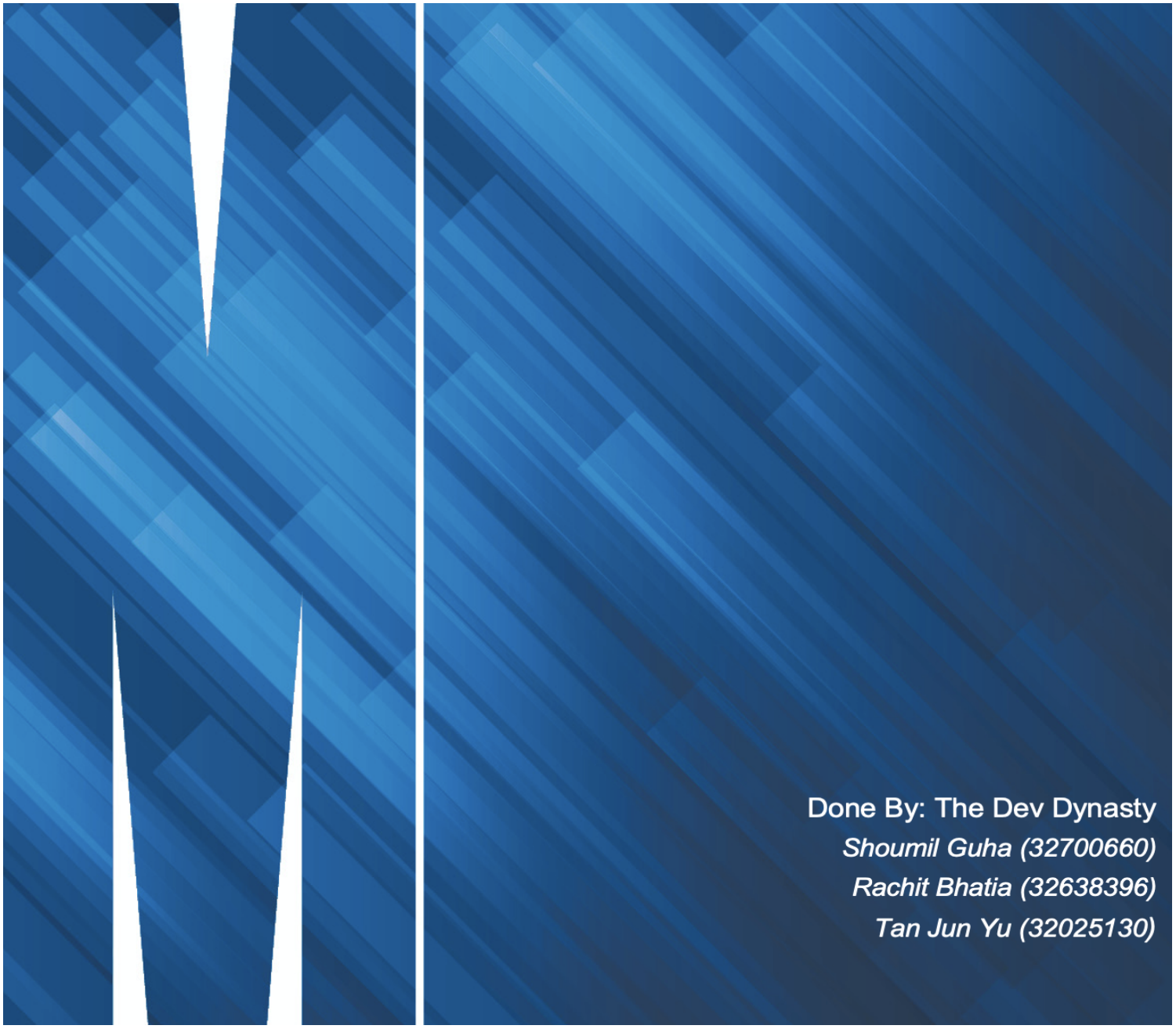
MONASH University

FIT3077
SPRINT 1: PROJECT INCEPTION



Done By: The Dev Dynasty

Shoumil Guha (32700660)

Rachit Bhatia (32638396)

Tan Jun Yu (32025130)

INDEX

1. Team Information

Document the following pieces of information related to your team.

■ 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.

Team Name : The Dev Dynasty

Team Photo :

Team Members Contact Details

1. Shoumil Guha
    - **Phone**: 011-5640-9250
    - **Discord**: shoumil#3182
    - **Email**: sguh0003@student.monash.edu

---

2. Rachit Bhatia
    - **Phone**: 019-405-7032
    - **Discord**: racrage#2757
    - **Email**: rbha0031@student.monash.edu

---

3. Tan Jun Yu
    - **Phone**: 016-278-3792
    - **Discord**: junyu#7430
    - **Email**: jtan0245@student.monash.edu

Team Members Info

1. Shoumil Guha
    - **Technical and Professional Strengths :**
        - ➢ Well versed in object oriented programming and its design using Python and Java.
        - ➢ Experience in database management systems, web development, image processing.
        - ➢ Fast learner and starts work early which leads to no panicking close to deadlines.
        - ➢ Proficient in the agile working pattern due to past experience in Scrum projects
    - **Fun Fact :** I eat breakfast an hour before lunch just because I can.

2. Tan Jun Yu

   - **Technical and Professional Strengths :**
     - ➢ A well-developed understanding on Object Oriented Programming principles which is vital for this assignment .
     - ➢ Good collaboration skills by being able to work effectively with team members and actively participate in any group discussions for the best project outcome
     - ➢ Familiar with the programming language we have chosen for this assignment that is Java being applied with Object Oriented Programming
     - ➢ Experienced in the field of agile projects
   - **Fun Fact :** I dislike playing 9 Men's Morris and will not ever want to play or see this game again after this assignment

---

3. Rachit Bhatia

   - **Technical and Professional Strengths :**
     - ➢ Very familiar with Object-Oriented Programming principles due to experience from prior projects.
     - ➢ Equipped with some knowledge in UI development for iOS mobile applications
     - ➢ Well-experienced with Java principles and semantics
     - ➢ Proficient in team management
     - ➢ A good team player with the ability to provide valuable input to every discussion while respecting everyone's opinions
     - ➢ Well adapted to an agile team format
   - **Fun Fact :** I love to listen to music in all moods except when feeling overwhelmed or stressed

Team Schedule

❖ Meeting Schedule

- ➢ At the beginning of each sprint, the team will hold a sprint planning meeting to discuss and agree on the sprint goal, scope, and tasks to be completed. Task division for the sprint will be done in this meeting.
- ➢ A weekly standup meeting will be held to check on the team's progress to ensure that everything is on the right pace to lessen the chance of missing deadlines. Each team member will update one another on the progress of one's assigned tasks and make sure no one is left behind. This meeting will usually be on every Friday at 4pm. The timing may change subject to the team's convenience but the meeting cannot be cancelled.
- ➢ The frequency of these weekly standup meetings will change from once a week to 4 times a week when closer to the end of the sprint. When this change takes place will depend on the remaining amount of work for that particular sprint (frequency of meetings could increase from 7 to 10 days before the end of the sprint).
- ➢ After the end of each sprint (preferably the same or next day after a sprint ends), an additional sprint retrospective meeting will be held to discuss the shortcomings of the completed sprint and possible improvements for future sprints.

- Work Schedule
  - Every member is expected to spend at least 1-2 hours per day on the allocated work. This may increase based on the workload for the sprint.
  - The tasks given to each team member will be assigned an internal deadline considering the availability of the member and the difficulty of the task.
  - In every standup meeting conducted, the work done by each member will be discussed and reviewed. At the same time the team will look at any issues faced by a member and try to resolve them to smoothen the workflow.
  - It will be aimed to allocate roughly the same amount of workload to each sprint to ensure consistency in the workflow.

- Workload Division and Management
  - Every type of activity (writing user stories, coding, prototyping) is distributed equally
  - Tasks allocated to each member will cover all aspects of the project to prevent siloing.
  - Tasks will be created at the beginning of the sprint and will be managed through a Trello board. Each task's priority will also be recorded. All of this will be done during the sprint planning meeting.
  - Each member will update the status of task completion on the Trello board.
  - Team members should voluntarily choose tasks from different aspects to make sure everyone is comfortable with their workload. The final decision should be agreed upon by the other members.
  - Any task that might be harder to implement can be divided into smaller tasks and assigned to more than one team member.
  - Any severe difficulty encountered by a team member on the given tasks can be further discussed in a standup meeting and a decision whether to reassign the particular task to another member can be made.

Technology Stack and Justification

Definition and Purpose:

Chosen Programming Language: Java

Pros:

1. Strong structured language
2. Industry standard for Object-Oriented Programming
3. Widespread support online as it has been around for decades
4. Platform independent

Cons:

1. Slow
2. Poor GUI tools especially for complex GUIs
3. Verbose. Takes much more code to implement the same tasks in other languages

Justification for Selection:

Java is selected as our programming language due to its strict rules of syntax that precisely conforms to the Object-Oriented Programming principles. With Java being an object oriented language, methods and variables are declared or implemented in each of their respective classes. Hence, it allows a better readability of the code. Besides that, Java provides easier employment of encapsulation by just using access modifiers for the variables declared as compared to other languages.

Most importantly, all of our team members have experience with using Java as an object-oriented language from prior projects and thus will ease the process of code development rather than trying to familiarise ourselves with other languages being applied with Object-Oriented Programming.

Discarded Alternative Programming Language: Python

Pros:

1. More unstructured with less rules to follow when programming
2. Overwhelming support online
3. Extensive support libraries
4. Has many third party packages
5. Simplicity

Cons:

1. Requires additional libraries to implement and use classes
2. Slower than Java as code is interpreted at runtime
3. Unconventional for developing object oriented programs

Justification for Discardment:

Due to the flexibility of Python which supports variety of programming styles, Object-Oriented may not be rigorously enforced. Scripts can still be executed without any classes being declared and this strongly violates the rules of Object-Oriented Programming.

Moreover, none of our team members have experience in using Python as an object oriented language that may end up being time consuming to adapt to the new kind of formats in which Python supports.

<u>Chosen GUI Tool: JavaSwing</u>

Pros:

1. Vast amount of component types
2. Components support additional features well
3. Code base is entirely implemented in Java
4. Standard GUI Library
5. Platform independent
6. Components easy to customize

Cons:

1. Hard to move components around in the Swing Designer
2. Slower Performance than AWT if the programming is not well designed

Justification for Selection:

JavaSwing and JavaFX are the two main GUI Frameworks we have been considering. After trying out both the GUI Frameworks, JavaSwing is found to be the easier one to apply with Object-Oriented Programming. It is found that the components created in JavaSwing can be easily delegated into their respective classes while we encountered some difficulties in JavaFX.

Discarded Alternative GUI Tool: JavaFX

Pros:

1. Designing UI can be easy with the help of external application like SceneBuilder.
2. Independent on operating system
3. Easy to create complex UIs with lesser code

Cons:

1. Smaller community support compared to Swing

Justification for Discardment:

Although JavaFX was initially more preferable than JavaSwing (chosen GUI Framework) , but we have discovered a hurdle while testing out JavaFX that we have yet to find a solution for it. While creating components and adding them to the GUI frame, it is found that the code implementation for the all the different components will have to be added to the same one and only one Controller Class .This could be a major issue that might result us in not being able to conform to Object Oriented Programming principles as we might not be able to separate different components into different classes together with their own methods and variables.

Discarded Alternative GUI Tool: AWT (Abstract Window Toolkit)

Pros:

1. Low crashing rate
2. Basic user interface design, easy to use especially for beginners

Cons:

1. Limited types of component in the toolkit
2. Components may not have enough support for special features like images
3. Simple layout management makes it difficult to build complex GUIs
4. Limited customization for components
5. No extensibility.

Justification for Discardment:

Abstract Window Toolkit(AWT) has considerably lesser amount of components and functionalities as compared to both JavaSwing and JavaFx. With that being so, code implementation may be more abundant when AWT is used due to the lesser amount of in-built features and components. In addition,the appearance of the components in AWT is mainly unconfigurable which might hinder the UI designing process of the 9 Men’s Morris game.

Discarded Alternative GUI Tool: SWT (Standard Widget Toolkit)

Pros:

1. Rich component types
2. Components support additional features well

Cons:

1. Not in the standard library of the JRE (Java Runtime Environment)
2. Steep learning curve if unfamiliar with underlying native GUI libraries
3. Can be complex to use due to its sophisticated features and customizability
4. Harder to find support

Justification for Discardment:

Standard Widget Toolkit(SWT) are known to have unstable performance when is being operated on any other operating systems other than Windows. In other words, SWT are vulnerable to crashes in other operating systems like MacOS in which two of our members are currently using. This may be an aggravating hindrance that could impact the development process that our team should certainly avoid.

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 requirements.