FIT3077 Software Engineering: Architecture and Design

Sprint 1 Specifications

Team Members:
Anson Wong Sie Yuan
Ethan Too
Teh Rong Wen
Khoo Kai Neng

Team Information

Team Name:	Code Corgi
Team Photo:	

Team membership:

Name:	Anson Wong Sie Yuan
Contact Details:	Email: awon0048@student.monash.edu
Basic Information:	Age: 21 From: Sarawak Specialisation: Software Engineer Current Semester: Year 2 Semester 2 Hobbies: Playing Badminton
List of Technical Strength	● Java ● Python ● Matlab
Fun fact:	● Likes anime ● Plays chinese yoyo

Name:	Ethan Too
Contact Details:	Email: etoo0004@student.monash.edu
Basic Information:	Age: 22 Specialisation: Advance Computer Science Current Semester: Year 3 Semester 1
List of Technical Strength	● Java ● Python ● HTML + CSS
Fun fact:	● Likes games ● Likes anime ● Plays the piano ● Plays guitar

Name:	Teh Rong Wen
Contact Details:	Email: rteh0003@student.monash.edu
Basic Information:	Age: 23 Current Semester: Year 3 Semester 2 Specialisation: Data Science Hobbies: Playing Badminton
List of Technical Strength	● Java ● Python
Fun fact:	● Cat ● Dog ● Mouse ● Gamer ● Badminton

Name:	Khoo Kai Neng
Contact Details:	Email: kkho0020@student.monash.edu
Basic Information:	Age: 20 Specialisation: Software Engineer Current Semester: Year 2 Semester 2
List of Technical Strength	● Java ● Python
Fun fact:	● Games ● Anime ● Plays the violin

Team Schedule:

Team Weekly Meeting Schedule:	Tuesday 9pm
Team Meeting Location:	Discord / Zoom
Team Weekly Work Schedule	Friday: 9.30pm - 11.30pm Saturday: 9.00pm - 11.30pm
Workload Distribution:	● The low-fi prototype drawings are distributed according to each phase of the 9 Man Morris game. ● The user stories are divided evenly among the team members and each member will create a partial domain model which will then combine with other members' domain analysis.

Tech Stack:

Programming Language to be used:

- Java
    - A language that focuses mainly on OOP (Object-Oriented Programming)
    - All team members are experienced with the language
    - Easy to use
    - A platform-independent language
    - Able to reuse some of the previous code through inheritance.

APIs to be considered for the assignment:

- JavaFX
    - Can provide platform support for creating desktop applications
    - Have better and modern UI components
    - Have rich toolkits
- Swing
    - Can be used to create GUI
    - No new toolkits to be used

Other Technologies to be used:

- IntelliJ
    - Offers adaptable libraries
    - Provides varied setup, a consistent way to configure everything
    - Provides convenience for testing
- GitLab
    - All team members are familiar with git
    - Allows all team member to collaborate and create codes
- LucidChart
    - Able to draw class diagrams
    - Free to use
    - User-friendly for sketching diagrams
    - All team members have experience in using it

Alternatives for Tech Stack:

- Python
  - Supports OOP principles
  - Have more concise syntax
  - Have a rich set of framework and libraries

- ➔ Our team choose Java over Python because Java in mainly focusing on Object-Oriented Programming (OOP) which is one of the main requirements for this assignment
- ➔ Our team are more experienced in using Java to program OOP concept rather than in Python

- Diagram.net
  - Free to use
  - Friendly to new users
  - Only for web-based platforms

- ➔ Our team choose to use Lucidchart over diagram.net mainly because all team members are more comfortable with using Lucidchart
- ➔ Lucidchart also provides more features to create diagrams as compared to diagram.net.

- VScode (Visual Studio Code)
  - Allows cross platform supports
  - Able to add extensions and plug-ins
  - Supports multiple language

- ➔ Since our main programming language is Java, our team has chosen IntelliJ as our main IDE, as IntelliJ mainly focuses on Java based programming.
- ➔ VScode is less reliable, responsive and stable as compared to IntelliJ
- ➔ Although VScode supports multiple languages but it may not provides the same amount of support to Java as compared to IntelliJ

Advanced Requirements Chosen

A. Considering that visitors to the student talent exhibition may not necessarily be familiar with 9MM, a tutorial mode needs to be added to the game. Additionally, when playing a match, there should be an option for each player to toggle “hints” that show all of the legal moves the player may make as their next move.

C. A single player may play against the computer, where the computer will randomly play a move among all of the currently valid moves for the computer, or any other set of heuristics of your choice.

User Stories

Basic Requirements:

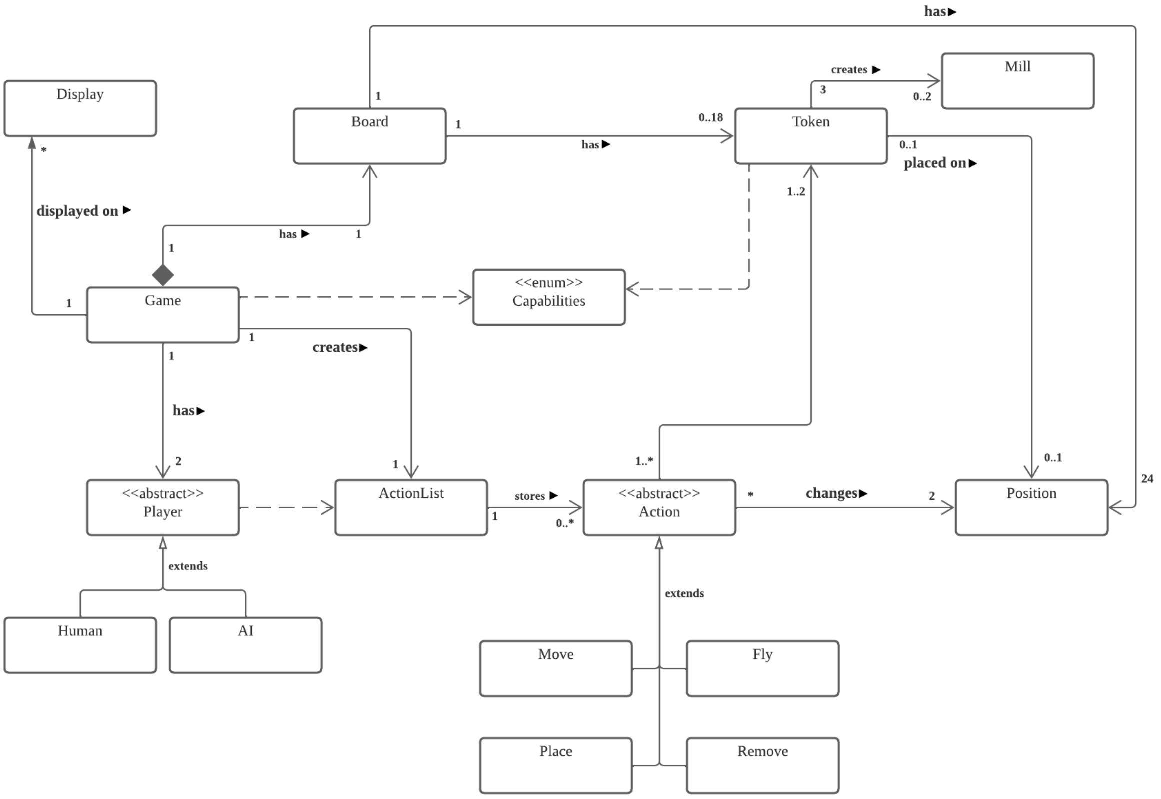
1. As a new player, I want to be able to learn the rules of 9MM before the game, so that I can strategize while playing the game.
2. As a player, I want to be able to take turns with another player within the same game client instance, so that we can play the game together.
3. As a player, I want to be able to see the current state of the game board, so that I can make informed decisions about my moves.
4. As a player, I want to be able to make my move by selecting a token then selecting a valid destination for that token, so that I can play the game according to the standard rules.
5. As a player, I want to be able to end the game and declare a winner, so that I can finish the game and see who won.
6. As a player, I want to be able to see the amount of tokens my opponent has during the placing phase of the game, so that I can plan my next move accordingly.
7. As a player, I want to be able to add a token onto the board, so that I can start the game
8. As a player, I want to be able to remove a token from the board, so that I can win the game
9. As a player, I want to be able to make a mill when three tokens are aligned in one line, so that I can remove one of the tokens from my opponent.

10. As a player, I want to be able to ‘fly’ my tokens anywhere on the board when I have three tokens left on the board, so that I can win against my opponent.
11. As a player, I want a board to be displayed on the screen when the game starts, so that I can play the game.
12. As a developer, I want to have a pop out box when the game ends, so that I can show the winner of the game.
13. As a developer, I want to be able to display the turn of a player, so that the players are aware on who’s turn to make their move

Advanced Requirements:

1. As a new player who is not familiar with 9MM, I want to be able to access a tutorial mode that provides guidance for the game, so that I can learn how to play the game.
2. As a new player, I want to be able to toggle hints that show all of the legal moves I can make as my next move, so that I can see my options and make informed decisions.
3. As a player, I want to be able to play against a computer opponent that randomly selects a move, so that I can play the game when I don't have another player available.

Basic Architecture:



Design Rationale:

- **Display** - The Display domain is responsible for showing the current state of the game to the player. It is important to keep the player informed about the progress of the game and their current situation so that they can make informed decisions. The relationship between Display and Game is that Display checks the state of the Game class and updates the display for the player every round.
- **Game** - The Game class is the main class of the game. It is responsible for checking the board and token, then updating the game states. The Game class is also in charge of updating each Token's capabilities depending on the phase of the game. The relationship between Game and Token is that Game updates the Token's capabilities.
- **Board** - The Board class is responsible for storing all the positions and tokens. Within every position, there can contain a token. The relationship between Board and Token is that Board stores the Token objects.

- **Mill** - The Mill class is responsible for tracking whether a Token is part of a Mill, which is when 3 Tokens are lined up. The relationship between Mill and Token is that Mill tracks the Tokens that are part of a Mill.
- **Token** - The Token domain is an object that the player or AI is able to interact with in order to play the game. The relationship between Token and Board is that Board stores the Token objects. The relationship between Token and Mill is that Mill tracks the Tokens that are part of a Mill. The relationship between Token and Game is that Game updates the Token's capabilities based on the phase of the game.
- **Position** - The Position class represents each of the 24 positions on the board. The relationship between Position and Board is that Board stores the Position objects.
- **Player (*abstract*)** - The Player class is an abstract class representing each of the two players playing the game. It was initially planned to be a regular class with a playerType indicating the type of player, though it is designed as an abstract class to adhere to the Open-Closed Principle as modification to the type of AI (easy, normal, hard) can be further implemented in the future if necessary. The relationship between Player and Human/AI is that Human and AI are subclasses of Player.
- **Human** - The Human class is a subclass of Player and it represents a player who makes moves by manually clicking on the board. The relationship between Human and Player is that Human is a subclass of Player.
- **AI** - The AI class is a subclass of Player which represents a “player“ that randomly makes a move based on the possible legal moves available. The relationship between AI and Player is that AI is a subclass of Player.
- **ActionList** - The ActionList class is a list containing all possible actions of a player. This class also includes a “hints” function to display all possible moves each token can make to the current player. The class can also be used by the AI to randomly select their next move. The relationship between ActionList and Player is that ActionList is associated with a Player object.
- **Action (*abstract*)** - The Action class is an abstract class that can be extended to create sub-classes, as each of the Action contains the same method but different implementation. This adheres to the Open-close principle as modification can be done without modifying any other classes. The relationship between Action and its subclasses (Move, Fly, Place, Remove) is that they are all subclasses of Action. Furthermore, Action also adheres to the Single-Responsibility Principle

as all actions are split up into different subclasses instead of one Action class being in charge of all actions.

- **Move** - The Move class is a subclass of Action which enables the player or AI to move the tokens on the board when all tokens are placed on the board. The relationship between Move and Action is that Move is a subclass of Action.
- **Fly** - The Fly class represents an action that consists of the implementation for the tokens to move to any position on the board when there are only three tokens left. Its relationship with the Action domain is that it is a subclass of Action.
- **Place** - The Place class represents an action that allows the Player or AI to place a token onto the board. Its relationship with the Action domain is that it is a subclass of Action.
- **Remove** - This Remove class represents an action that allows the Player or AI to remove a token from the board. Its relationship with the Action domain is that it is a subclass of Action.
- **Capabilities (*Enum*)** - This Capabilities class represents the different capabilities of the token, and each capability allows each of the tokens to carry out specific actions. Some of the capabilities we currently have for the tokens are flyable and movable. Its relationship with the Token domain is that it is assigned to the Token objects by the Game class depending on the phase of the Game.

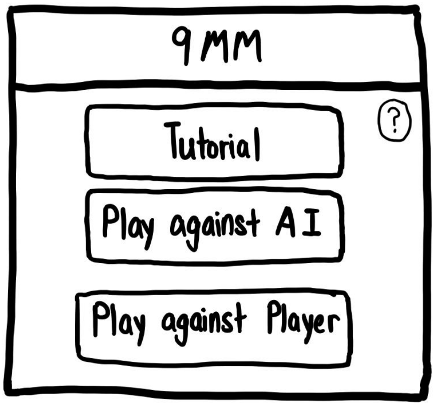
Assumptions Made

- The assumption that has been made by the team is that for the advanced requirements, we assume that a slideshow will be demonstrated to the player whenever the player decides to click on the tutorial button on the homepage.

Basic UI Design

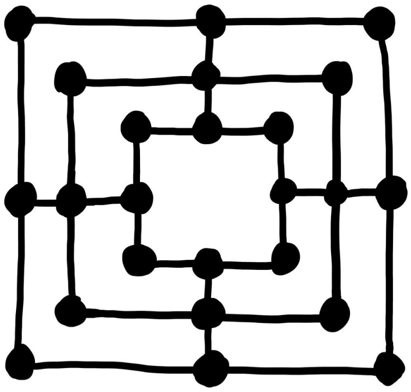
Lo-Fi Prototype

Homepage:



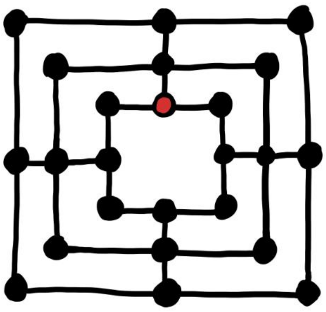
Frame 1: The homepage displays a few options to the player, whether to play against an AI or another player. There is also an option to enter the tutorial mode for a new player to learn the game.

Initial board:

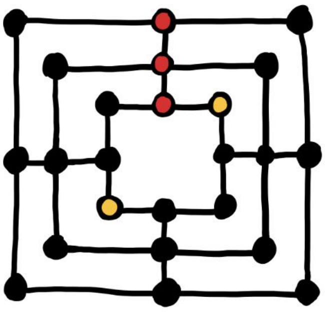


Frame 2: The board will be displayed whenever the player starts the game.

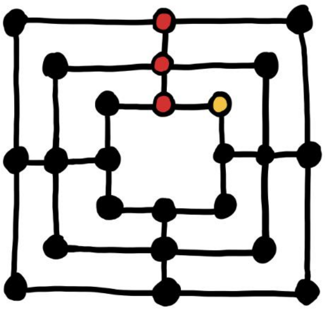
Placing phase:



Frame 3: A token is added on the the board at the start of the game

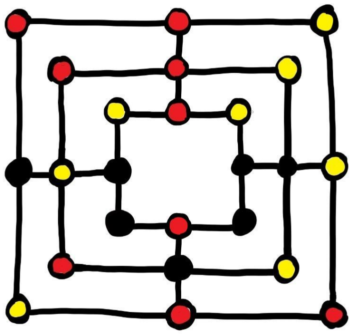


Frame 4: A mill is formed on the board when three tokens of the same colour is aligned on a line

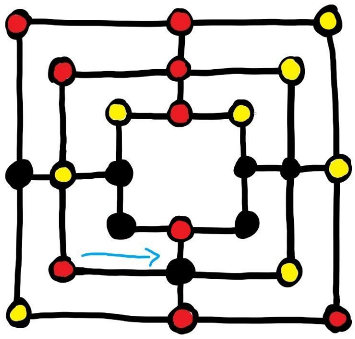


Frame 5: One of the opponent’s token is removed from the board

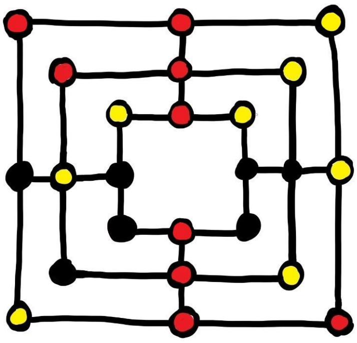
Moving Phase:



Frame 6: When both players have placed all nine tokens, the game enters the moving phase.

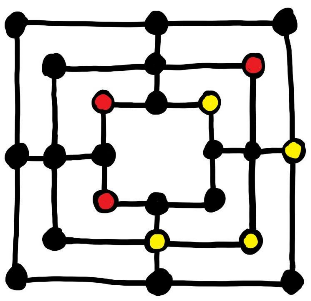


Frame 7: During the moving phase, players move their own token to adjacent position.

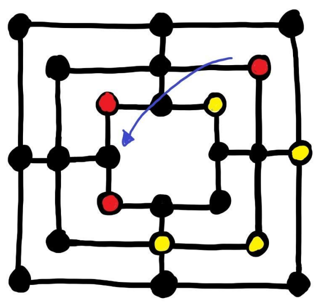


Frame 8: To win, players must continue aiming to form mills and reduce their opponent's tokens.

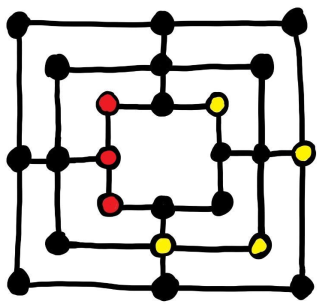
“Flying” Phase:



Frame 9a



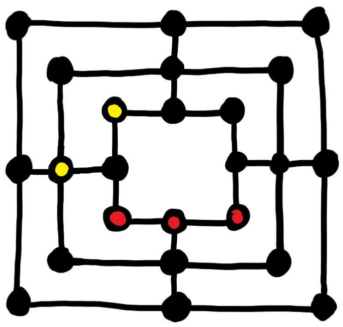
Frame 9b



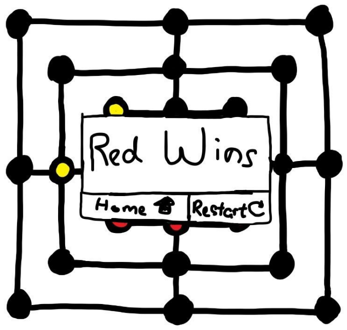
Frame 9c

Frame 9a, 9b, 9c: When a player is left with 3 tokens, the player can move one token to any point on the board during their turn.

End Phase:



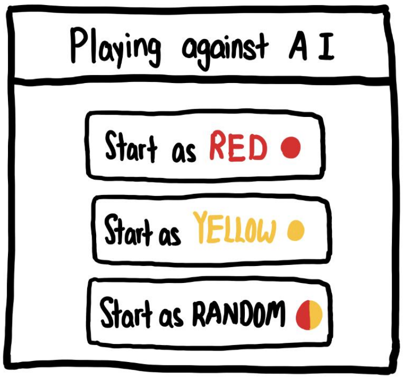
Frame 10a



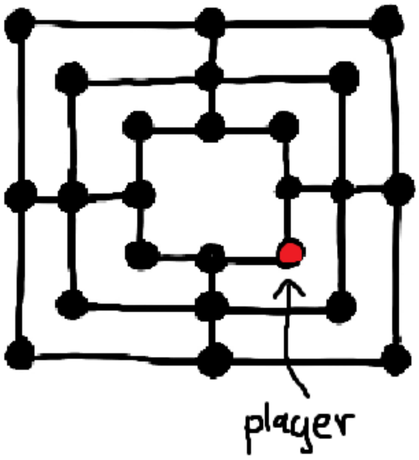
Frame 10b

Frame 10a, 10b: A player wins when the opponent either runs out of legal moves, or is left with 2 tokens. When this happens, a pop up box will appear letting the users know which player has won. Additionally, the pop up will display a home button and a restart button.

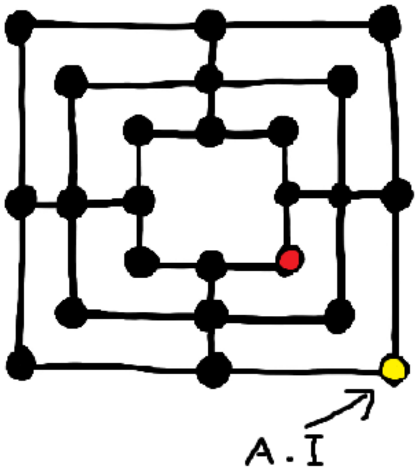
Advanced Requirements:



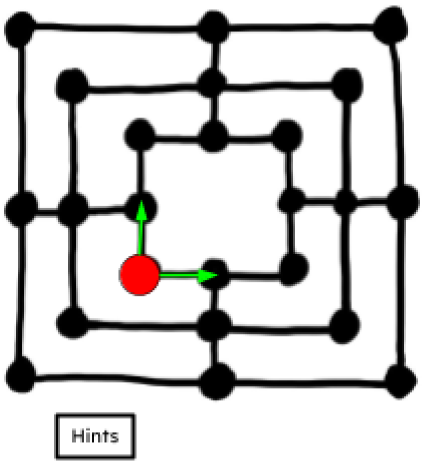
Frame 11: If the player chose the option to play against AI, the player will be directed to another page to choose which colour to start as.



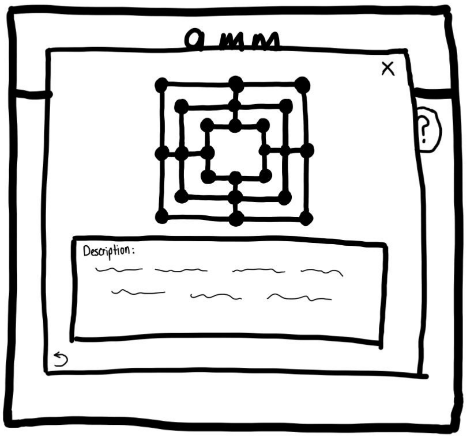
Frame 12: If you start as red, you will place a token first.



Frame 13: Token will be placed randomly by the AI after your turn, and the game continues…



Frame 14: Clicking the “hints” button will cause the game to show the player all current possible moves



Frame 15: The tutorial of the game presents a series of situations to the player to show them how to play the game in each situation