to deliver the code for their UU Game AI component. They also agree to provide their D3 for documentation and understanding purposes.

2. Deliverables:

2.1 The Provider shall deliver:

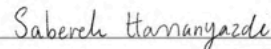
- The complete code of the AI component of their UU Game at a singular point in time as a ZIP file.
- For the contents of this ZIP file and any Git commits before and including the Feb 18 commit cc04e8d, henceforth referred to as the "Software", the Provider grants Recipient irrevocable permission to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense and/or sell copies, subject to including credit to Provider with any substantial portion of the Software and subjecting any entities granted a sublicense to these same conditions.
- D3 visualizations as per project requirements.

3. Meeting and Collaboration:

3.1 Provider graciously agrees to answer questions about the delivered AI component via email.

4. Payment:

4.1 This collaboration is undertaken on a mutual exchange basis and does not involve monetary compensation.



Sabereh Hassanyazdi
Group A



Therese Björkman
Group B

Figure A4.1. Contract with group B for the use of their game engine software.

Signed Document

via <https://min.ebox.nu>



Software Integration Agreement

Signed text:

Jag har läst och förstår innehållet i PDF-filen (1) och samtycker och godkänner allt som avtalas däri, som om jag skrivit under en fysisk utskrift av PDF-filen (1):

(1) Avser PDF-fil enligt följande

Namn: Software Integration Agreement.pdf

Storlek: 57341 byte

Hashvärde SHA256:

7803d762ff0df0b8b86917638c7b88b31394cd076dc2bb9341c805c3f57c9474

The original file and all signatures are attached to this PDF.

To open the attachments, a dedicated PDF reader may be required.

Signed By 2:

ANNIKA KRISTIN ELVERS

Signed with BankID 2024-02-23 19:12 Ref: 6a6e96b4-6c74-4edb-b56b-30fed487cf70

ISAK HVARFNER

Signed with BankID 2024-02-23 16:46 Ref: 5dbc77ca-bb77-4acc-ab61-4a4b16a4ddb8

Figure A4.2a. Signed document with group C for the use of their game engine software.

Software Integration Agreement

2024-02-23

1 Purpose

The purpose of this agreement is to set the terms and conditions that apply as Group A and Group C communicate and give access to the code of the game AI and/or platform for the board game UU Game.

2 Permission

1. Group A and Group C grant each other irrevocable permission to deal in the current version of their software and any related documentation or files without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute and/or sell copies of it and them, subject to the terms and conditions within this document.
2. Group C also furthermore grants to Group A the above permission to any future versions, which they also agree to make accessible to Group A. Furthermore, Group C must, at any future request of Group A, publish any version of their software or related files under the MIT or GPL license.
3. Group A grants the permission in 2.1 to future versions 1 week after they made accessible if they have not expressed anything to the contrary. During this one week period a more limited permission is granted allowing unrestricted internal development and communicating and exhibiting copies to the client and teachers.

3 Communication

1. Both groups agree to maintain an open line of communication throughout the integration process. Group A agrees to make some effort to support and to answer at least some questions from Group C regarding their software for the purpose facilitating the successful integration of the platform and the AI. Group C agrees to make a reasonable effort to answer any question regarding their project.
2. Group C agrees to, as soon as reasonably possible, communicate if they wish to have certain features that Group A has planned to implement completed by a certain date. An example of such a feature is the scoreboard, which may help Group C with things like selection of difficulty and which player(s) are controlled by the AI. Other examples include the adjacency requirement or a real acknowledgment that the game has ended.

4 Credit and Acknowledgment

Group C agrees to credit Group A for the development of the original Software by prominent display to all users of the application in addition to including it in the code repository in a customary way.

5 Signatures

By signing this document, Annika Elvers as Group C's communicator, and Isak Hvarfner as Group A's project manager testify that their respective groups have decided, collectively or through other way of governance, that they agree and promise to uphold this agreement.

Figure A4.2b. Agreement with group C for the use of their game engine software

Meeting Notes

Below are the notes and summaries taken from the group meetings. This does not include the scrum retrospective, since these are mentioned in [Annex 2](#).

Meeting 220123:

e) Are roles static (everyone will have the same role(s) for the entire project) or fluid (you will change roles during the course)? When and how you are going to make changes, if any?

Individual roles are static, general roles are dynamic. Feb 12 we will evaluate the group agreement and individual performance within the roles. Changes within roles might occur based on this evaluation. When a person needs help in a role, other people in the team can provide support.

f) What communication/collaboration tools are you going to use besides Trello? How?

Discord / email / Google Drive

g) Other specific rules agreed between the members of the team (e.g., how meetings are organized)

During working hours. Online and in person.

h) How you want to proceed in the unlikely case there is/are some free-rider/s in the team?

(Constructive) Meeting. Are there any issues that we can help you resolve that are causing the freeriding behavior.

Scrum master responsible.

i) When and how are you going to collaborate?

Weekly meeting using Doodle (so no fixed time).

Booking a meeting room.

Hybrid option if need be. In-person meeting priority.

Try pair programming at least once.

The collaboration between people is flexible, each subtask is approached on a task-by-task basis.

Meeting with TA on Friday 26-01

Revise team contract individually before end of Wednesday 24-01

Final revision of Team contract by copy-editor and submitted by Thursday 25-01

Sabereh to submit Kanban link Tuesday 23-01

Meeting 220130:

Missing: Shiqi

Agenda

- Using Trello / Discord
 - Less #general discord, more diverse channels, more direct messaging, more Trello (Dev Ops guidelines)
- Changing daily updates to weekly updates?
 - Start of week Meeting
 - Midweek Update
 - In-person meeting from 10:00 onwards (Doodle)
- Low-fidelity prototype or is the hi-fi one flexible enough?
- Setting up client meeting

Sprint 2 planning

D3 due: Sun Feb 11, 2024

Sprint 2 = week 5, 6, 7

To do

- 1) Revise group agreement: group meetings in-person, client meetings whatever works best.
 - Start of week Meeting
 - Midweek Update
 - In-person meeting from 10:00 onwards (Doodle)
- 2) Develop low-fidelity prototype and feedback with client
- 3) Set up GitHub & Guidelines

- 4) Set up Trello guidelines – including adding tasks, everyone responsibility.
- 5) Upload prototype to GitHub
- 6) Written explanation of game (before meeting)
- 7) Write up scope statement D3
- 8) Revise Scrum guide / User stories resources and reflect it in workflow
- 9) Monday 10-12 AM meeting
- 10) Drawing logical architecture model and coming up with ideas to test the prototype
- 11) Revise Discord channels, add/adjust where needed (everyone)

Hold

- 12) Set up meeting with Game Engine group to agree on interface between the different components

Meeting 220131:

D2:

User requirements should from user perspective, how they wanna play the game, not about technology.

May be good for 20-25 requirements.

Functional: Developer

Non-functional: Tester and how it can be improved in the future

Just regard the user as a player.

Should have a limitation of standing pieces.

Up to five to move from a stack and put it everywhere

4 boxes: 2 for each player. 1 box for stand pieces, 5 pieces. 1 box for flat pieces, 16 pieces.

This should work in Windows as well. As a non-functional requirement.

Another non-func: Should work in the future version.

Meeting 220205:

Agenda

1. Scrum guide and user stories (25 min)
2. Turn the rules into user stories, but recognize when ambiguity is a quick code change (20 min)
3. Go through the prototype and the tasks that need to be completed in addition or before the user stories (20 min)
4. Git guidelines (10 min)

Pre-notes

Finish the rest of the user stories right here on the spot? They are going to be what we go through to verify it all works later. This also means rules of the game D2 Initial requirements.

In order to do scrum properly we should do another sprint planning, since we didn't even know how to do it the previous meeting. I like the way they do it in slides 19-23 in Files/SEO Project because it aligns with the links Thom has now read. But the priority can be a simple low/medium/high.

Notes from Elsa if you are only doing an in-person meeting

How I'm doing and what I'm doing this week:

Feeling like shit and coughing like an idiot.

This week I will be focusing on my thesis draft except for this course so I will be pretty busy! have a draft for the scope statement, but I want to finish it when we have more text for D3 since they also want the statement to be a bit of a "summary" of the doc.

Please read [Rules/Explanation of the game](#) before going through the prototype, after listening to the last client meeting I realized we have some misunderstandings of what the client has asked for. So please read so we all can have a common understanding of what she has asked for. I have tried to write the rules as straightforward and to the point as I could, but that doesn't mean they are easy to understand for someone else. If you

have any questions, comment them in the doc. (Isak: Note that most unclear rules mean like a one-line code change and therefore don't need much attention)

Secondly, I think it's good if we send an email to the client. Firstly, asking about how to move pieces that are already on the board. If you still can only move them to the adjacent squares or if you now can place them anywhere. From the last client meeting it seems that you understood that she had changed this to placing them anywhere but when I listened back to it I could not find where she confirmed this. Also have we emailed about adding the stuff we agreed on in the client meeting to her to put it in the contract as she asked for?

Thirdly, if anyone has notes from the first client meeting please send them to me. I got a lot of what she said but I surely missed some too. So it would be nice to double check with what you got from the meeting. Here, is my email lillao.holmstrom@gmail.com or elsa.gronbeck.0516@student.uu.se

Meeting 220212:

To do

- Update the Group agreement with a new meeting structure (Thom)
- Add reqs to Trello and distribute coding work (Product team)
- Version control for the documents (David/Isak)
- D2 resubmission

Implementation

- Full instructions / Summary of instructions (user input e.g. "h").
- Number of stones left to place
- Storage box / moving hand

Documentation -> artifacts

Meeting 220212:

Integration

Difficulty: yes or no?

This is less about integration and more about project management.

In D5: we need to be clear what there is to be integrated from group B.

What was in the system of B?

What changes have been made to the system?

Prototypes

A few digital prototypes were developed for the project. The prototypes were intended to spark discussion with the different stakeholders. When demonstrated to the client, the prototypes would help to elicit new requirements, get a better understanding of previous requirements, and validate implemented requirements.

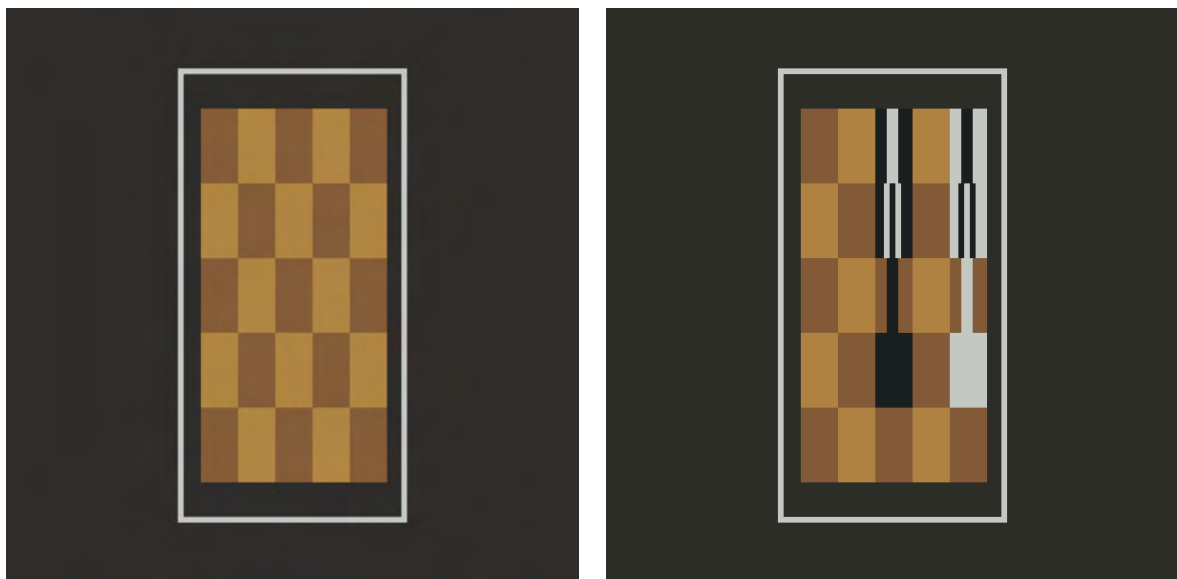


Figure A4.3. The first prototype of the software



Figure A4.4. Software prototype for the client meeting on 240227

Trello Guidelines

To homogenize the use of the Kanban board, a few guidelines were introduced. These guidelines are not limited to Kanban use but also endorse some of the concepts used in the scrum process, adapted for a small-scale project.

1. Backlog:

- 1.1 Each member can pick an unassigned task if they have enough free points.
- 1.2 Point Allocation to Tasks: Use the 5-point estimation process for task sizing (1 to 5).
- 1.3 Limit WIP: No member is to be assigned more points than currently available for that member.
- 1.4 Point Allocation to Members: Each member can have a maximum of 5 points.

2. To Do:

- 2.1 When grabbing a task from the sprint backlog, assign yourself to it.
- 2.2 The group decides what tasks to move to the sprint backlog.
- 2.3 The Scrum Master ensures that the correct tasks are in the sprint backlog.

3. In Progress:

- 3.1 When working on a task, move the card to "In Progress."

4. Done:

- 4.1 When a task is completed, move the card to "Done."

4.2 If you cannot meet a deadline, contact the Scrum Master and try to find a solution.

5. **Hold:**

5.1 The "HOLD" column is for tasks that have been decided to be paused temporarily.

6. **General Guidelines:**

6.1 If you are blocked by a task allocated to another member, add yourself to the notifications of the card (Watch it) to be notified when the task is completed.

6.2 If there are several subtasks, add a checklist to the card and check off each completed subtask.

6.3 If there are several members assigned to a task, agree on who is responsible for updating the task card. If it's a group effort, the Scrum Master is responsible.

6.5 If a task is too hard for you to complete and you have given it a fair try, consult another member to try and resolve the task. There is no shame in failing a task, and it is better to receive help and learn than to hide the failure.

6.6 The rules are up for discussion and are subject to change.

Version Control Guidelines

To get better control of the software, the group decided to use GitHub. With several members having different experiences with GitHub, a set of rules and recommendations was decided on. By having a common understanding and structure when using GitHub for version control, the project development becomes easier to manage and maintain.

1. Git

1.1 Some Git rules

There are a set of rules to keep in mind:

- Perform work in a feature branch.

Why:

Because this way all work is done in isolation on a dedicated branch rather than the main branch. It allows you to submit multiple pull requests without confusion. You can iterate without polluting the main branch with potentially unstable, unfinished code. [read more...](#)

- Branch out from dev

Why:

This way, you can make sure that code in main will almost always build without problems, and can be mostly used directly for releases (this might be overkill for some projects).

- Update your local dev branch and do an interactive rebase before pushing your feature and making a Pull Request.

Why:

Rebasing will merge in the requested branch (main or dev) and apply the commits that you have made locally to the top of the history without creating a merge commit (assuming there were no conflicts). Resulting in a nice and clean history.

[read more...](#)

- Resolve potential conflicts while rebasing and before making a Pull Request.
- Delete local and remote feature branches after merging.

Why:

It will clutter up your list of branches with dead branches. It ensures you only ever merge the branch back into (main or dev) once. Feature branches should only exist while the work is still in progress.

- Before making a Pull Request, make sure your feature branch builds successfully and passes all tests.

Why:

You are about to add your code to a stable branch. If your feature-branch tests fail, there is a high chance that your destination branch build will fail too.

- Protect your dev and main branch.

Why:

It protects your production-ready branches from receiving unexpected and irreversible changes. read more... [GitHub](#), [Bitbucket](#) and [GitLab](#)

1.2 Git workflow

Since this is a smaller project, we will use a simplified version of a [Feature-branch-workflow](#). We will use x.y.z to represent what version we are currently using, where x is the major version, y is the minor version and z is the patch.

- We have two main branches, main and dev.
- We do not work directly in the main or dev branches!
- We also have one type of support branch type, "feature".
- When implementing a new feature, we branch from the dev branch and give the new branch a descriptive but short name. I.e. move_validator.
- When the feature is completed, merge dev into your branch to avoid conflicts in the dev branch. Then increment the minor version and merge your branch into dev (This will be in the code at a later decided location).
- Bugfixes/Hotfixes are implemented as a feature type, before merging the fix, increment the patch (z). Bugfixes are allowed to be merged into main if the bug can break the system.
- Update the versions.txt located in the root folder with a short description of what the feature/fix does.

1.3 Writing good commit messages

Having a good guideline for creating commits and sticking to it makes working with Git and collaborating with others a lot easier. Here are some rules of thumb ([source](#)):

- Limit the subject line to 50 characters.
why
Commits should be as fine-grained and focused as possible, it is not the place to be verbose. [read more...](#)
- Capitalize the subject line.
- Do not end the subject line with a period.
- Use imperative mood in the subject line i.e. "Add ability to toggle game token".
Why:
Rather than writing messages that say what a committer has done. It's better to consider these messages as the instructions for what is going to be done after the commit is applied on the repository. [read more...](#)
- Use the body to explain what and why as opposed to how. [read more...](#)

2. Documentation

- Keep README.md updated as a project evolves.
- Comment your code. Try to make it as clear as possible what you are intending with each major section.
- If there is an open discussion on GitHub or stackoverflow about the code or approach you're using, include the link in your comment.
- Don't use clean code as an excuse to not comment at all.
- Keep comments relevant as your code evolves.

3. Environments

TODO...

4. Dependencies

TODO...

5. Testing

TODO...

6. Structure and Naming

Since this is a very small project, there is no specific standard for structure and naming. The source files go in the src folder located at root.

7. Code style

No code style is needed for this project. It could be wise to follow the code style in the prototype to make the code more consistent. This is not a requirement, more of a recommendation.

Sources: [RisingStack Engineering](#), [Mozilla Developer Network](#), [Heroku Dev Center](#), [Airbnb/javascript](#), [Atlassian Git tutorials](#), [Apigee](#), [Wishtack](#), [A successful Git branching](#)

[model, How to Write a Git Commit Message](#)

Accountability System

This is what we decided during the team retrospective:

To ensure an even workload and that what people are writing is consistent with the instructions (and that the instructions themselves make sense / have been clarified) and what has already been written, we will introduce a "witness" system:

Each time you write a text you must recruit a witness who will provide a second set of eyes that check that what you have written makes sense, is high quality, and is consistent with everything. If you had to do additional work in terms of research, clarification or understanding theory etc, you add that in the Trello card, and depending on how crucial it is to what everyone else is writing, write in general or mentions-only on Discord. Both should assign themselves to the same card in Trello and when the witness does the check, they comment on the card.

This way we discover early when there's a problem that can require more meetings, and we get accountability in terms of effort put in so that it can be slightly more equal.

Notes About Rules and Requirements From Client Meetings

Rules

The rules as we have interpreted them from different client meetings

Basics of the game:

- When you open the game the first thing that should see is full length instructions
 - 21 pieces each (is a piece called a stone??)
 - 16 flat pieces
 - 15 standing pieces
- One black player
- One white player
- The board is 5x5 - only for testing? What board does she want?

- Each player has a box for each piece-type (flat/standing), where they can see how many pieces they have left to place on the board
- Both players use the same computer/keyboard/mouse

Starting game:

- The black player starts by placing their opponent's (white) piece
- The white player then places their opponent's (black) piece
- After that the white player places their own white piece and from then on the players take turns placing their own pieces

Placing a new piece on the board:

- When you place a new piece on the board you can place it in two positions flat/standing, you can see how many you have left from your boxes of pieces
- You cannot switch the position of a piece between standing/flat

Moving a piece/pieces on the board:

- You can only move a piece/stack to an adjacent square?
- You can move up to five pieces (from a stack) in one move
- If you have picked up more than one piece you can place pieces in more than one square, but you always have to leave one piece behind when you move into a new square.
- You can move your opponent's pieces too?

Stacking pieces:

- You can not stack pieces on top of a standing piece
- You can stack a piece on any flat piece.
- There is no limit to how many pieces you can stack

Game ends:

- When somebody wins or if there is a draw

Someone wins:

- You win when you have five pieces in a row in a straight or diagonal line
 - Only the top piece of a stack can be part of a row
 - Only flat pieces can be part of a row
- You can admit defeat resulting in the opponent winning the game
- A flat win occurs when:
 - None of the players have managed to get five in a row
 - The board is fully covered, Then the top flat pieces on the board are counted, and the player that has the most top flat pieces winPieces that doesn't count towards a flat win:
 - Standing pieces do not count
 - Or flat pieces that are in a stack but not on the top

A draw:

- First you can suggest a draw to the other player, who can reject it, but if accepted the game results in a draw
- Secondly if two players meet winning conditions on the same turn - one after another (either by a win (five in a row) or by a "flat win") the game results in a draw

Checklist for things that always should be shown on the playing view

- Instructions for the game (shorter version) - sidebar
- Show how many pieces you have left - by number
- Show who's turn it is
- Show error when a player does a wrong move
- In a stack all of the pieces should be visible

Suggestion:

Instead of just showing an error, maybe we can use colors or a similar solution to show which squares you can place the pieces in. Then we also I argue we need less text to explain the instructions of the game

“References”/Notes

Notes taken by Elsa Grönbeck from the first client meeting. The rules have been crossed out if they no longer apply after updates from the client.

From the first client meeting:

21 white

21 dark

Start the game with the black

Place it in both standing and flat position

Winning by reaching the other end of the board

You can stack the the pieces on your opponents

Alternating turns

You can move up to five

Always leave one piece behind

You can't stack on a standing piece

You should have the instructions for the game always available

Show how many piece you have placed

Show who's turn it is

Start with instructions

Computer only game

If somebody does a wrong move, show an error

You can not complete the line if its a standing piece in it

Lines can be diagonal too, from your side to the other side

~~If all pieces are out it's a tie~~

A move can be to switch in between positions

Starting by placing the opponent's piece

The stack is unlimited in amount of pieces in it

How many turns you can play is undecided

Programming language? - Python

Move always to the adjacent places

UU-game

Target - any age

Sidebar always showing instructions

Show the whole stack at all times, with colours

~~You can make the top piece standing in a stack~~

Both players use the same keyboard

Feby's changes (by email 25-01-2024):

I have made a change in mind, hence one of the requirements as well - You can only place standing stones from your hand. You can not flatten them. And you can't stand flat stones.

Additionally for the draw, I say there should be two options. First, there should be a way for a human player to suggest a draw. The other one might reject it. Then there should be an option to claim defeat. Then the other player becomes a winner. Second option is when both players meet winning conditions on the same turn - one after another. It will result in a draw.

If a road has not been built by either player, and the board is fully covered the game ends and the flat stones of each player are counted. The player with the most flats wins. This is called a "flat win." Standing stones do not count, nor do stones under the top most flat stone on a stack.

Other changes from client meeting 31-01:

Should have limitation of standing pieces.

Up to five to move from a stack and put it everywhere

4 boxes: 2 for each player. 1 box for stand pieces, 5 pieces. 1 box for flat pieces, 16 pieces.

Recording meeting 31_01:

<https://recorder.google.com/8fb0ef2e-5391-418d-8ec4-2169542a4149>

Appendix

Who have done what for individual contributions:

Isak Hvarfner	REQ-R2, REQ-I3 NFQ: 2.6.15, 2.6.21, 2.6.22, 2.6.23, 2.6.24
David Carlsson	REQ-R7 REQ-R8 NFQ: 2.6.4, 2.6.7, 2.6.9
Thom Kunkeler	REQ-I3.1, REQ-I.1.1 NFQ: 2.6.1, 2.6.11 ,2.6.2
Ayse Efdal Erdem	REQ-R10, REQ-R11 NFQ: 2.6.14, 2.6.5, 2.6.13, 2.6.10
Shiqi Shu	REQ-R4, REQ-R3 NFQ: 2.6.16, 2.6.19, 2.6.17
Elsa Grönbeck	REQ-R9, REQ-A1 NFQ: 2.6.18, 2.6.12, 2.6.20
Sabereh Hassanyazdi	REQ-COOP1.1, REQ-R6 NFQ: 2.6.3, 2.6.6, 2.6.8

Figures

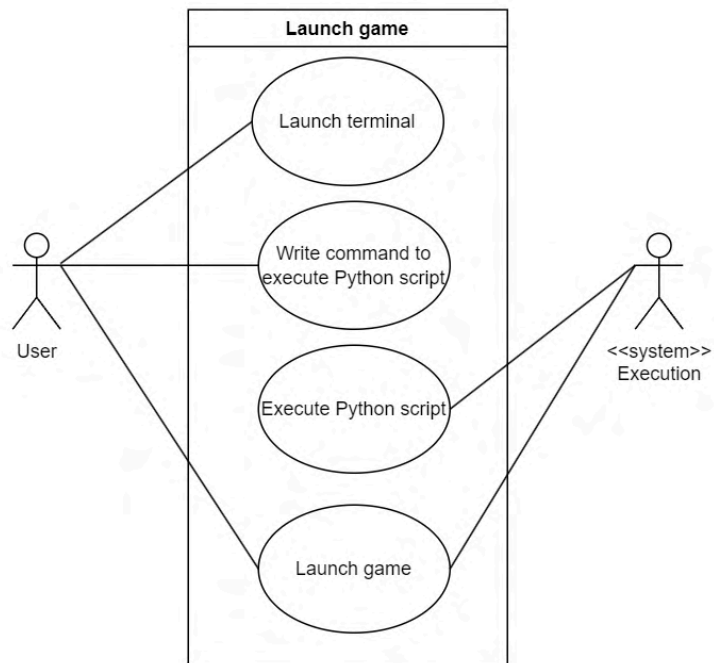


Figure 4. User terminal interaction diagram

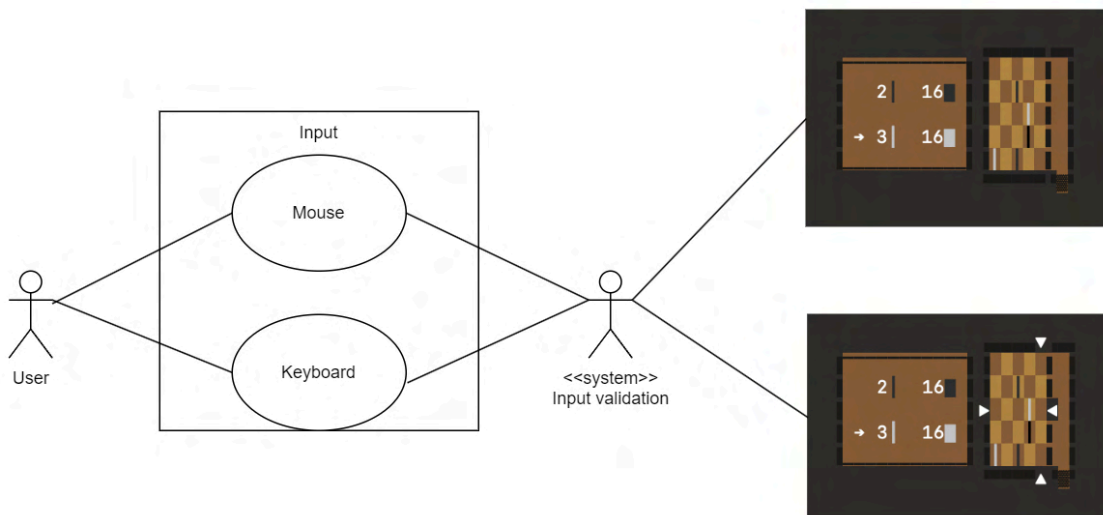


Figure 5. User input and user interface interaction

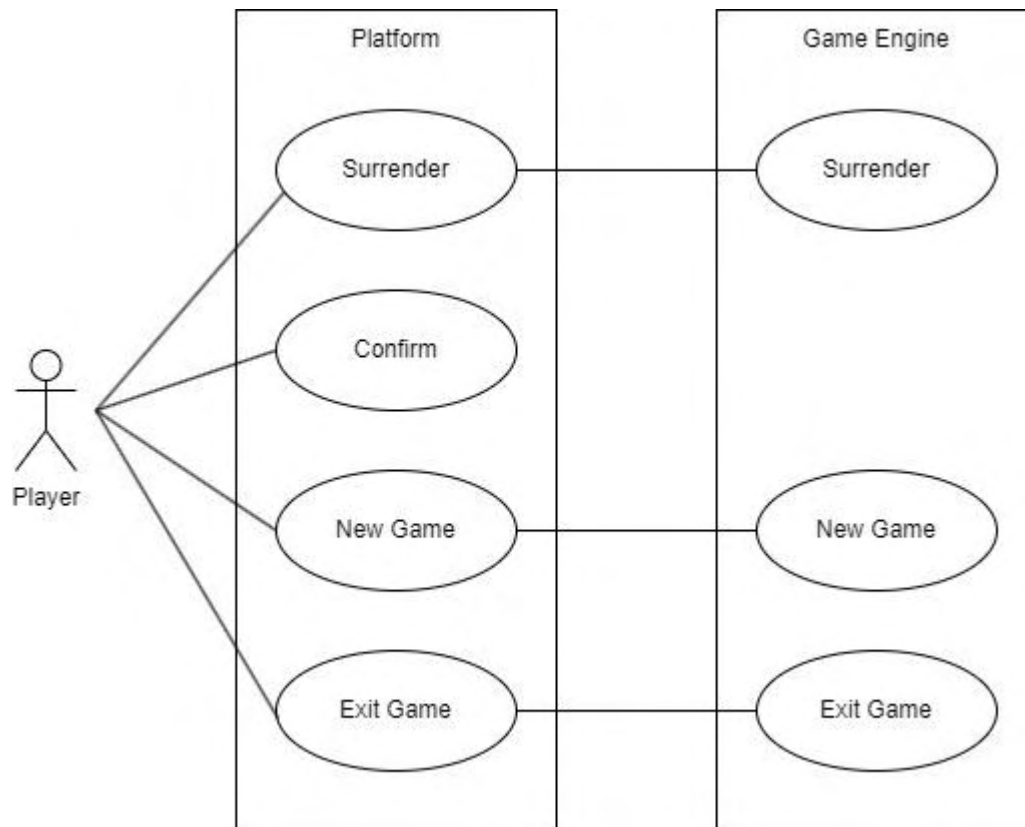


Figure 6.a. Use case diagram for surrender feature REQ-R8

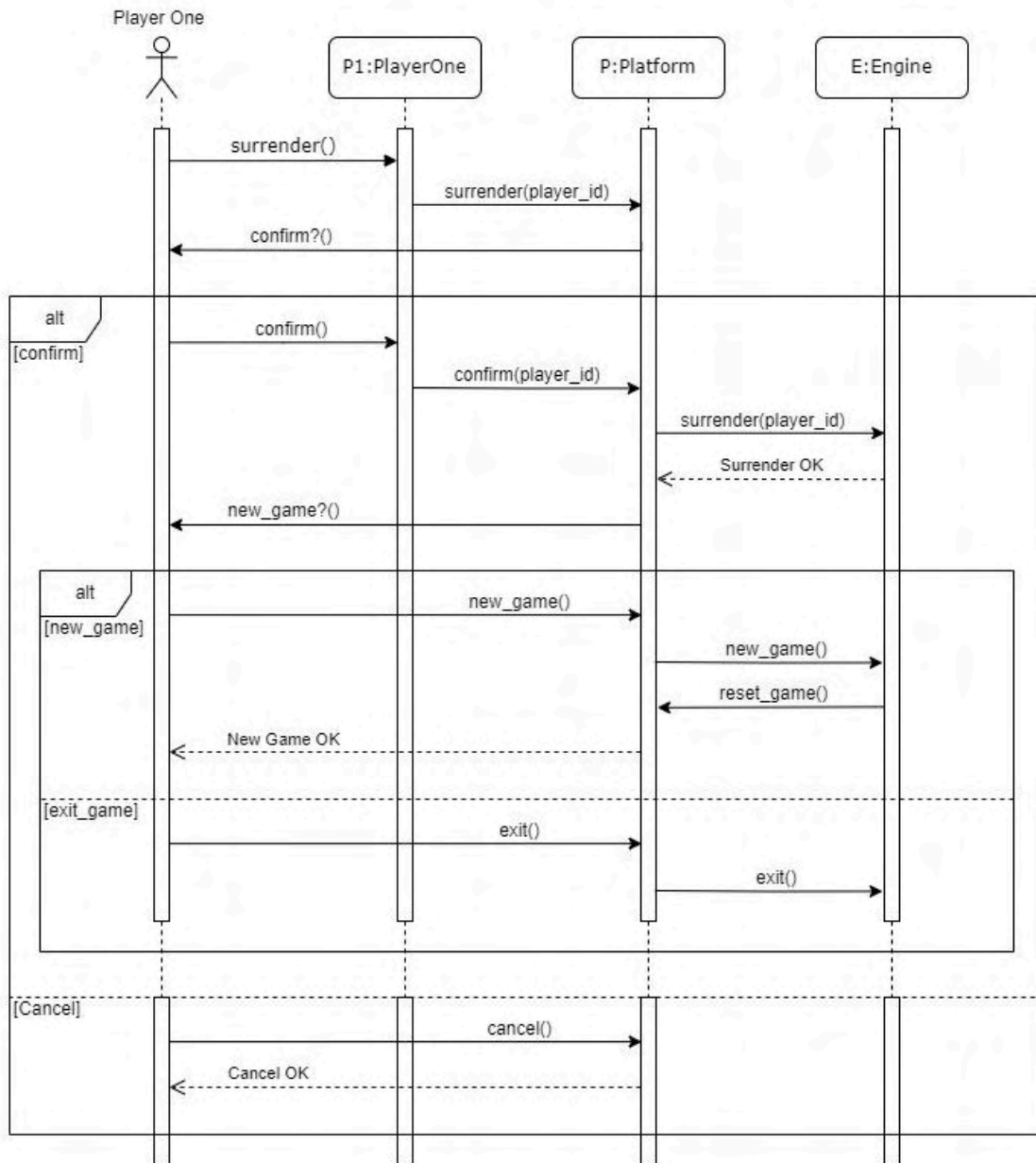


Figure 6.b. Sequence diagram for surrender feature REQ-R8

Due to the decision to use the MVC architecture, this model is currently not accurate and is planned to be changed in upcoming sprints to more accurately reflect the interactions between the actor and the components of the system.

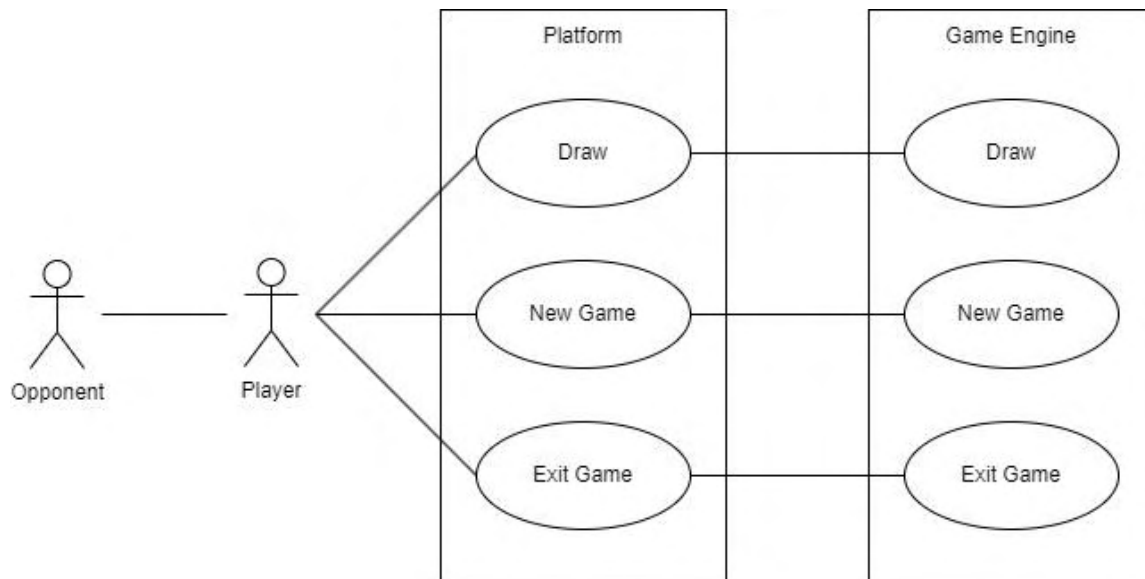


Figure 7.a Use case diagram for draw feature REQ-R9

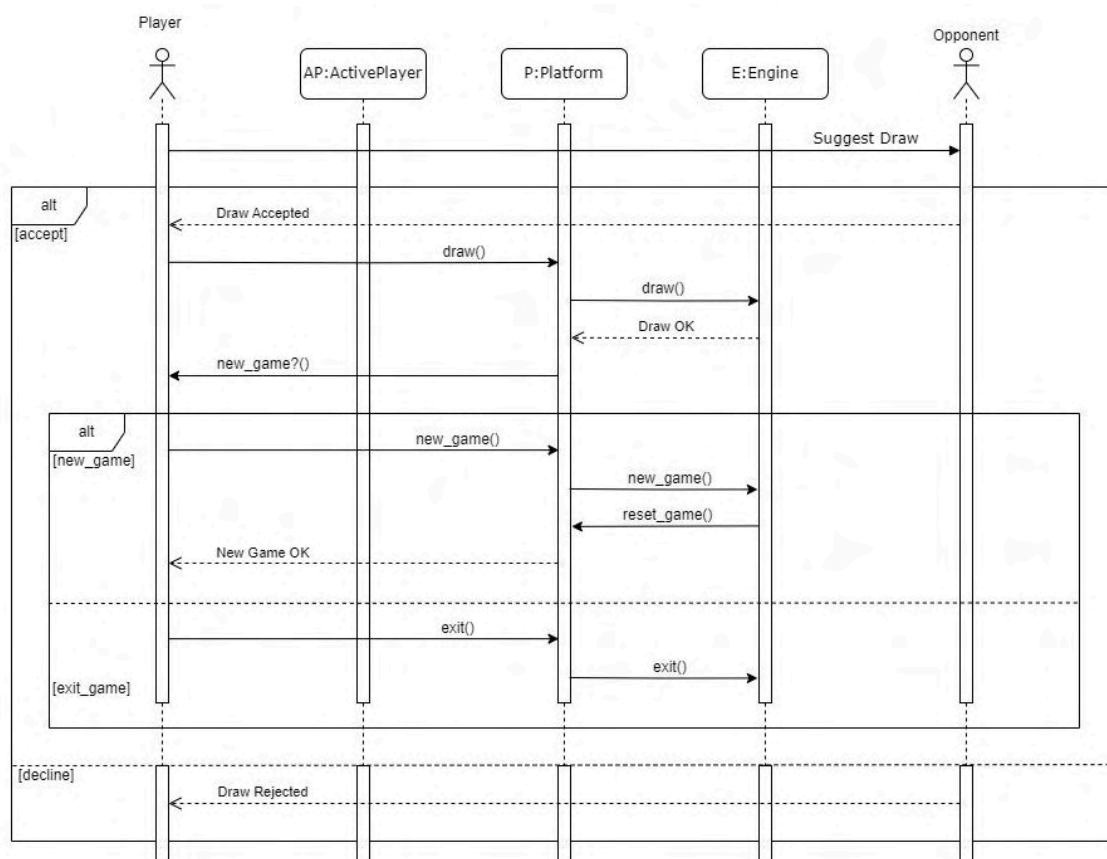


Figure 7.b. Sequence diagram for draw feature REQ-R9

Due to the decision to use the MVC architecture, this model is currently not accurate

and is planned to be changed in upcoming sprints to more accurately reflect the interactions between the actors and the components of the system.

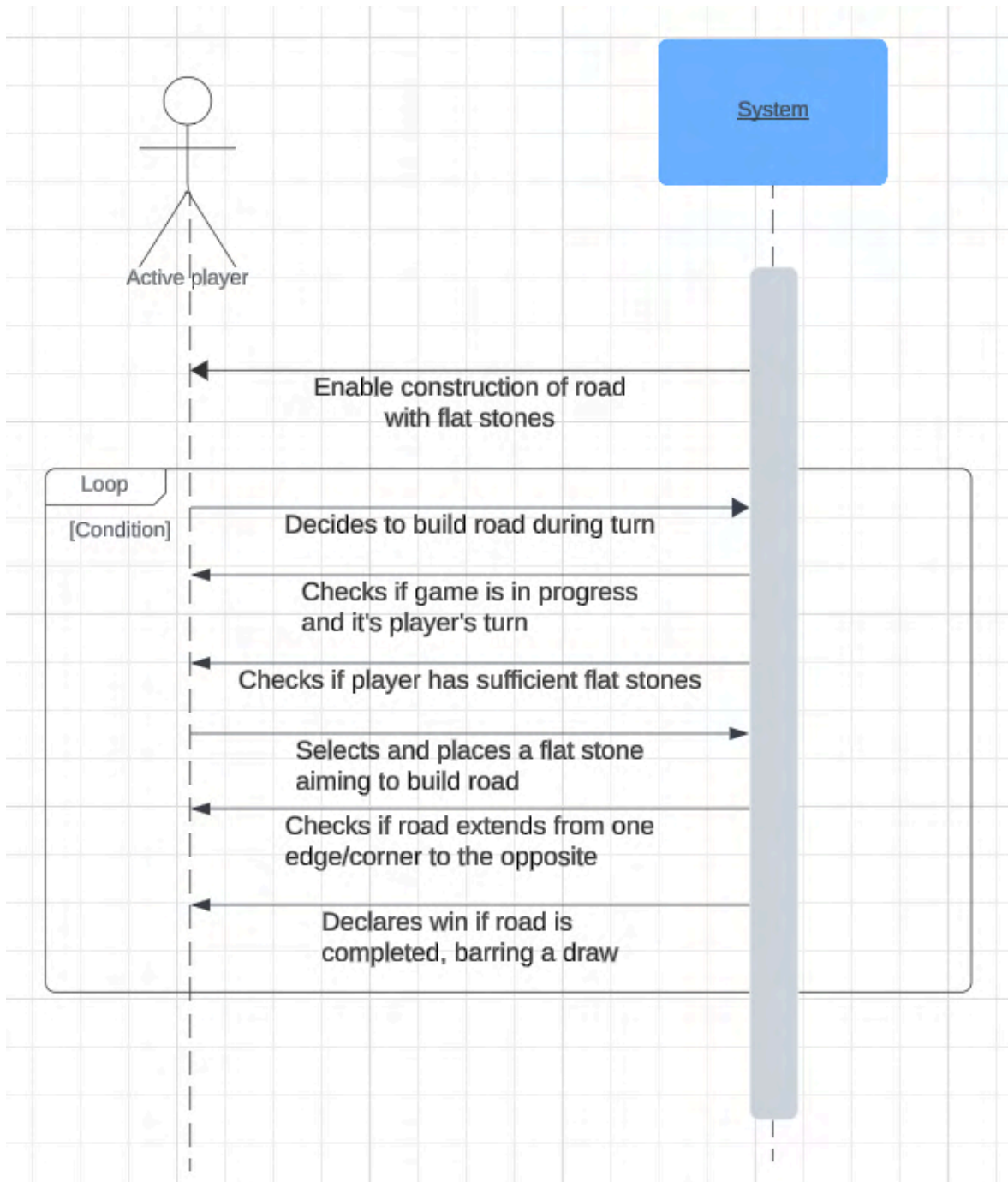


Figure 8. Sequence diagram for draw feature REQ-R4

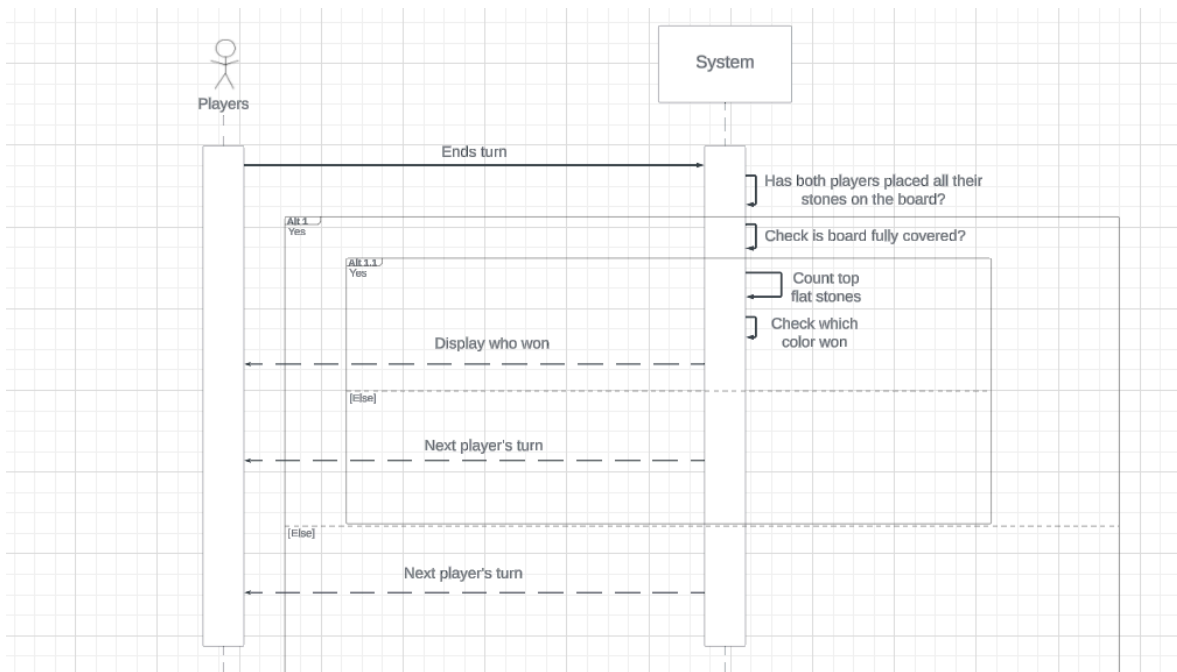


Figure 9. Sequence diagram depicting REQ-R12 - Flat win

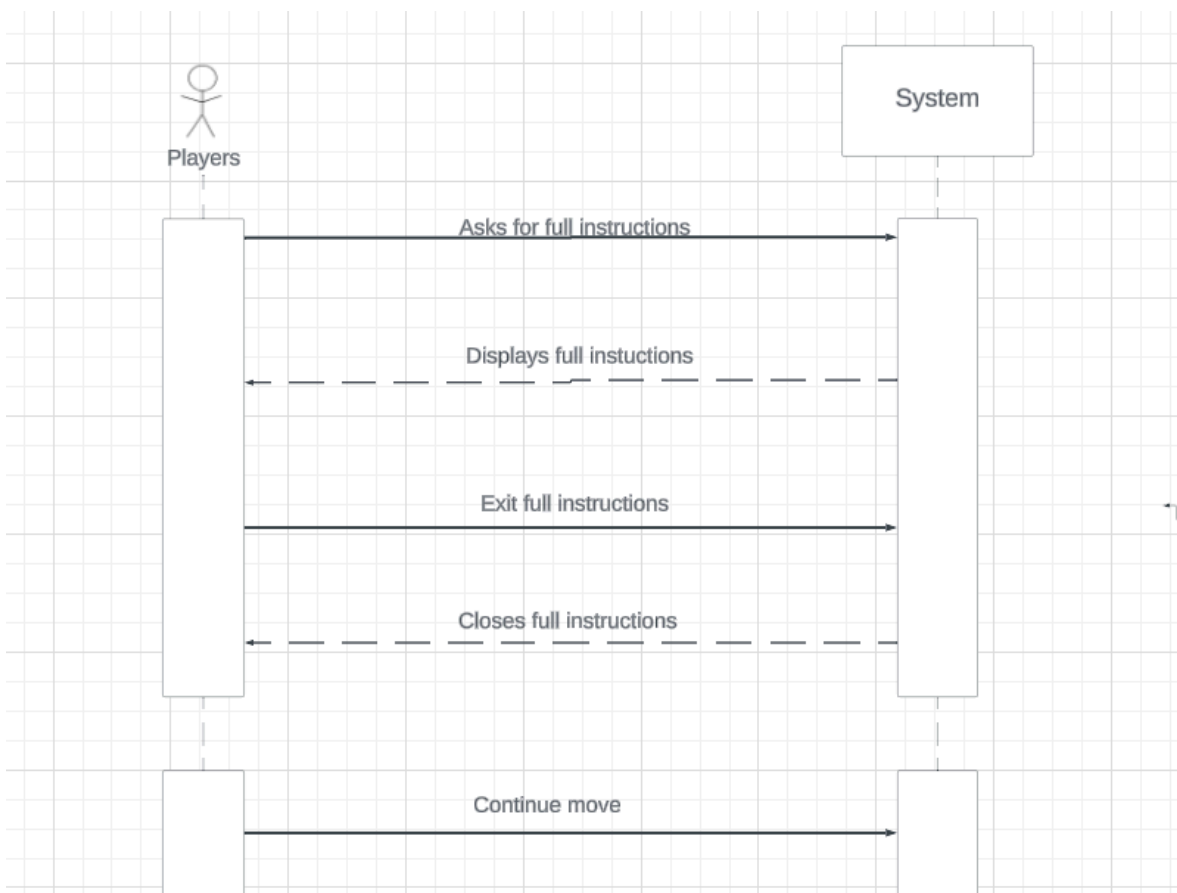


Figure 10 Sequence diagram depicting REQ-R13 - Accessing full instructions

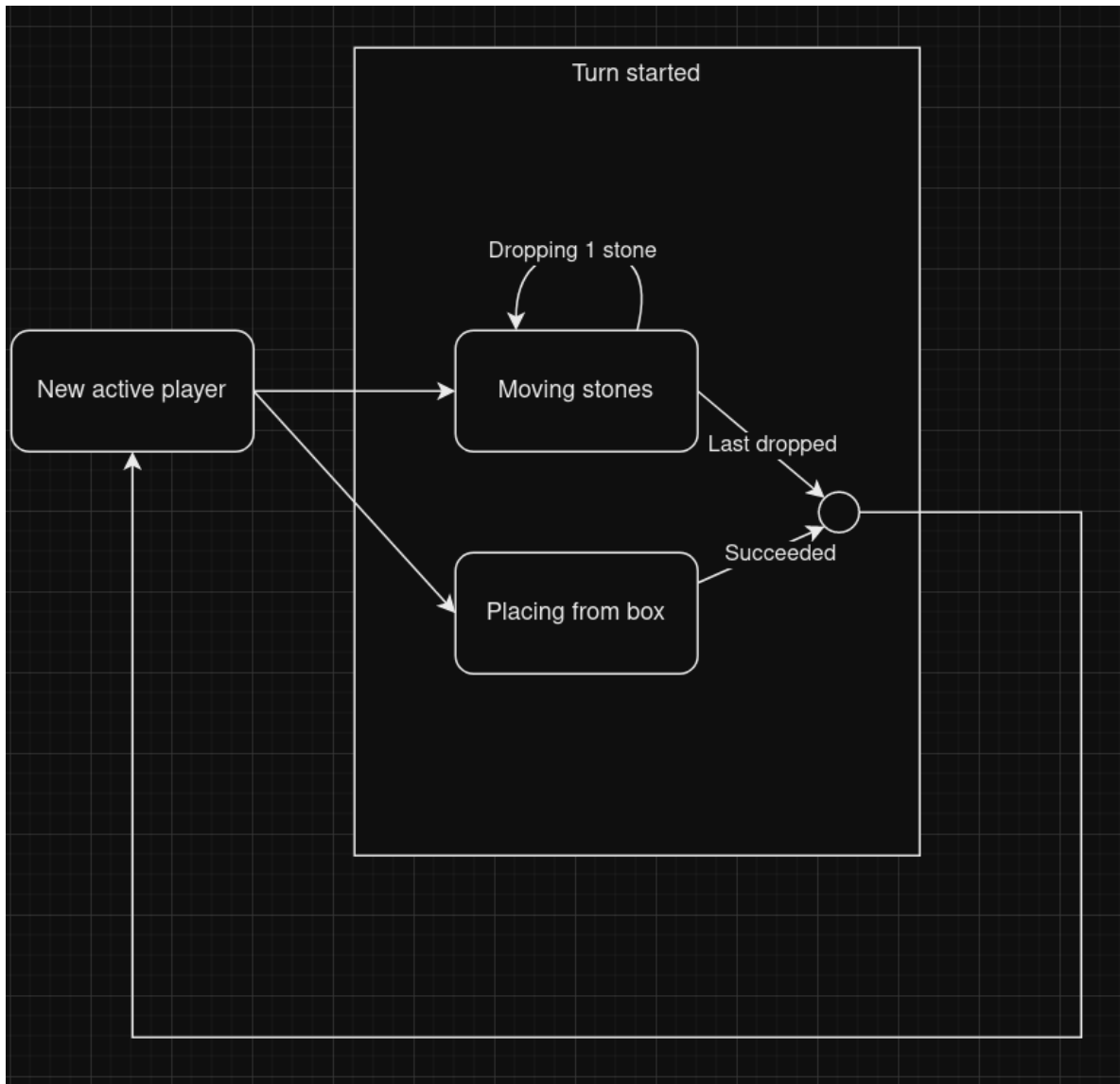


Figure 11. State diagram. The arrows going into a turn being started is a successful pick up or place, which matters because stones placed recently by the opponent remain highlighted until this happens.

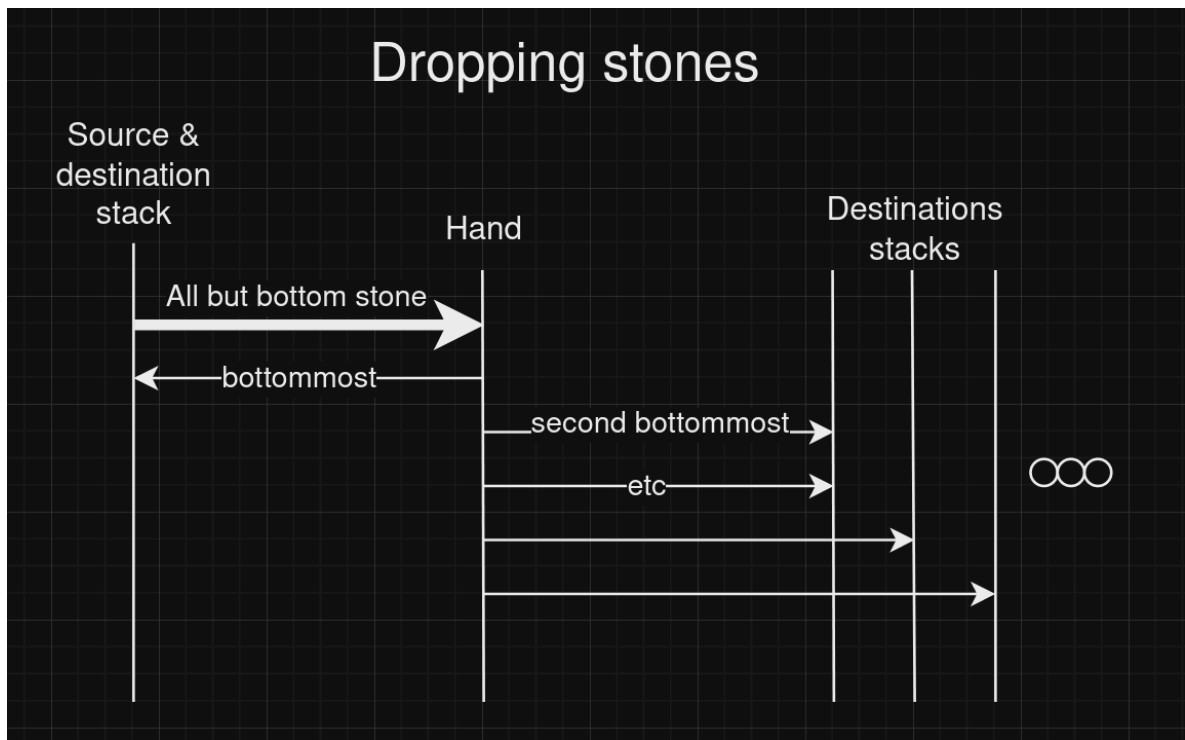


Figure 12. Example scenario of the user transferring 5 stones.

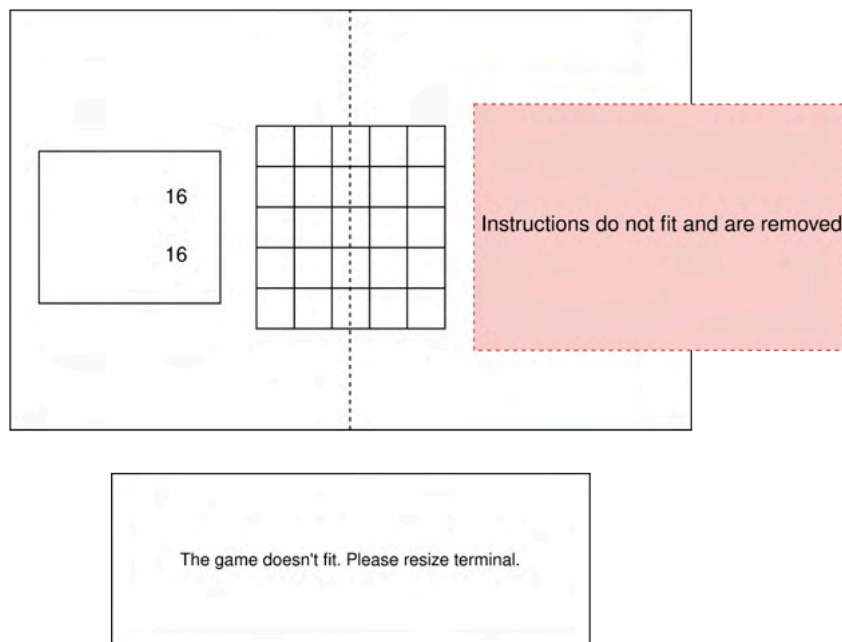
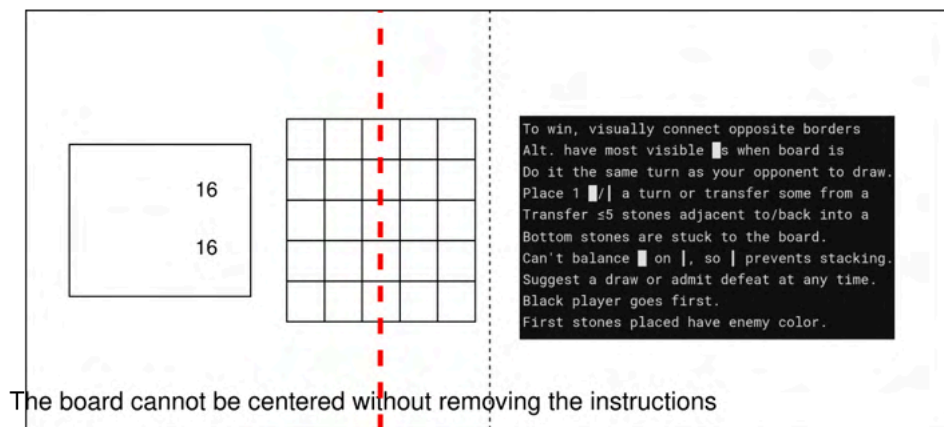
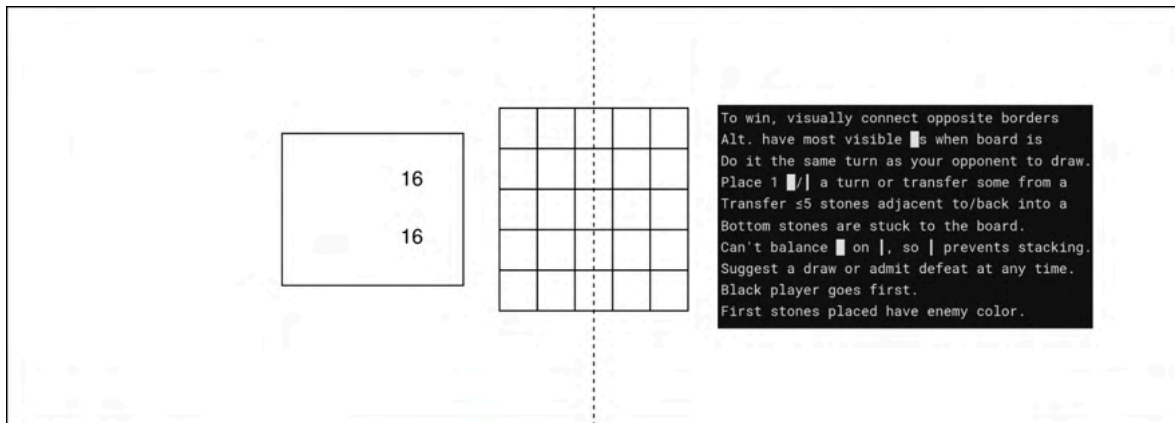


Figure 13. Layout as the terminal gets increasingly small. Does not show the button to show the full instructions, which will always be included.

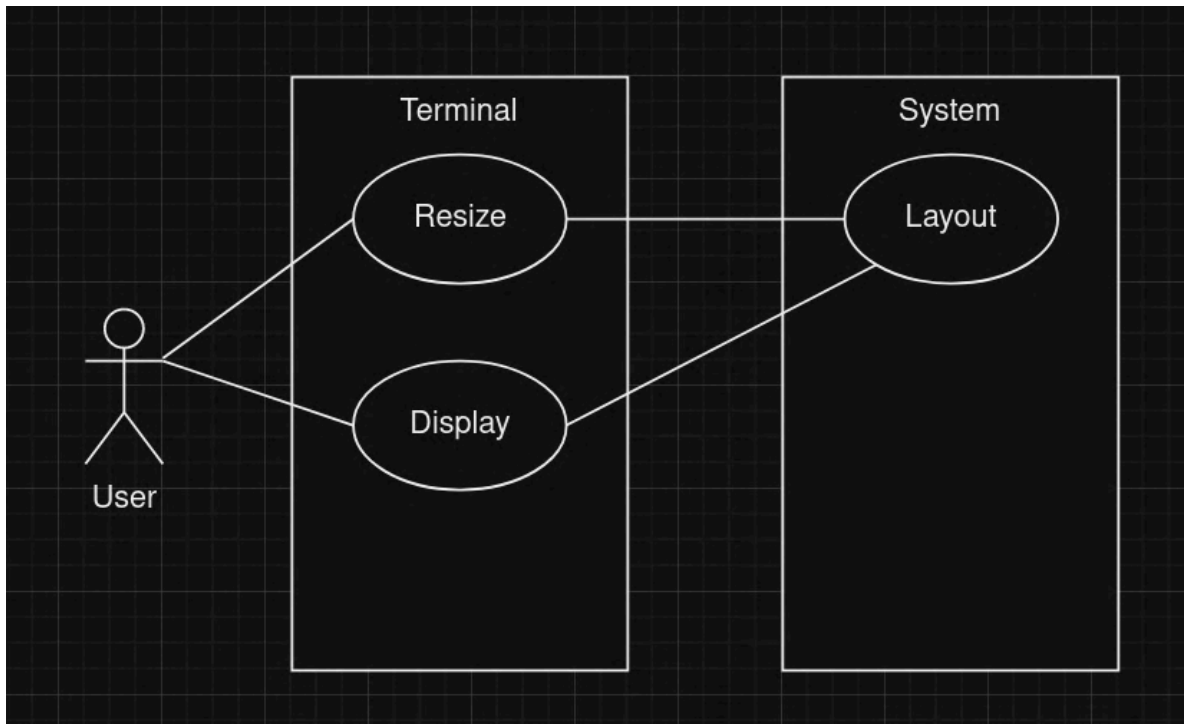


Figure 15.

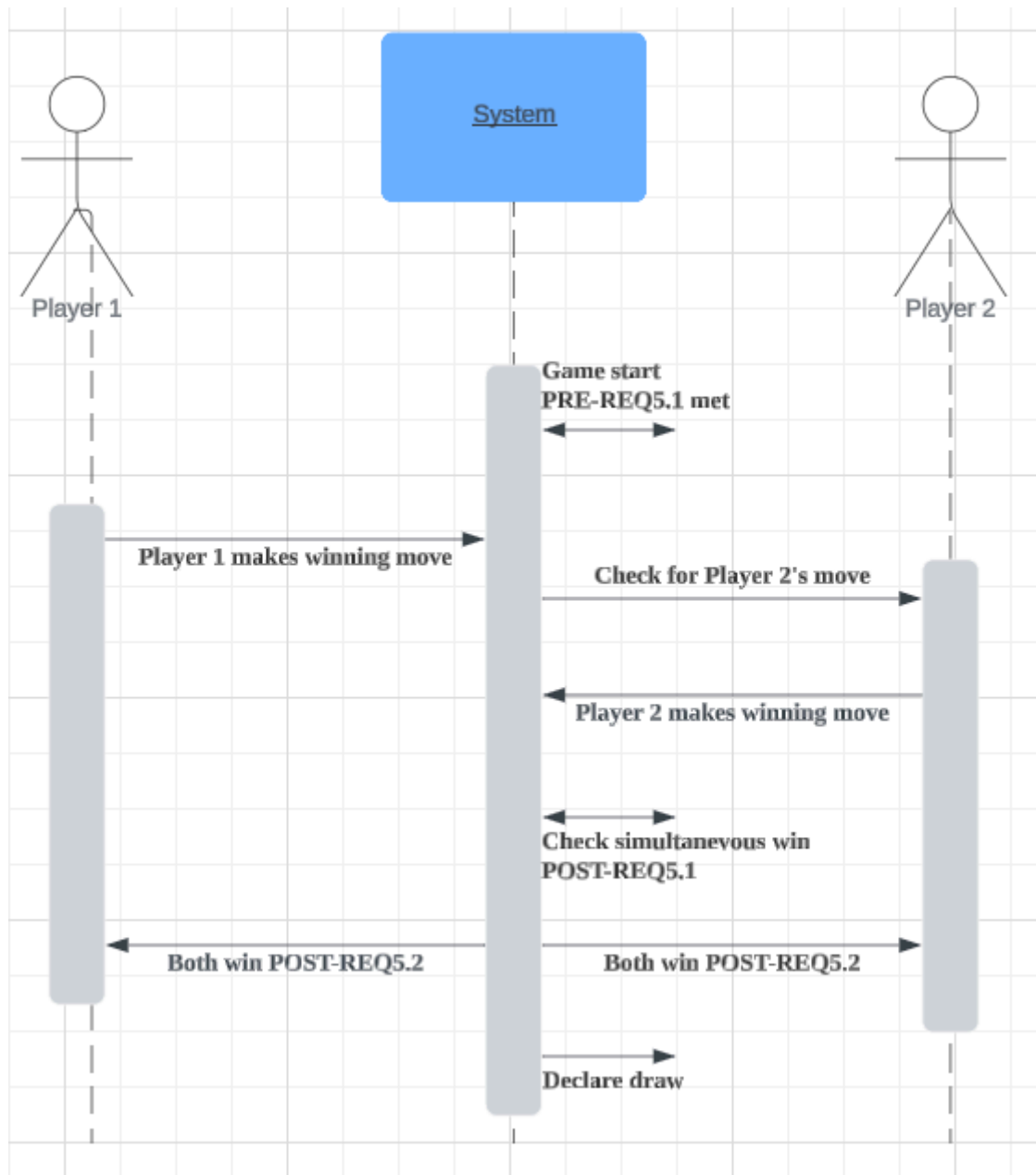


Figure 14. Sequence diagram for draw feature REQ-R5

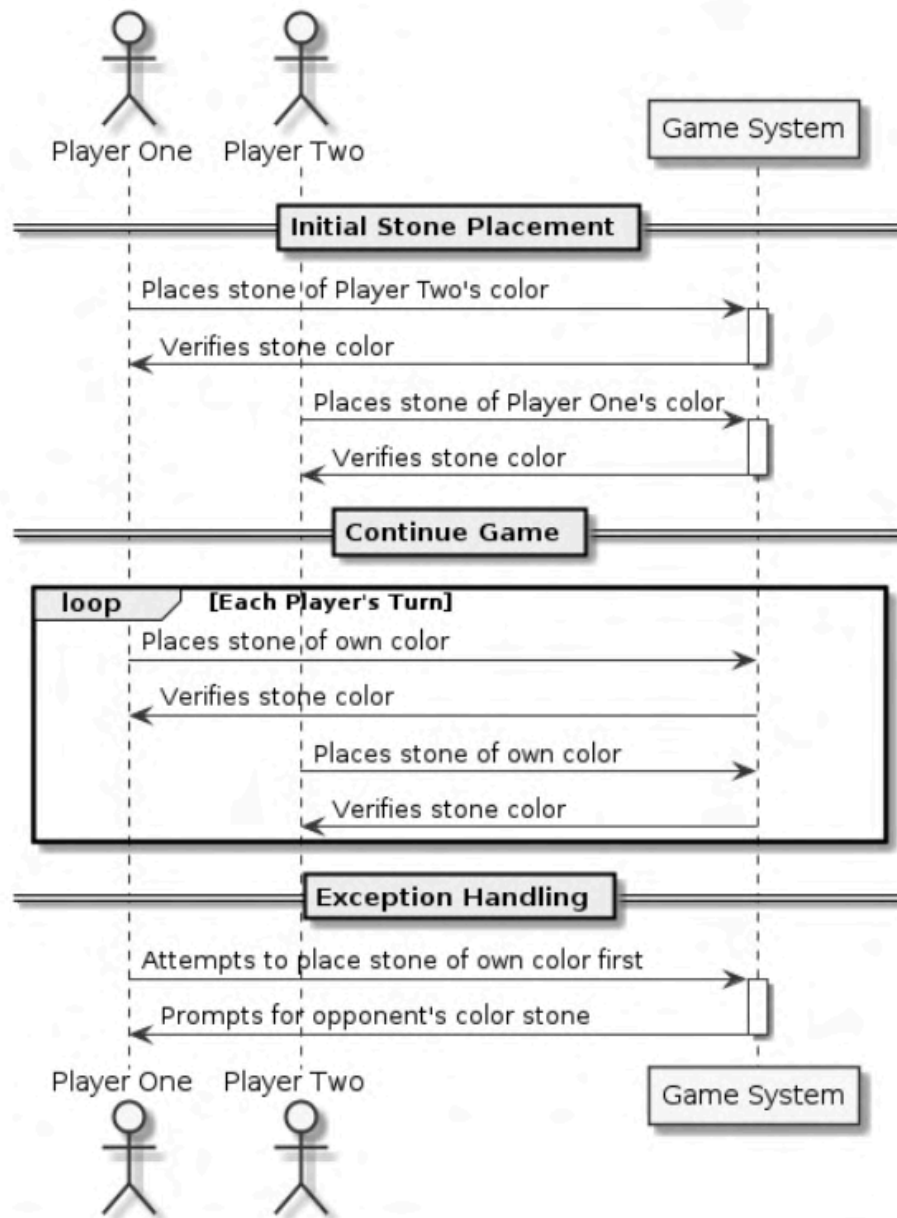


Figure 16. Sequence diagram for draw feature REQ-R7

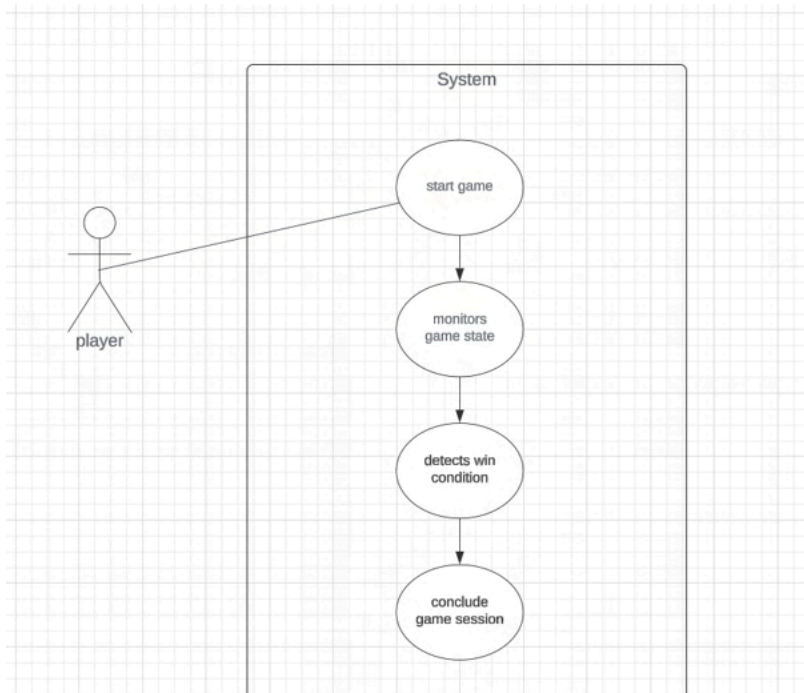


Figure 17. for REQ-10

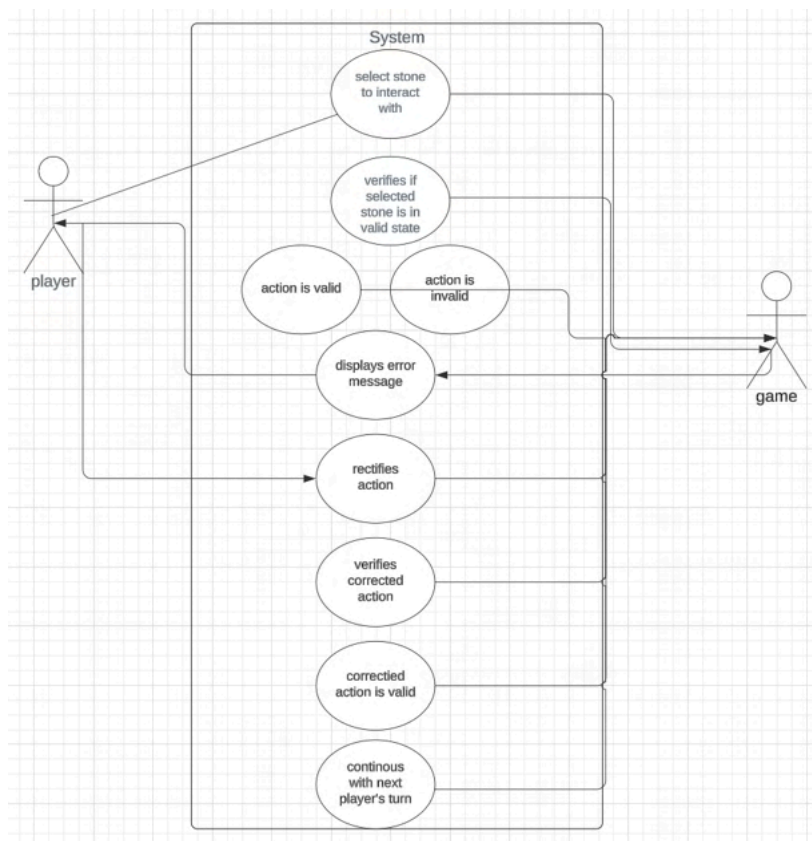


Figure 18. for REQ-11

Refined user requirements

Upon inspection of the D2 of Group C, the following user requirements have been identified for integration:

User Requirements added After Integration

1. REQ-PLAYER1. A player needs the ability to play a game against an AI able to play UUGame.
2. REQ-PLAYER2. A player needs the ability to choose the difficulty of the AI to be easy, medium or hard, so that they can play a fair game fitting their skill level.
3. REQ-PLAYER3. A player needs to be notified of when the AI is one move away from winning the game, so that they can adjust their moves accordingly.
4. REQ-PLAYER5. A player requires the AI to always return a valid move.
5. An additional user requirements is needed to allow the user to choose the game mode:
6. REQ-COOP4. As a player, I want to be able to choose the game mode, so that I can play with a friend or an AI.

Group Agreements

Contract for Code and D3 Documentation

This agreement is entered into on the 29th of February 2024, between group B, hereinafter referred to as the "Provider," and group A, hereinafter referred to as the "Recipient", collectively referred to as the "Parties".

1. Scope of Work:

1.1 The Provider agrees to deliver the code for their UU Game AI component. They also agree to provide their D3 for documentation and understanding purposes.

2. Deliverables:

2.1 The Provider shall deliver:

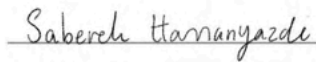
- The complete code of the AI component of their UU Game at a singular point in time as a ZIP file.
- For the contents of this ZIP file and any Git commits before and including the Feb 18 commit cc04e8d, henceforth referred to as the "Software", the Provider grants Recipient irrevocable permission to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense and/or sell copies, subject to including credit to Provider with any substantial portion of the Software and subjecting any entities granted a sublicense to these same conditions.
- D3 visualizations as per project requirements.

3. Meeting and Collaboration:

3.1 Provider graciously agrees to answer questions about the delivered AI component via email.

4. Payment:

4.1 This collaboration is undertaken on a mutual exchange basis and does not involve monetary compensation.



Sabereh Hassanyazdi
Group A



Therese Björkman
Group B

Software Integration Agreement

2024-02-23

1 Purpose

The purpose of this agreement is to set the terms and conditions that apply as Group A and Group C communicate and give access to the code of the game AI and/or platform for the board game UU Game.

2 Permission

1. Group A and Group C grant each other irrevocable permission to deal in the current version of their software and any related documentation or files without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute and/or sell copies of it and them, subject to the terms and conditions within this document.
2. Group C also furthermore grants to Group A the above permission to any future versions, which they also agree to make accessible to Group A. Furthermore, Group C must, at any future request of Group A, publish any version of their software or related files under the MIT or GPL license.
3. Group A grants the permission in 2.1 to future versions 1 week after they made accessible if they have not expressed anything to the contrary. During this one week period a more limited permission is granted allowing unrestricted internal development and communicating and exhibiting copies to the client and teachers.

3 Communication

1. Both groups agree to maintain an open line of communication throughout the integration process. Group A agrees to make some effort to support and to answer at least some questions from Group C regarding their software for the purpose facilitating the successful integration of the platform and the AI. Group C agrees to make a reasonable effort to answer any question regarding their project.
2. Group C agrees to, as soon as reasonably possible, communicate if they wish to have certain features that Group A has planned to implement completed by a certain date. An example of such a feature is the scoreboard, which may help Group C with things like selection of difficulty and which player(s) are controlled by the AI. Other examples include the adjacency requirement or a real acknowledgment that the game has ended.

4 Credit and Acknowledgment

Group C agrees to credit Group A for the development of the original Software by prominent display to all users of the application in addition to including it in the code repository in a customary way.

5 Signatures

By signing this document, Annika Elvers as Group C's communicator, and Isak Hvarfner as Group A's project manager testify that their respective groups have decided, collectively or through other way of governance, that they agree and promise to uphold this agreement.

Signed Document

via <https://min.ebox.nu>



Software Integration Agreement

**Signed
text:**

Jag har läst och förstår innehållet i PDF-filen (1) och samtycker och godkänner allt som avtalas däri, som om jag skrivit under en fysisk utskrift av PDF-filen (1):

(1) Avser PDF-fil enligt följande

Namn: Software Integration Agreement.pdf

Storlek: 57341 byte

Hashvärde SHA256:

7803d762ff0df0b8b86917638c7b88b31394cd076dc2bb9341c805c3f57c9474

The original file and all signatures are attached to this PDF.

To open the attachments, a dedicated PDF reader may be required.

Signed By 2:

ANNIKA KRISTIN ELVERS

Signed with BankID 2024-02-23 19:12 Ref: 6a6e96b4-6c74-4edb-b56b-30fed487cf70

ISAK HVARFNER

Signed with BankID 2024-02-23 16:46 Ref: 5dbc77ca-bb77-4acc-ab61-4a4b16a4ddb8

AKE, IH – Powered by TellusTalk

Agreement honoring

Re: Question regarding your component

Isak Hvarfner <isak.hvarfner.7355@student.uu.se>

Thu 2024-02-29 18:34

Vilma van der Schoot <vdschoot@hotmail.se>

Muhammad Farid Retistiano <muhammad-farid.retistiano.2872@student.uu.se>; Vilma van der Schoot <vilma.van-der-schoot.5447@student.uu.se>; Zakarie Warsame <zakarie.warsame.2759@student.uu.se>; Annika Kristin Elvers <annika-kristin.elvers.7417@student.uu.se>

Sorry for needing to send a second message.

I want to clarify that it is always the bottom stone that's dropped from the hand, and that's what the "semi-transparent" cell under the hand means. This seems the most natural if you imagine playing the game with a single hand in the air that's picking up and dropping coins.

Second, there's a negation missing in the "1. ". The point is that the color of the bottom stone doesn't have an effect in our version.

Third, I want to preempt questions about being able to move stones anywhere and not just adjacently -- we will add this very soon. Let us know if there's any feature you would like us to prioritize and a desired date.

Kind regards,

Isak Hvarfner

Main Developer / Product Owner

Group A

From: Isak Hvarfner <isak.hvarfner.7355@student.uu.se>

Sent: Thursday, February 29, 2024 17:54

To: Vilma van der Schoot <vdschoot@hotmail.se>

Cc: Muhammad Farid Retistiano <muhammad-farid.retistiano.2872@student.uu.se>; Vilma van der Schoot <vilma.van-der-schoot.5447@student.uu.se>; Zakarie Warsame <zakarie.warsame.2759@student.uu.se>; Annika Kristin Elvers <annika-kristin.elvers.7417@student.uu.se>

Subject: Re: Question regarding your component

Hello,

I am responsible. Informing me would be much appreciated.

Regarding the thin black stripes: This is partially intentional. Before we used an empty or filled circle ("○" and "●") and it was much clearer what that meant, with the complete outline serving as an indication that there's something there even though it's the same color. The reason we changed it was so that it was believable that the same type of stone could be rotated to be flat or standing. This rotation requirement has since changed for us, so we will go back to how it was before.

For these next issues, I hope you have tested them on our main branch after Feb 21.

Regarding the white player being able to move pieces when it's the black player's turn:

1. If you are referring to being able to move a stack of stones no matter the color of the bottom stone, this is not in our requirements.
2. If you are referring to being able to click anywhere in the box to select which pose/shape to place, this is intentional, as it means you don't have to have great "aim". When we recently watched the client play the prototype, we noticed that it was easy to miss who the turn indicator arrow was pointing to or what it meant, so we will make it larger, and we will also add text instructions explaining it. We could perhaps add a border between the different player stone counts.

Regarding the stack supposedly being upside down: I don't understand what you mean. The board view is top-down, and gravity is down in the hand just like the down is on the screen. If you do "2, space, 2, 2, 1. Space, left arrow, space" you will place down as if you dropped them towards the screen: white, black, black, standing white. All but the bottom white is then picked up and one of the black ones is dropped.

I will let you know of fixes and improvements at irregular intervals, so it might be quicker to watch the Git log.

Kind regards,

Isak Hvarfner

Main Developer / Product Owner

Group A

From: Vilma van der Schoot <vdschoot@hotmail.se>

Sent: Thursday, February 29, 2024 16:57

To: Isak Hvarfner <isak.hvarfner.7355@student.uu.se>

Cc: Muhammad Farid Retistiano <muhammad-farid.retistiano.2872@student.uu.se>; Vilma van der Schoot <vilma.van-der-schoot.5447@student.uu.se>; Zakarie Warsame <zakarie.warsame.2759@student.uu.se>; Annika Kristin Elvers <annika-kristin.elvers.7417@student.uu.se>

Subject: Question regarding your component

Hi!

We have found some issues with the code that we would like to inform you about. We were also wondering about who is responsible for fixing these issues? Maybe the best approach would be if we inform you every time we find a bug in your code, and that you then fix the bug, and maybe vice versa if you find bugs in our code? Also we would like to be informed if you do any other upgrades or bug fixes of your component, so that we can import it into our work.

These are the things we have found so far:

- The white player can move pieces when it's the black player's turn, it's not blocked from making moves.
- The stack is upside down which is confusing.
- When a standing white stone is added to a stack of white stones, it turns into two thin black stripes, which seems to be a visual error

Kind regards,

Vilma and group C

Changelog

Sorry, we didn't have time for this. See the Google Docs history in the links for D2 and D3 we sent you.