FIT3077 - Architecture and Design

Sprint One

MA_Tuesday08am_Team123

Members:

Maliha Tariq

Table of Contents

1. Team Information

1.1 Project Summary

This project involves developing a game 'Fiery Dragons'. The game needs to adhere to proper software development practices and object-oriented design principles.

At the onset of development, our focus lies on the implementation of the core functionalities essential to the Fiery Dragons gaming experience. This involves crafting a robust framework that enables two to four players to engage in gameplay within a unified client instance. Paramount to our efforts is the meticulous alignment of the game mechanics with the standard rules, ensuring a cohesive and immersive gameplay experience for all participants.

This report comprehensively covers various aspects of our project, including the technology stack to be used, the user stories, the domain model, and the low-fidelity (low-fi) prototype design.

1.2 Professional Team Name and Professional Team Photo

Team Name:

Team Photo:

1.3 Team Members Information

Member 1
Name: Maliha Tariq
Student ID: 33473692
Email address: mtar0012@student.monash.edu
Technical Strengths: Java, Python and CSS
Fun Fact:

Member 2

Name: Loo Li Shen

Student ID: 32619685

Email address: lloo0011@student.monash.edu

Technical Strengths: Java, Python

Fun Fact:

Member 3

Name: Khor Jia Wynn

Student ID: 33033919

Email address: jkho0044@student.monash.edu

Technical Strengths: Java, Python

Fun Fact:

Member 4

Name:

Student ID:

Email address:

Technical Strengths:

Fun Fact:

1.4 Team Schedule

1.5 Technology Stack and Justification

Programming Language: Java
Justification: Our team has collectively decided to use Java as the primary programming language for developing the "Fiery Dragons" game. This decision is primarily based on our team's extensive experience and proficiency with Java. Java's object-oriented programming (OOP) capabilities make it an ideal choice for designing and implementing the complex interactions and behaviours required in a game environment. Furthermore, Java's platform independence ensures that our game can be run on any operating system without the need for modification, thereby broadening our potential user base.

User Interface API: Swing
Justification: For the game's user interface development, we have chosen the Swing library. Swing is a part of Java's standard library, which provides a set of widgets and tools for creating graphical user interfaces (GUIs). Though our team is not familiar with Swing, its comprehensive documentation and support available online make it a convenient choice for developing the game's interface. Swing's integration with Java also facilitates a smoother development process, allowing us to focus on game logic rather than compatibility issues.

IDE: IntelliJ IDEA
Justification: The Integrated Development Environment (IDE) selected for this project is IntelliJ IDEA. Our choice is motivated by IntelliJ's advanced code completion, refactoring, and debugging capabilities, specifically tailored for Java development. IntelliJ IDEA supports direct integration with Swing, which simplifies the GUI design process through its intuitive drag-and-drop interface builder. The majority of our team members are already proficient with IntelliJ IDEA, which we anticipate will streamline the development workflow and enhance productivity.

Version Control: Gitlab
Justification: To manage our project's source code and facilitate collaborative development, we will be using Gitlab as our version control system. It will enable our team to track changes, revert to previous versions if necessary, and manage different development branches effectively.

Anticipated Support Needs:

Swing Advanced Features: While our team is confident in our ability to utilize Swing for basic GUI development, we may require support in implementing more advanced features or optimizing the game's performance.

Game Design Principles: As we are more experienced with general software development than game development specifically, guidance on best practices and design patterns specific to game development would be beneficial.

Testing and Debugging: Support in setting up a robust testing framework for game development could help ensure the reliability and quality of the game.

Final Justification:

Our choice of technology stack is grounded in our team's current expertise and the specific requirements of the "Fiery Dragons" game. Java, with its powerful OOP features, combined with Swing for GUI development, provides a solid foundation for creating a complex and interactive game. IntelliJ IDEA enhances our development efficiency through its sophisticated Java and Swing support. This technology stack not only leverages our existing skills but also aligns with the project's needs, ensuring a smooth development process and a high-quality final product.

1.6 Analysis of Alternative

We consider Python with Pygame as a viable alternative to our primary choice of Java and Swing for the development of "Fiery Dragons." Python offers simplicity and readability, which could facilitate faster development cycles, especially beneficial for teams with a strong Python background. Pygame, a library tailored for video game development in Python, provides essential tools for creating interactive and graphically intensive games, making it suitable for both beginners and advanced developers.

While Python is a powerful and flexible language that supports object-oriented programming (OOP), its dynamic nature can introduce complexities in enforcing strict OOP principles compared to Java. Java was designed with a strong emphasis on OOP, making it inherently structured to support complex object-oriented designs. Its syntax and language features encourage the use of interfaces, abstract classes, and strong typing, facilitating the implementation of design patterns and OOP principles straightforwardly.

On the other hand, Python's dynamic typing and flexibility, though advantages in many scenarios, can make it more challenging to apply strict OOP methodologies. The dynamic type system allows for more flexibility in how objects and classes interact, but this can lead to less predictability and more difficulty in ensuring type safety and adherence to specific OOP conventions. The choice between these two sets of technologies hinges on the specific needs of our game, the team's familiarity with the programming languages, and the target platforms for the game.

2. User Stories

1. As a game designer, I want to enforce a rule that prevents players from using the same chit card more than once during their turn, ensuring all players adhere to fair play standards.

2. As a game designer, I want the game to stop a player from moving its token when the token tries to gain a position that is already occupied by another player's token.

3. As a game designer, I want to expel players who try to violate regulations more than 3 times.

4. As an admin, I want to move the players' tokens backward to the same expected position as other users and flipping the chit cards that misaligned the card that the token settles.

5. As an admin, I want to reset players' tokens and chit cards to their correct positions if discrepancies occur, maintaining order and consistency in the game environment.

6. As a player, I want a unique coloured animal to differentiate my game character from others.

7. As a player, I want to be able to select chit cards where the animal depicted matches the animal in my current position on the game board so that I can strategically move forward in the game.

8. As a player, I want my game to start with my token placed at the cave, allowing me to plan and execute strategic moves to navigate back to the cave, thus fulfilling the game's primary objective.

9. As a game designer, I want to shuffle the uncut segments cut segments as long as the cut and uncut segment

10. As a player, I want to be able to shuffle the deck and place them down in the centre.

11. As a player, I want to be able to reveal another chit card after a successful move to potentially move again.

12. As a player, I want my turn to end if the revealed chit card shows a different animal than my current space.

13. As a game designer, I want to check if a player reaches their cave with an exact number of moves so that I can end the round and declare the winner.

14. As a game designer, I want to track whether a square contains a dragon so that I can prevent other dragons from occupying the same square.

15. As a game designer, I want to cover up all the dragon cards that were flipped at the end of a player's turn, to implement the memory component of the game.

16. As a game designer, I want to implement the dragon pirate card to give the player the option of moving backwards.
17. As a game designer, I want to incorporate a feature that allows the player to choose different board configurations, enhancing the replayability of the game.

18. As an admin, I want to provide clear guidelines for the game rules to ensure a fair and enjoyable experience for all the players.

19. As a solo player, I want the option to play with a dummy player controlled by the game that follows the rules of the game

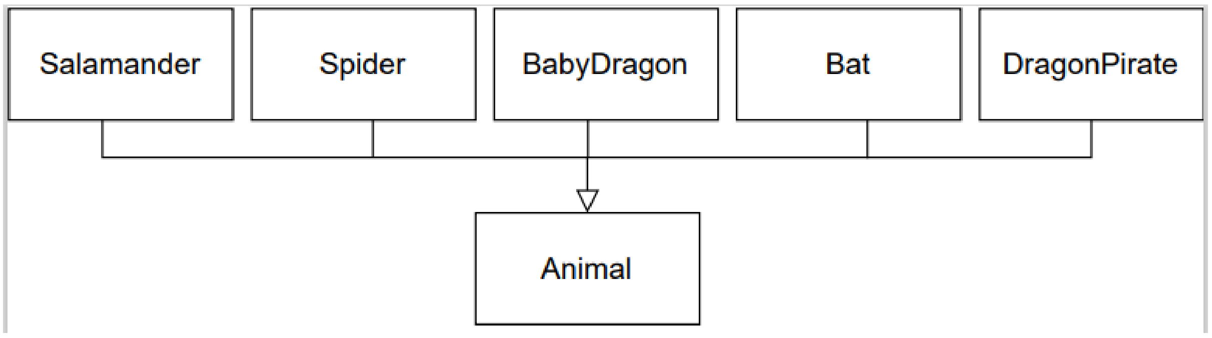
20. As a parent, I want the game to offer educational benefits, such as memory skills and decision-making, for my child.

21. As a player, I want the game to last around 15 minutes so that it's quick and enjoyable for casual play.

22. As a player, I want to celebrate and congratulate the winner of the game, whether it's myself or another player, to acknowledge their achievement.

23. As a player, I want to have different game modes/versions of the game. For example, like in Monopoly where a specific location can grant you cards which gives your character effects whether it be good or bad.

3. Domain Model



4. Basic UI Design

https://www.figma.com/file/Wwzd74lduk5JHXwxuhV8y7/Untitled?type=design&node-id=0%3A1&mode=design&t=jIC1Wbmb3yO3Xro1-1