MONASH
University

ENGINEERING

SPRINT 1

TABLE OF CONTENTS

1. **Team Information**
   Document the following pieces of information related to your team.
   - Team Name and Team Photo
     - Come up with your own personal team name. Your team name must be professional. The name of the team you belong to in Moodle (with the format <campus>_<workshop session>_<team number>) is not an acceptable team name for this task.
     - Your team photo must not be edited/photoshopped. All team members in an on-campus group must be present together physically at the time of taking the photo. For online groups, a Zoom team photo is accepted. Team photos via Zoom will not be accepted for on-campus groups.
   - Team Membership
     - Document the basic information of each team member - for example name and contact details.
     - List out what the technical and professional strengths of each member are.
     - Provide a fun fact about each member that not many people know about.
   - Team Schedule
     - Document your team's regular meeting schedule and regular work schedule.
     - Document how the workload will be distributed and managed within your team.
   - Technology Stack and Justification
     - Document what programming languages, APIs, and technologies are you planning to use and how this maps to the team's current expertise, and which ones you anticipate needing support from your tutors with.
     - Justify your team's final choice of technologies that will be used.

2. **User Stories**
   Submit a list of user stories (e.g., 10 to 25 stories) that covers both the basic 9MM gameplay and the chosen advanced requirements specified above. A majority of the user stories are expected to be devoted to the basic requirements for the Basic prototype.
   If your group consists of 4 members, your user stories must also cover the additional advanced requirement.

3. **Basic Architecture**
   Design and draw a domain model that covers both the basic 9MM gameplay and the chosen advanced requirements specified above.

   Provide detailed justifications for the domain model that you come up with, with a focus on the following aspects:
   - Rationale for each chosen domain and their relationships (if any)

- Were there any design choices that you had to make while modelling the domain and WHY?
- Explain any assumptions you have made, as well as any other part of your domain model that you feel warrants a justification as to WHY you have modelled it that way.

If your group consists of 4 members, your domain model and justifications must also cover the additional advanced requirement.

4. **Basic UI Design**

Draw low-fidelity (low-fi) prototype drawings of the proposed user interface for the application. The low-fi prototypes need to demonstrate both the basic 9MM gameplay and the chosen advanced requirements specified above. The prototypes should cover all the key interaction scenarios, e.g. initial board, placing tokens, moving tokens, 'flying', forming a mill (win condition), and the advanced feature of your choice. This can be achieved in one large drawing space or across multiple pages. Avoid redundancy, i.e. do not create multiple prototypes for the same interaction. All drawings should be large and clear enough to understand and any writing should be legible. You may use pen and paper, or digital drawing tools. If your group consists of 4 members, your lo-fi prototype drawings must also cover the additional advanced requirement.

If your group consists of 4 members, your lo-fi prototype drawings must also cover the additional advanced requirement.

1 Introduction

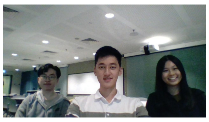
The project’s goal is to create an object-oriented design for a game named Nine Men's Morris that will be fully functional implemented in the last sprint. Our team, The Brewing Java will be asked to advance this game’s development and demonstrate that the stated Sprint milestones are met in each Sprint. In early sprints, we will concentrate on the basic elements of the game, such as User stories, Basic Architecture and UI Design, Tech Stack and Rationales. During the final sprint, we will be asked to expand our implementation with new features.

In this document, we will first introduce the team members, with the team information provided, such as team name, photo, team membership and team schedule. Then, we will cover the programming languages, APIs and technologies we plan to use and provide justification on our final decision. We will create a list of user stories that covers both basic Nine Men's Morris gameplay and chosen advanced requirements by applying all aspects of the INVEST acronym. We will also highlight the basic architecture of our project which includes domain model and rationale for each chosen domain. Moreover, low-fidelity prototype drawings of the proposed interface for the game are drawn. An appendix with all the links attached to the documents such as Work Breakdown Structure, Domain Model and Prototypes. With that, it concludes our report on sprint one deliverables.

2 TEAM INFORMATION

1. Team Name and Team Photo

Team Name: The Brewing Java
Team Photo:



2. Team Membership

Name	Email	Technical & Professional Strengths	Fun Fact
Fong Zhiwei	zfon0005@student.monash.edu	Object-oriented design	Night Owl
Kennedy Tan	ktan0087@student.monash.edu	Software development methodologies	Plays futsal
Soh Meng Jienq	msoh0007@student.monash.edu	UX design	Plays piano

3. Team Schedule

Our team decided to have a regular meeting every Saturday 10am - 12pm. A discussion will be made in WhatsApp group to confirm the agendas every Friday. The meeting will be held on Discord and one of the members will be responsible to note down a meeting minute. It contains the meeting content and it will be very useful if any of the members is not available for the meeting and for members to check on the details in the meeting.

For our team, we created a Work Breakdown Structure (WBS) which is attached in the appendix below this document. Project was broken down into smaller, manageable tasks. All members are required to update the WBS while they are working on this project, so that we can monitor each other's progress. We found this way is easier for us to manage and plan the project as workload was distributed evenly. If any of the members falls behind on his/her tasks, we will discuss among ourselves and help on the member's task accordingly.

4. Technology Stack and Justification

- Document what programming languages, APIs, and technologies are you planning to use and how this maps to the team's current expertise, and which ones you anticipate needing support from your tutors with.
- Justify your team's final choice of technologies that will be used.

3 USER STORIES

We come out with 16 user stories for this project, that covers both basic Nine Men's Morris gameplay and chosen advanced requirements based on the regular template, “As a (role), I want to (desire action/reaction), so that (benefit).”

User Stories (Basic and advanced requirements)

- As a user, I want to have a tutorial mode in the game, so that I can learn how to play the game.
- As a user, I want to have hints of all legal moves, so that I know where to place the next piece.
- As a user, I want to be unable to make an illegal move, so that I cannot cheat in the game.
- As a user, I want my piece to have a different design than my opponent’s piece, so that I can differentiate my piece from my opponent’s piece.
- As a user, I want to be able to remove one of my opponent’s pieces when a mill is achieved, so that I can increase my winning possibility.
- As a user, I want to move my piece to an adjacent point after 18 pieces have been played, so that I can make a mill.
- As a user, I want to know the result of the game, so that I know who wins the game or it is a draw.
- As a user, I want to know the amount of remaining pieces, so that I can plan my future moves easily.
- As a user, I want to place one piece at a time onto the board, so that the players can take turns to play all their pieces.
- As a user, I want to be able to place my piece only on an empty place, so that it is only possible to only have a single piece in one spot on the board.
- As a user, I want my opponent to not remove any pieces that I formed as a mill, so that I can protect my mills on the board.
- As a user, I want to be able to move my piece to any empty intersection without the limitation of only moving to an adjacent dot after I have been reduced down to the last 3 pieces, so that it can increase my winning possibility.
- As a user, I want the game to end when either of the players has no possible move or has reduced to only 2 pieces on the board, so that I can quickly start a new match.
- As a user, I want to have 2 players in a single game so that I can play the game with an opponent.
- As a user, I want to be able to restart the game, so that I can start a new match anytime.
- As a user, I want to be able to exit the game, so that I can stop playing at any time.

4 BASIC ARCHITECTURE

Domain Model

Design and draw a domain model that covers both the basic 9MM gameplay and the chosen advanced requirements specified above.

We come out with a domain model that covers both Nine Men's Morris gameplay and the chosen advanced requirements. It is shown below.

Rationale

Provide detailed justifications for the domain model that you come up with, with a focus on the following aspects:

- Rationale for each chosen domain and their relationships (if any)
- Were there any design choices that you had to make while modelling the domain and WHY?
- Explain any assumptions you have made, as well as any other part of your domain model that you feel warrants a justification as to WHY you have modelled it that way.

GameMode is a composition of Game. Abstract class GameMode is a child class of a parent class Game. Every Game of Nine Men’s Morris has a GameMode and GameMode will not exist if Game does not exist. GameMode is an abstract class because there will be multiple child classes that inherit abstract class GameMode into specific different game modes.

NormalMode extends GameMode, **TutorialMode extends GameMode.** NormalMode and TutorialMode inherit GameMode abstract class as they are specific game modes for the Game class.

Board is a composition of Game. Board is a child class of a parent class Game. Every Game of Nine Men’s Morris has a single Board and Board will not exist if Game does not exist. Board is where the game of Nine Men’s Morris will be played.

Intersection is a composition of Board. Intersection is a child class of parent class board. The Board in every game of Nine Men’s Morris has 24 Intersections and these Intersections cannot exist if there is no Board. Each Intersection acts as a place where the player can place their tokens.

Player is a composition of Game. Player is a child class of a parent class Game. Every Game of Nine Men’s Morris has 2 players and players will not exist if there is no game. 2 players play the game of Nine Men’s Morris by taking turns to perform actions.

Player---<<has>>--->Token and Game---<<has>>--->Token. Each Player has 9 objects of Token class at the start of the game and it may decrease as the game of Nine Men’s Morris goes on. Each Game starts with all its Intersections of the Board as empty and it can be filled with 18 objects of Token class maximum if both Players placed all their Token objects.

Intersection---<<has>>--->Token. Every Intersection on the board is empty when the game of Nine Men's Morris starts and each Intersection can be filled with a single object of Token class.

WhiteToken extends Token, **BlackToken extends Token.** WhiteToken and BlackToken inherit the Token abstract class. WhiteToken and BlackToken inherit all the methods and attributes from the Token abstract class. The only difference between WhiteToken and BlackToken is their colour to differentiate between them as they are held by different Players.

Token has an aggregation relationship with Mill. A Mill is formed by having 3 objects of Token class from the same owner (e.g. 3 WhiteTokens or 3 BlackTokens) placed in a certain Intersections of the Board. A Mill has to have 3 objects of Token class but objects of Token class can exist without Mill class, thus they have an aggregation relationship.

Mill is a composition of Board. Mill is a child class of a parent class Board. Every Board can have 0 to 16 Mills and the Mills cannot exist if there is no Board as there will be no possible way to form them.

Player---<<performs>>--->Action. Player creates an object of the Action abstract class to perform various actions when it is his turn. Players can use actions to perform actions onto the game such as restarting the game, quitting the game and turning on hints. Players can also use actions to perform moves onto the board and doing this will end the player’s turn.

RestartAction extends Action, **QuitAction extends Action,**
HintAction extends Action, MoveAction extends Action.
These 4 types of Actions are concrete classes that extends the abstract class Action, they inherit the attributes and methods of Action but they each have their own specialised methods or attributes to help me affect the Game in a different way, RestartAction will restart the Game into a new Game, QuitAction will stop and exit the current Game, HintAction can be toggled to show the valid moves that the Player can perform, MoveAction will allow the Player to affect the Board.

Move is a composition of MoveAction. Abstract class Move is a child class of a parent class Move. Every MoveAction has an object of Move abstract class and the Move cannot exist if there is no MoveAction so there is no possible way to move affect the Board without a MoveAction. Move is used to affect the status of the board.

Move---<<is validated on>>--->Board. Any object of Move class will be validated on the Board to ensure it is a valid move based on Nine Men’s Morris rules before it is able to affect the Board..

PlacingMove extends Move, **FlyingMove extends Move,**
SlidingMove extends Move. These 3 concrete classes extend the Move abstract class. PlacingMove is used by the Player to place a Token onto an empty intersection on the Board. FlyingMove is used by the Player to move a piece to any intersection if and only if they have less than 3 tokens left. SlidingMove is used to move a token to an adjacent intersection.

5 BASIC UI DESIGN

Draw low-fidelity (low-fi) prototype drawings of the proposed user interface for the application. The low-fi prototypes need to demonstrate both the basic 9MM gameplay and the chosen advanced requirements specified above. The prototypes should cover all the key interaction scenarios, e.g. initial board, placing tokens, moving tokens, ‘flying’, forming a mill (win condition), and the advanced feature of your choice. This can be achieved in one large drawing space or across multiple pages. Avoid redundancy, i.e. do not create multiple prototypes for the same interaction. All drawings should be large and clear enough to understand and any writing should be legible. You may use pen and paper, or digital drawing tools.

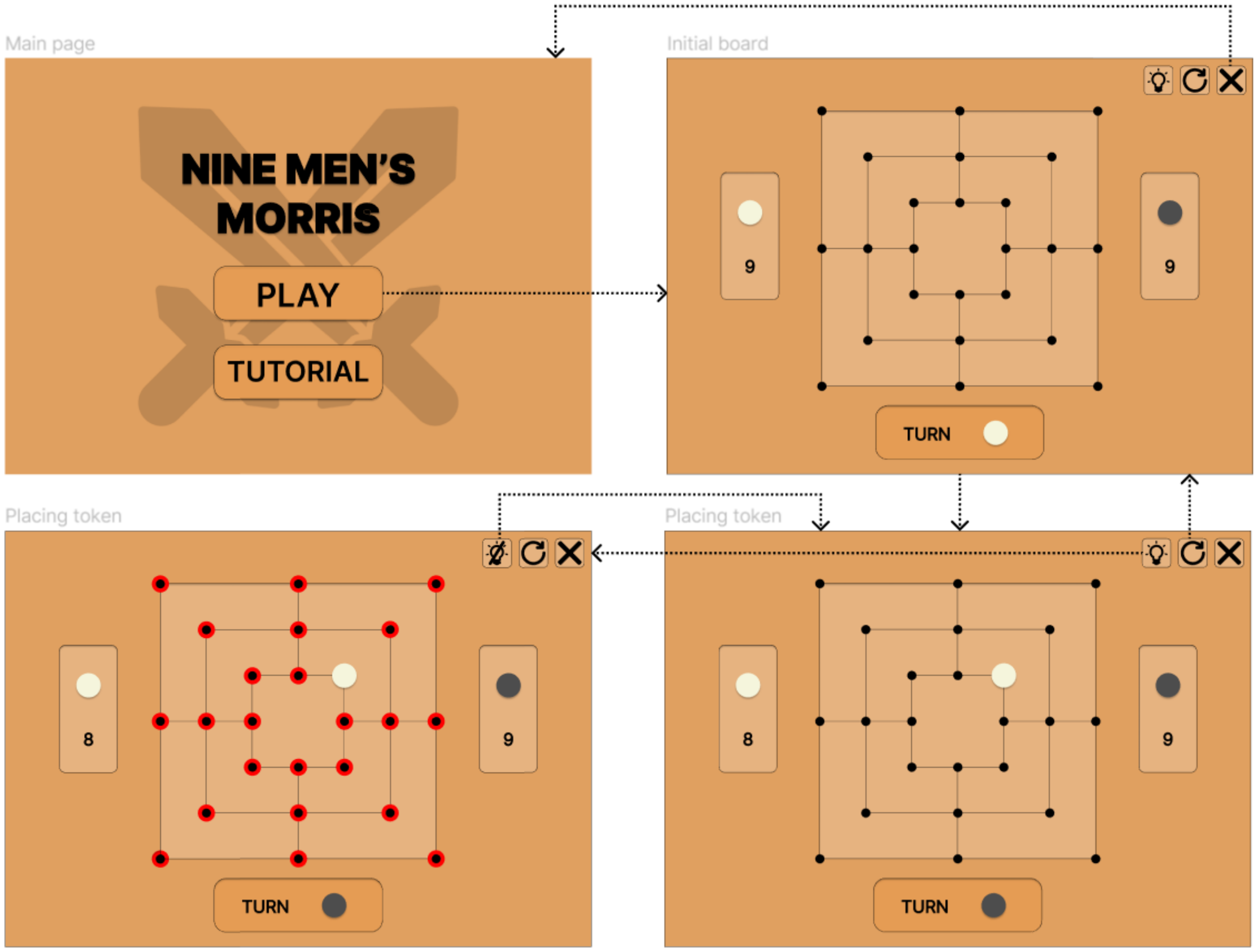
Low-Fidelity Prototype Drawings

There are two modes in our game, which are game and tutorial mode.

Game Mode

When players press “PLAY” to start a match, it will bring them to an initial board which consists of 24 intersections. Each player has 9 tokens in their respective colour, which are either black or white. The game always begins with a player holding white tokens, and they take turns to play.

On the top right corner, there are 3 buttons, which represent hint, restart and close. The hint button will display all the legal moves of the player. And, when the player presses the hint button again, the hints will be disabled. Restart button will bring them to the initial board to start a new game. Close button will lead them to the main page.



After both of the players place their 9 tokens,

Tutorial Mode

5 APPENDIX

Work Breakdown Structure

https://docs.google.com/spreadsheets/d/10B28ICSSwzXOpijzydfFUe32XY_a1BIKvHQG8HIq6rk/edit?usp=sharing

Low-Fidelity Prototype Drawings

https://www.figma.com/files/team/1222773496085811278

Domain Model

https://lucid.app/lucidchart/72bd3d5f-67f6-485c-8377-ad7802a7a456/edit?viewport_loc=-108%2C-155%2C2219%2C1097%2C0_0&invitationId=inv_ca1797a5-fe22-4640-8cd0-134bc8e9cfe4