

SPRINT ONE

FIT3077 Software Engineering:

Architecture & Design

Table of Contents

[**Team Information** 2](#_Toc131370693)

[Team Name & Team Photo 2](#_Toc131370694)

[Team Member 2](#_Toc131370695)

[Team Schedule 3](#_Toc131370696)

[Technology Stack & Justification 4](#_Toc131370697)

[**User Stories** 6](#_Toc131370698)

[Basic Requirements 6](#_Toc131370699)

[Advanced Requirements 7](#_Toc131370700)

[**Basic Architecture** 8](#_Toc131370701)

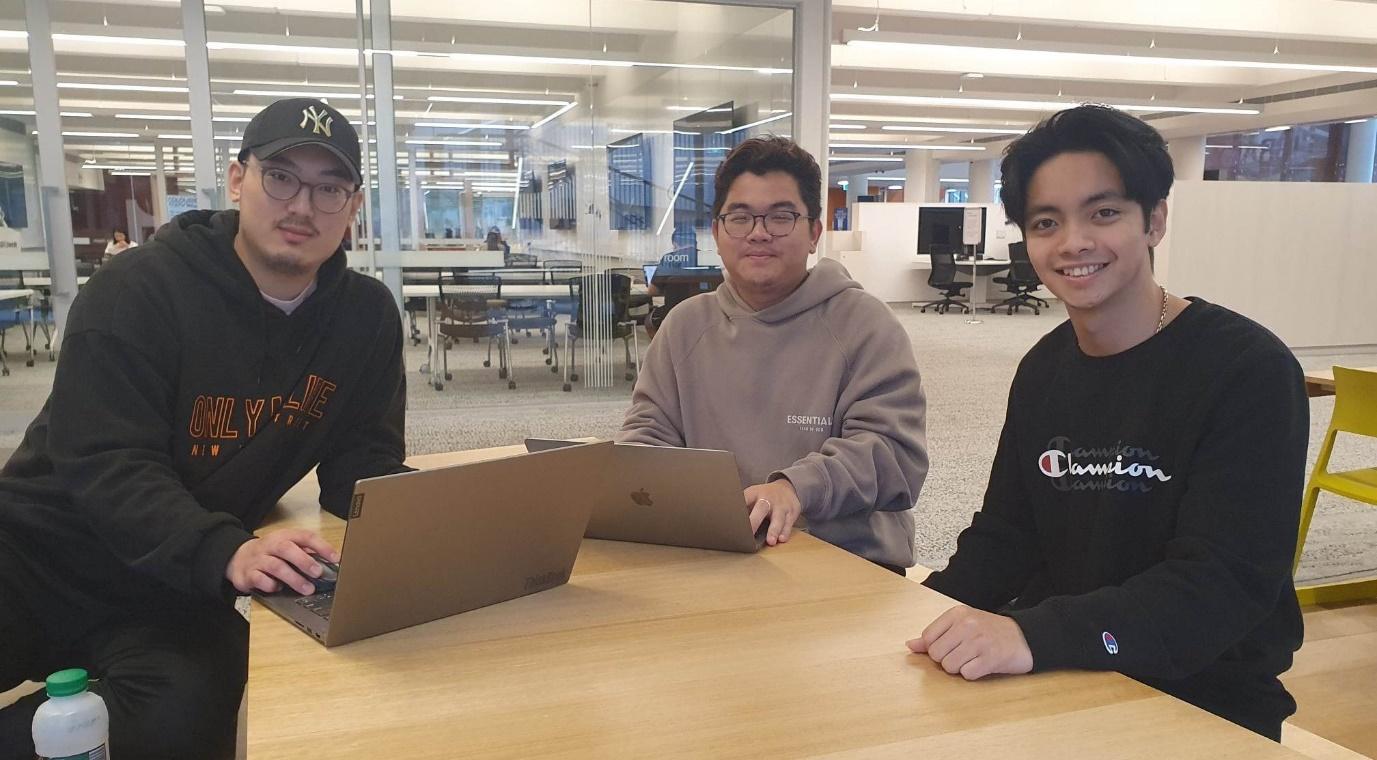
[**Basic UI Design** 10](#_Toc131370702)

[**Reference List** 11](#_Toc131370703)

# **Team Information**

## Team Name & Team Photo

**Optimistics**



## Team Member

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Email Address** | **Technical Strength** | **Professional Skill** | **Fun Fact** |
| Deokkyun Kil | dkil0003@student.monash.edu | Java, Python, C++, CakePHP | Teamwork, Creative and strategic thinking | Korean,  Working in Korean restaurant |
| Chandra Putra | cput0004@student.monash.edu | Java, HTML, CakePHP | Cooperation & Collaboration, Real-life problem solving | Listens to Lana Del Rey religiously |
| Hong Kien Nguyen | hngu0053@student.monash.edu | Java, C++, CakePHP, Html, Javascript | Communication, Collaboration, Time Management | Can not handle spicy food |

## Team Schedule

**Regular meeting schedule**

|  |  |
| --- | --- |
| Online Meeting (Zoom) | 9.00pm Tuesdays |
| Face-to-face Meeting (On campus) | 6.00pm Thursday |

**Regular work schedule**

|  |  |
| --- | --- |
| Chandra | Tuesday: 2-7pm Wednesday: 11-7pm Friday: 3-9pm Saturday: 2-7pm  P.S. Time is relative to shifts given but usually work on the days listed. |
| Deokkyun | Monday: 10-10pm Tuesday: 10-8pm Wednesday: 4-7pm Thursday: 10-8pm Saturday or Sunday: 1-11pm (Decided on the previous Sunday) |
| Hong Kien | Monday: 12pm-8pm Tuesday: 12pm-4pm Wednesday: 12pm-6pm Thursday: 12pm-6pm |

**Workload Distribution**

Distribution of workload will be distributed equally to each team member where there is a mutual agreement within all team members during team meetings or discussions with a condition where all team members have to contribute or give an input to every aspect of the development or project.

Furthermore, in special circumstances only one team member is in charge of certain tasks. Tasks are listed below which are subjects to changes regarding the situation in the future.

1. Documents/reports formatting
2. Generating meeting link
3. Ensuring no merge conflict
4. Reminding each other regarding development
5. Generating initial project/file to work on

## Technology Stack & Justification

|  |  |
| --- | --- |
| **Technology** | **Justification / Alternatives** |
| Java (ORACLE) & IntelliJ IDEA | Justification:   * The main technology for developing the Application. * Since Java is an object-oriented language, it allows us to create modular programs and reusable code. * One of the advantages of Java is its ability to move easily from one computer system to another, which is platform independent. It is suitable for developing a standalone application |
| Alternatives: C++, C   * Steep learning curve since not all team members familiar with the language |
| GitLab | Justification:   * Core part of co-developing a project where developers could keep track of who contributed on which part. * The main technology for Version control * Act as a backup where history of development is shown |
| Alternatives: GitHub   * Essentially provide the same tools but extra account needs to be created given us students already been provided with one for GitLab * Need extra tutor’s approval * Not within Monash server |
| Lucid Chart | Justification:   * A comprehensive tool that provides collaboration in creating any kind of diagram * All the team members already familiar with it in previous units |
| Alternatives: Diagrams.net   * Essential features are covered but collaboration is not as simple as lucid chart has to offer |
| Figma | Justification:   * All the team members already familiar with it in previous units * Offers extensive prototyping tools * Good starting point for low-fidelity prototype which can extended into high-fidelity if needed in a single place |
| Alternatives: Balsamiq   * Limited uses with free account * Does not offer student account |
| Facebook Messenger | Justification:   * The main technology for Regular communication * Team members can connect socially * Simple messaging app to keep contact with each other |
| Alternatives: WhatsApp, Instagram   * Need to create another account to use * Unnecessary extra features * Instagram main feature is not messaging or texting |
| Zoom | Justification:   * Connect team members regardless of space which is important for us busy/active students with tight work schedule * Provide all necessary online meeting tools with simplicity at its core * Convenient to send invitation or make recurring meeting * Less distractions |
| Alternatives: Discord   * A bit of learning curve * More distractions with another servers present * Screen sharing is limited to free user |
| Google Drive | Justification:   * Offers quick and efficient web-based paperwork tools * File-sharing with minimal effort |
| Alternatives: OneDrive   * Web-based tools are not as efficient as Google Drive * Extensive tools yet complicated to use if not familiar |
| MongoDB | Justification:   * Has a separate application that act as GUI which helps to view the database * NoSQL that gives a lot of flexibility in storing entry or document * Compatibility with object-oriented programming |
| Alternatives: MySQL, Oracle SQL   * Relational database seem a bit restrictive in this case where development are not set in stone where how we develop could change depending on how each sprint goes * Relational database requires more time to adjust to changes due to nature having to set attributes early in the stage |
| API | Swing (+JButton)   * offers a suite of tools for creating graphical user interfaces (GUIs), as well as for enhancing Java's rich graphics functionality and interactivity. |

# **User Stories**

## Basic Requirements

**Setup:**

As a player,  
I want the game to have clear and concise instructions  
so that I can easily understand the game mechanics.

As a player,  
I want to see all placed tokens on the board  
So that I know the status or state of the game.

As a game board,  
I want to ensure that there are 2 players  
So that the game can be started.

As a game board,  
I want to show instruction to players,  
So that players can play easily.

As a game board,  
I want to check if player’s move follows the game rule  
So that a fair game can be played.

As a player,   
I want to be able to quit to the menu during the game from the initial setup phase  
So that I can change settings or start over.

**Gameplay:**

***First stage (placing a token)***

As a white token player,  
I want to place a token anywhere on the board at the beginning of the game  
So that I can build my opening and start the game according to the rules.

***Second stage (moving a token)***

As a player,  
I want to move my token adjacently to available position  
So that I can move the pieces precisely according to the game rules.

As a player,  
I want to be notified if I have formed a mill  
So that I am given the option to remove an opponent's piece.

As a player,  
I want to remove any available token (i.e., not in a mil) from the opponent when I place 3 tokens in row  
So that I can reduce my opponent’s available tokens to win the game.

***Third stage (flying a token)***

As a player,  
I want to move my token anywhere on the board when I have only 3 tokens left  
So that I can prevent my opponent from having 3 tokens in a row.

**End of Game:**

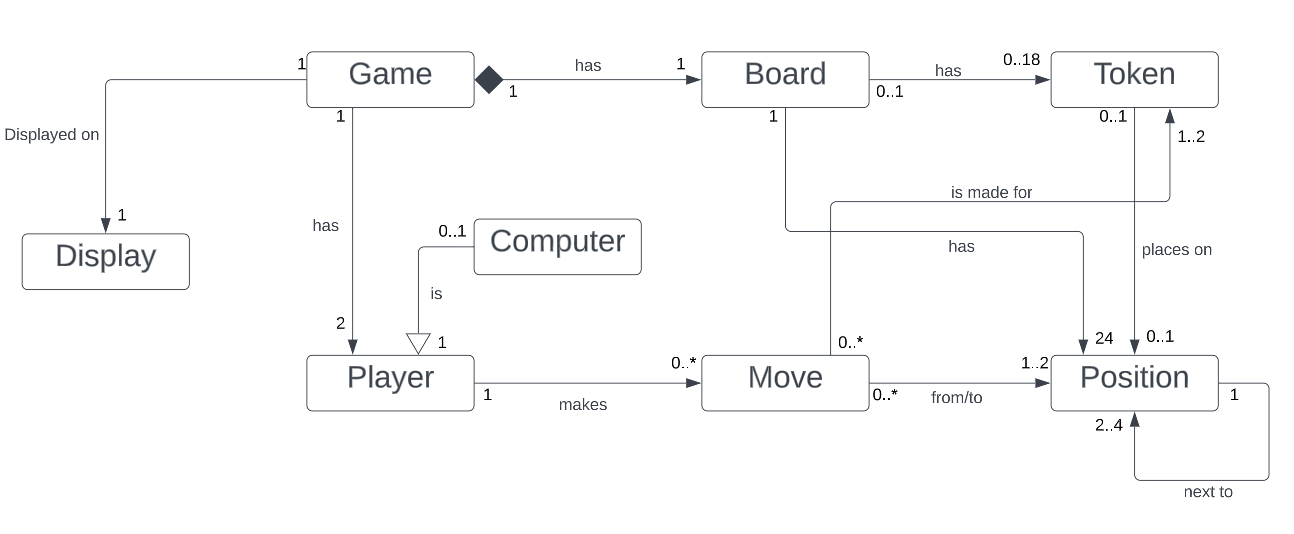
As a game board,  
I want to check how many token each player has on the board  
So that the game can be concluded once a player has less than 3 tokens.

## Advanced Requirements

**If ‘c’ is chosen (Our team choose this for now),**

As a player,  
I want to be able to choose a game mode,  
So that I can play against another human or against the computer

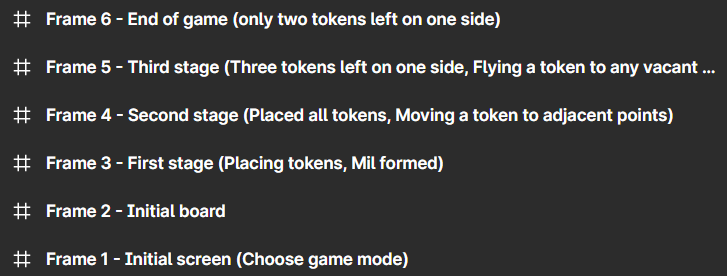
# **Basic Architecture**

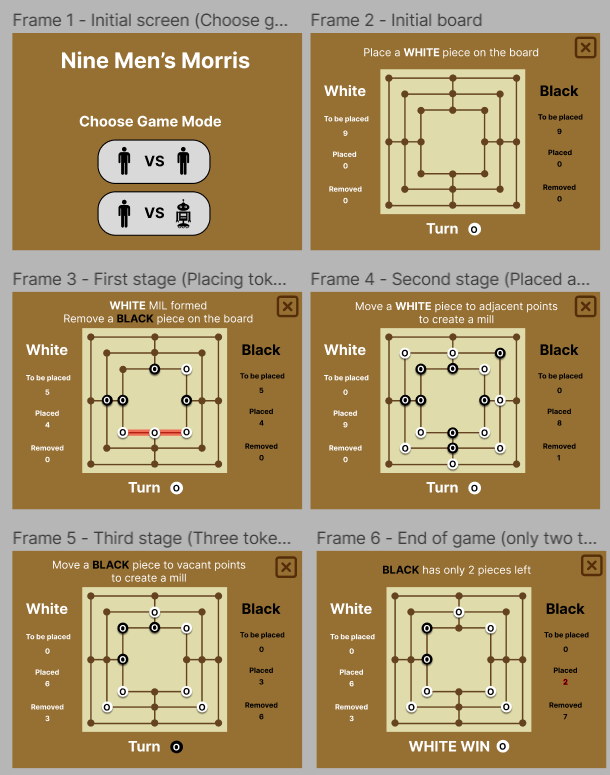


*Advance requirement C is selected here as part of the model diagram.*

|  |  |
| --- | --- |
| **Relationship** | **Justification** |
| Display & Game | Notation:  The Game and Display are related but neither is a part of the other, so the notation is Association.  Multiplicity:  One game can be displayed on one to many displays.  A display can display only one game. |
| Game & Player | Notation:  The Game and Player are related but neither is a part of the other, so the notation is Association.  Multiplicity:  One game has two players.  A player can only belong to one game. |
| Player & Computer | Notation:  The computer can act as a Player so it can inherit all the attributes from the Player entity, so the notation is Inheritance.  Multiplicity:  A player can be 0 to 1 computer.  A computer can act as a player. |
| Player & Move | Notation:  The Move and Player are related but neither is a part of the other, so the notation is Association.  Multiplicity:  A player can make 0 to many moves.  A move can be made by only one player. |
| Computer & Move | Notation:  The Move and Computer are related but neither is a part of the other, so the notation is Association.  Multiplicity:  A computer can make 0 to many moves.  A move can be made by only one computer. |
| Game & Board | Notation:  The Game is composed of one Board, and the Board cannot exist without a Game, so the notation is Composition.  Multiplicity:  A game has only one board.  A board can only belong to one game. |
| Board & Token | Notation:  The Board and Token are related but neither is a part of the other, so the notation is Association.  Multiplicity:  A board can have 0 to 18 tokens.  A token belongs to 0 to 1 board (placed or not). |
| Board & Position | Notation:  The Board and Position are related but neither is a part of the other, so the notation is Association.  Multiplicity:  A board has 24 positions.  A position belongs to only one board. |
| Move & Token | Notation:  The Move and Token are related but neither is a part of the other, so the notation is Association.  Multiplicity:  A move can be made for 1 or 2 tokens (movement of a player’s token and removal of another player’s token).  A token can have 0 to many moves. |
| Move & Position | Notation:  The Move and Position are related but neither is a part of the other, so the notation is Association.  Multiplicity:  A move can be made that affects 1 or 2 positions. (movement of a player’s token position and removal of another player’s token position).  A position has 0 to many moves made onto it. |
| Token & Position | Notation:  The Token and Position are related but neither is a part of the other, so the notation is Association.  Multiplicity:  A token can be in 0 to 1 position, with 0 being out of board.  A position allows 0 to 1 token to be on. |
| Position & Position | Notation:  The Position and another Position are related but neither is a part of the other, so the notation is Association.  Multiplicity:  A position is next to 2 to 4 adjacent positions. |

# **Basic UI Design**





*Attributes such as font, colour and size will be changed and improved as the project progresses.*

# **Reference List**

(n.d.). SVG Repo - Free SVG Vectors and Icons. Retrieved April 2, 2023, from https://www.svgrepo.com/

*Nine Men's Morris | Online Games for Kids*. (n.d.). Toy Theater. Retrieved April 2, 2023, from https://toytheater.com/nine-mens-morris/

*Wood color hex code is #BA8C63*. (n.d.). Color-Name.com. Retrieved April 2, 2023, from https://www.color-name.com/wood.color