FIT3077 Sprint 1

Team InfiTech

Shannon Wallis, Ziheng Liao, Harshath Muruganantham

Monash University

FIT3077 - Software Architecture

27/03/2023

Table of Contents

Table of Contents **3**
Team Information **4**
InfiTech Team 4
InfiTech Team Members 4
Harshath Muruganantham 4
Shannon Wallis 5
Ziheng Liao 5
Team Schedule 5
Workload Management 6
Technology Stack and Justification 6
Selected Programming Language 8
Python Supporting Library Evaluation 9
Selected Libraries 12
User Stories **13**
Proposed Extensions 16
Domain Model Design **18**
Entity Justifications 19
Relationship Justifications 23
Assumptions 34
Discarded Domain Model Associations, Entities and Cardinalities. 35
Low-fi diagram **38**
References **44**

Team Information

The proceeding sub-section - ‘Team Information’ provides an overview of the members that make up ‘InfiTech’ - our chosen team name. It delves into some of our skills and interests. Further, we will outline the preferred working schedule of the team, that we will attempt to adhere to throughout the duration of our 12 week project.

InfiTech Team

InfiTech Team Members

Name	Email
Harshath Muruganantham	hmur0018@student.monash.edu
Shannon Wallis	swal0059@student.monash.edu
Ziheng Liao	zlia0050@student.monash.edu

Harshath Muruganantham

Hey, I’m Harshath, a third year Software Engineering student at Monash University. Some of my technical skills include knowledge of Python, R, Java, Javascript and C. I have worked on a variety of projects in a team environment both within university, as well as in volunteering which has helped me equipped me with the right skills to approach this group assignment.
On the soft skills side, through my volunteering experiences in Monash DeepNueorn as well as 180Degrees Consulting, I have improved my communication as well as my teamwork skills.
A fun fact about me is that I’ve never drank coffee before!

Shannon Wallis

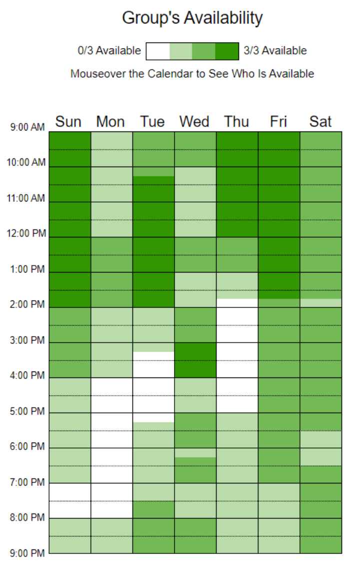
Hi I am Shannon, a fourth year Software Engineering and Commerce student at Monash University. I am excited to be a part of the InfiTech team! I have technical skills in programming, UML design and business analysis. I have worked on a range of projects in and outside of university that have provided me with a broad background for the upcoming work. I think my professional strength is organisation, communication and attention to detail.
On the less serious side, I love sports including Netball and Hockey, I also really love skiing! The last place I was lucky enough to ski in was Hakuba, Japan. For part time work, I organise and run different events.

Ziheng Liao

Hello I'm Ziheng and currently in my third year of Software Engineering and I'm thrilled to be here.. Some of my strengths are my ability to problem solve out of problems I am stuck with. With the help of Google and AI and most importantly Youtube, I am unstoppable in my pursuit of becoming the smartest programmer in the world. A fun fact about me is that Python is my first programming language and it's become my most comfortable language to work with!

Team Schedule

As a team, we have competing availability as we manage university, part time work and social time. In an effort to manage these and come up with a reasonable schedule, we have created a union chart of our availability for project work at InfiTech (see below)



We have a considerable number of hours with full 3/3 availability.
Our preferred meeting times are:

- Sunday Morning : 10am - 12pm
- Friday Morning : 10am-12pm
- Wednesday Afternoon: 3pm - 4pm

We ideally will schedule around 2 meetings in the lead up to a Sprint 'go live' (deadline) and more when in the submission week

Workload Management

We will be doing all distinguishable tasks as a team with calls 2 times a week to communicate and work together. We will strive to work collaboratively where nobody will be working by themselves on their own separate component of the tasks.
Ie: Where there are tasks of creating user stories, UML design for example, we will ensure that all members contribute some of their time to both tasks, rather than 1 member doing just the User stories, and 1 member doing solely the UML.

During the sprints, we will provide a numerical indicator of each user story to determine how much time and work needs to be put in to ensure everybody gets an even distribution of work. If a member has trouble, sufficient assistance will be provided and in some cases, a member of the team who is more comfortable with the relevant technology will take over the user story and the other member will move on to a different user story.

InfiTech Workload Management Values:

- At InfiTech, we strive to challenge ourselves by providing each member an opportunity to learn new skills and implement features they might not be 100% confident with.
- We also strive to ensure that work completed by members is balanced, with no member overbearing and not allowing the collaboration of members, while no member failing to contribute (we want the ‘happy medium’!)

Technology Stack and Justification

For the development of our ‘board game’ software implementation, InfiTech was offered the choice between 2 programming languages - Java and Python.

Further restrictions placed on development were surrounding its executability - the board game must be an executable capable of being run on any laptop device without the need to install separate libraries or programming languages.

The following is an analysis of the 2 programming languages, focusing on the libraries offered within each. We will provide a consensus on the selected programming language at the bottom.

Languages		
Python		
Evaluation		
Factor	Pro or Con	Reasoning
Familiarity/Confidence	Pro	Complexity of Python is very simple. It is a very high level language where a lot of the functionality is already handled for us such as allocating memory for us in a list. The debugging features are very intuitive with the assistance of VS code and other tools. The syntax is also fairly straightforward. Most importantly the team has years of experience with working with Python.
Readability	Pro	Python's syntax is extremely straightforward where a lot of the tasks are already handled for us with a simple call of a function. The indentation of the code block also increases readability and allows logic to be easily followed and implemented.
Existing Libraries Provided	Pro	Pygame is a library within Python that is well suited to the requirements of this project. Given we are creating a board-game with a GUI to play, we will need a library to facilitate this. Pygame is well understood and utilised by small application game developers in the IT community. Thus, we will be easily able to learn about the methods and functionality of the library from external resources
Well Documented	Pro	Python is extremely well documented with 49% of all programmers showing some familiarity with python ensures that any troubles that we encounter will most likely have already been encountered by somebody else.
Object Oriented Support	Con	Python does not strictly enforce OOP. It has a weaker encapsulation, meaning it allows for a more relaxed access to object attributes compared to Java, which could potentially compromise data integrity if not managed properly.

Languages		
Java		
Evaluation		
Factor	Pro or Con	Reasoning
Familiarity/Confidence	Pro	Complexity of Java is, in our opinion, semi-simple. It is a high level language where a lot of the functionality is already handled for us such as allocating memory for us in a list. The syntax is fairly straightforward but may require some adjusting, given that InfiTech team members have recently been working predominantly with Python. It should be noted that all team members have recently had experience in a strictly objected-oriented learning course working in Java. This means we are well suited to draw upon proper design practices when using Java
Readability	Pro	Python's syntax is extremely straightforward where a lot of the tasks are a
Familiarity of Existing Libraries	Con	Whilst Java does contain libraries that are capable of creating GUI games, team members do not have any experience with the use of these libraries. This means that there would be more time associates spent on researching libraries, reducing the amount of time spent on development work.
Well Documented	Con	Whilst Java does contain libraries that are capable of creating GUI games, we are less confident that there is widespread use in the IT community. Typically, this means there are less documentation and it could be harder to find solutions to our problems.
Object Oriented Support	Pro	Java enforces strict OOP principles like encapsulation. This leads to well-structured code and minimises risk of data integrity issues.

Selected Programming Language

The technology stack that we will be using is **Python**.

Python was selected due to the simplicity of the language as well as the confidence each member has over the language. Python also provides the team with a very well documented and easy to use library specifically designed for programmers to design games on – Pygames. Although

Python doesn't have complete support for Object Oriented functionalities, it is enough for the capability of this particular task and the fast development time due to readability of code to produce any functionality required from the team. Given the extension to the project will remain within the realm of standard development (no obscure features that might require a separate library), we believe that Python is best suited for this task

Python Supporting Library Evaluation:

Despite referring to Pygame consistently in our evaluation of programming language, we further analyse below 3 Python libraries that could be used for the development of our GUI game.
The libraries evaluated are; Pygame, Tkinter and Panda3D. Reasons for and against their selection are described below. The final decision made on the selected library for Infitech is quoted.

Libraries		
PyGame		
Evaluation		
Factor	Pro or Con	Reasoning
Ease of Implementation for developers	Pro	PyGame is designed specifically for games, offering built-in functionalities like handling sprites, animation, sound and game controllers directly. They also provide better lower-level access for customisation and optimisation. However, most importantly, Pygame benefits from a large community, with plenty of tutorials, documentation and example code available online allowing developers to efficiently and optimally problem-solve during sprints. This could help save time in troubleshooting pygame related errors, giving developers more time with addressing core functionalities / extensions of the game itself
Ease of use for players	Pro	As Pygame offers more built-in functionalities like smoother animations, better graphics handling and full-screen mode, it can offer a better and more immersive experience for users. As an added bonus, PyGame also offers gamepad control out of the box (console controllers) which can assist users who are more accustomed to console gaming.

Aesthetics	Pro	Pygame allows for greater control over graphics, enabling a widder range of aesthetics and styles from pixel art to more complex graphics. It can even leverage hardware acceleration for smoother animations and special effects. Pygame also supports a true-full screen mode to aid immersion and overall game aesthetics.

Libraries		
Tkinter		
Evaluation		
Factor	Pro or Con	Reasoning
Ease of implementation for developers	Pro	Tkinter is an extremely simple library often used in beginner projects. It’s simplicity lets us make use of its variety of prebuilt functions which help us. The layout management and 2D aspect would make the development easier in terms of the grid.
Ease of use for players	Pro	The simplicity of the 2D interface will allow the players to learn easily how to navigate through the GUI. Simple buttons will help how the game is handled.
Aesthetics	Con	With Tkinter being a fairly old library, alot of the features in terms of appearance are pretty outdated. There are a lot of other better alternative libraries in terms of interface that could be used instead.
Documentation	Pro	Tkinter is a pretty well documented library with a lot of tutorials online on how to use. Due to its beginner friendly use, there is a lot of help that is and has been offered on forums such as stackoverflow.

Libraries		
Panda3D		
Evaluation		
Factor	Pro or Con	Reasoning
Ease of Implementation for developers	Con	Panda3D exhibits compatibility with the Python programming language, making it feasible for our constraints (must code with Python given we selected it). Secondly, there seems to be a strong level of samples and code snippets online, supporting our use. However, quotational accounts suggest that the GUI is not intuitive and can be difficult to work with. Hence, a major factor that contributed to the categorisation of this section being a 'con'. Finally, there is a limited tutorial, with the few examples being too complex for beginners like ourselves.
Aesthetics	N/A	The Panda3D library is most applicable for 3D game GUI implementation. Through no fault of Panda3D itself, our team, Infitech, has chosen to make a 2D game, rendering this feature nullable. In short- we do not utilise the true potential of the Panda3D aesthetics, hence a 'n/a' status.
Documentation	Con	The documentation appears to be somewhat lacking, and given our dependency on it (never used Panda3D prior), this is a significant con for Panda3D.

Selected Libraries

The selected library to be used with the Python programming language was **Pygame**.
This decision was made after considering a range of features, at which Pygame outperforms the other considered libraries, Tkinter and Panda 3D, such as of ease of implementation for developers being strong due to the existence of an extensive community with plenty of tutorials, and documentations which helps developers efficiently troubleshoot any issues, good in-game aesthetics due to built-in functionalities that allows for greater control over graphics and enables a wider range of aesthetics and styles from pixel art to more complex graphics, and ease of use for players, due to built-in functionalities like smoother animations, better graphics handling and full-screen mode, which can offer a better and more immersive experience for users.

In conclusion,

Our technology stack looks as follows:

- Programming Language: Python
- Supporting Library: Pygame

User Stories

The table below describes the user stories created by the team that covers both the basic Fiery Dragons gameplay and initial ideas for the team's own extensions. Time denotes the time this particular user story would take to implement. This lies on a scale of 1 to 5, where 1 denotes that that user story would take minimal time to implement and 5 denotes it would take a long time to implement. Importance denotes the importance of that particular user story. Importance also lies on a scale of 1 to 5, with 1 denoting low importance and 5 denoting high importance. Each user story described was also evaluated against the INVEST criteria.

No.	User Story	Time	Importance
1	As a game player, I want to advance my dragon from its cave or from one cell in the volcano to another in a clockwise direction so that I can make progress in the game to reach back to my starting cave first and win the game.	3	5
2	As the game board, I want to ensure that players advance their dragon card as per the number of positions described in the chitcard they select, so that the game can remain competitive for all players.	3	5
3	As a game player, I want to select a chit-card and see the chit card turn over to reveal its animal. So I can try to remember what card I selected.	2	5
4	As a game player, I want to be able to leave my selected chit-card turned over for the duration of my turn so that I can use it to help me memorise the animal on the card for my future turns.	1	2
5	As a game player, I want to be able to attempt to memorise the **position** of the chit-cards by leaving them in their place throughout the game so that I can utilise my memorisation skills to win the game.	1	4

6	As a game player, I want to have another opportunity to select a different chit-card when I uncover one that matches the animal on my current volcano cell so that I can have a chance to move forward.	2	3
7	As a game player, I want to have another opportunity to select a different chit-card when I uncover a pirate so that I can try to redeem my position toward the cave.	2	2
8	As the game board, I want to ensure that the player moves back the number of cells shown on the pirate card if they select it, so that players are adequately challenged by this game.	2	4
9	As a game board, I want to make sure that before the game, the chit-cards are shuffled so that players cannot rely on any previously obtained knowledge to progress unfairly in the game.	2	4
10	As a game board, I want to make sure that all chit-cards start face down so that players cannot obtain an advantage to win unfairly and have their memory skills tested.	1	5
11	As a game board, I want to ensure that a maximum of 4 players play my game at a given time so that each player has the ability to start in one of the 4 provided caves.	2	5
12	As a game board, I want to ensure that a minimum of 2 players play my game at a given time so that each player has the opportunity to compete with another player	2	4
13	As a gameboard, I want to ensure that if a player selects the right chit–card that allows them to progress the exact number of steps to get back in their cave, they will win the game.	1	1
14	As a volcano card, I want to make sure that only 1 player rests on each of my positions at a given time, so that the game remains challenging.	2	3

15	As a pirate dragon chit-card, I want to make sure that a player moves backward in an anti-clockwise direction so that I can achieve my objective of blocking their progression to their cave	2	3
16	As a gameboard, I want to stop a player from moving into their cave if they have a chit-card that would make them overstep their cave, so that the game remains a challenge until the very end!	2	2
17	As a gameboard, I want to ensure that a player does not move positions if the chit-card they pick leads them to a volcano card already containing another player, so that I can make the game more challenging.	2	4
18	As a gameboard, I want to ensure that the player's move is ended if they select a chit card that does not match the volcano card they reside in, so that the game remains competitive and fun!	2	3
19	As the game board, I want to ensure that the youngest player starts their turn first so that I can determine an order for all game movements, allowing for the game to proceed.	1	1
20	EXTENSION - As a pirate chit-card, I want to be randomise the number of pirates are shown on my chit-card so that the player has a greater risk of moving further back, to keep the game more competitive and imperative to play with skill	2	2
21	EXTENSION - As the gameboard, I want to be able to randomly swap the position of my pirate chit-card with another after being selected so that players can face a challenge	3	2
22	EXTENSION - As a gameboard, I want to vary the number of cells within a volcano from 3 so that there is a variation in how long it will take players to return to their cave.	3	3
23	EXTENSION- As a gameboard I want to be forced to return to the start (in my cave) when I select a pirate card more than 3 times, so that I can increase the requirement for players to use their memory and increase the challenge to them.	4	3

Proposed Extensions

The following section has been developed for the sake of providing a comprehensive understanding to our client of the extensions beyond the basic "Fiery Dragon's" game that will likely be implemented (upon evaluation and client approval).

Each extension has an associated description and a short discussion of the factors to consider for development and game impact.

Extension #1:
Name: Randomising Number of Dragon Pirate's on a Dragon Pirate Chit Card
Description: Of the 4 'Dragon Pirate' chit-cards, the number of Dragon Pirates shown will not be the uniform arrangement of 2 cards with 1 dragon pirate, and 2 cards with 2 dragon pirates. Rather, the 4 chit-cards will display a random number of Dragon Pirates. The range will be between 1 and 5 dragon pirates.
Implementation design: The extension means visual altercation on a randomised basis to each "Dragon Pirate" chit card. The number of Dragon Pirates randomly allocated to the chit-card will need to be visible to the player if they choose to un-turn this chit-card on their move.
Developmental Impacts: We foresee the implementation of this extension to be of moderate effort requirement. Chit cards would exist in the developed executable game environment, we will need to add the element of "randomness" in integer selection between 1-5. Further, and likely to be most impactfully, we require a changing visual representation to the Dragon-Pirate chit-cards upon the random integer allocation.
Benefits to game players: It is understood that effective games are a combination and balance of skill and challenge. This new extension clearly adds to both of these game-dimensions. Greater number of Dragon-Pirates that could potentially send a player further backward, incentivises them to play with more skill, using their memory to do so. The game is secondly more challenging with this extension, as players begin to overtake one another with a few incorrectly memorised chit-cards.

Extension #2: Randomising the number of steps within the eight volcano cards.
Description: At the commencement of a game, the number of steps around a full ring of volcanoes (from cave to cave) will be randomly selected and visually presented to reflect the number.
Implementation Design: The game board will be reflective, visually, of the allocated number of steps a player must make. The position of the caves will remain as exactly ¼ apart from each other (to ensure fair game play for all)
Developmental Impacts: The implementation of this feature will require a highly extensible design, where the number of steps within a volcano card is not hard-coded. The visual implementation via the GUI will need to be flexible enough to account for the changing game board size.
Benefits to the game players: The benefits of this extension includes the increased importance of memorising other players chit card turns right from the get go, when the number of steps around the volcano ring, back to the starting cave is small. This is an increased challenge and skill

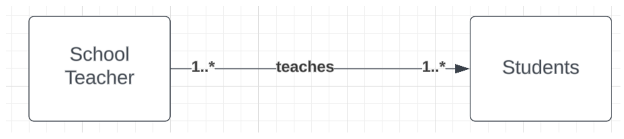
requirement for players, increasing their enjoyment of the game.

Extension #3: Players return back to their starting cave if they pick 3 or more pirate-dragon chit-cards in the game.
<u>Description</u>: If a player chooses 3 Pirate-Dragon chit cards across the duration of their game, they are forced to return back to their cave and restart the game.
<u>Implementation Design</u>: This implementation will involve the player being allowed to choose a chit-card, and, upon the selection of their 3rd pirate dragon, being transported back into their cave. Visually, they will move back to their cave after their selection, but within their turn
<u>Benefits to the game players</u>: This extension will seriously incentivise players to use their memory and avoid the pirate-dragon at all costs. It is a skillfully based extension, not allowing luck to interfere with their game-play, which might have disincentivised a player.

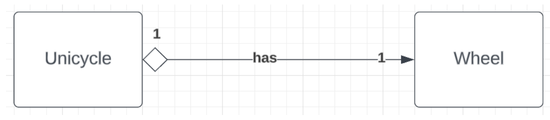
Domain Model Design

The first Domain Model we provide below is a representation of the Fiery Dragons problem space without factoring in proposed extensions.
The domain model represents the entities, and associations between them, of the original Fiery Dragons problem sphere. The industry standard notation for Domain Models have been adhered to.
For full clarity of you, our client, we have briefly summarised these notations and the best practice for understanding the models included:

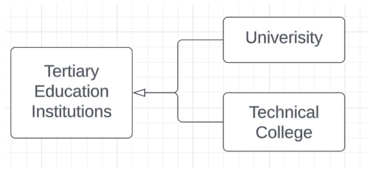
Notation:
Association:



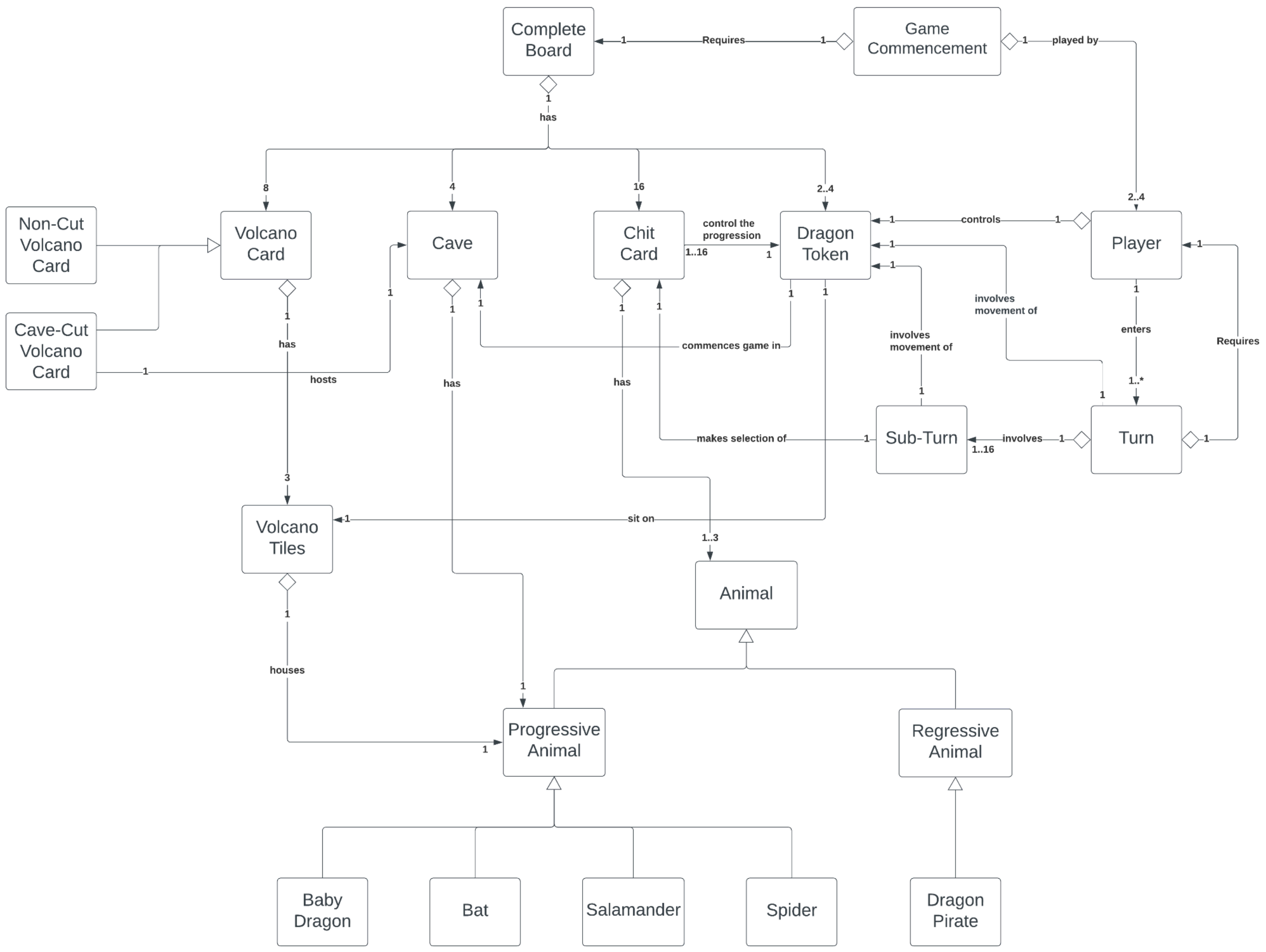
Aggregation:



Generalisation:



Best practice for interpretation of our Domain Model:
2 entities joined by a relationship should be considered from the perspective of arrow-end to arrowhead.
For example, in the ‘Association’ example, 1 to many school teachers (arrow-end entity), teach, 1 to many students (arrowhead entity)



To support the design approach taken for this Domain Model, justification descriptions are provided below.

First, entity justifications are provided.

An entity is defined as something that exists and can be identified as a distinct and independent unit. Further, an entity is an object in the domain that is defined by its identity, rather than its attributes. For an entity to be included in the Domain Model, it must be a part of the problem space.

1. Player Entity

A player in Fiery Dragons represents the person who controls the actions undertaken by the "Dragon Token" entity. A player is well within the definition of an entity, given that a person controlling the game decisions is a distinct unit and fits within the problem space (with no Player entity, there is no action control and no Fiery Dragon game play).

2. Dragon Token

A dragon token in Fiery Dragons game is the visual representation of the player on the board (in this case, GUI screen). The token is an entity because it remains imperative to the playability of Fiery Dragons (without a token, a player cannot account for their and their competitors' progress). Further, the Dragon Token aligns with the notion of having an identity aside from its attributes (a Dragon Token is a visible, tangible object, not defined by its attributes).

3. Chit-Card

The justification of a chit-card is similar to that of Dragon Token. The Chit-Card is a physical object that makes up the "required game board components" of Fiery Dragons as per the game rules. Whilst the Chit-Card does have a range of attributes that provide it core functionality (such as animals) in the game, it's own identity can stand alone (as it can contain different kinds of animals) from these (helping prove its classification of an Entity).
The chit-card is a rounded piece on the board that a player can physically upturn and downturn. It has 2 faces and is clearly part of the problem space (provides the player with an opportunity to advance or retreat it's player token).

4. Cave

A cave is an entity because it is an actualisation of a location within the game board. Further, it is part of the problem space (where a player commences and completes the Fiery Dragon game).
It is defined beyond its attributes, given it can be visually identified on the board.

5. Board

The board itself is also an entity that has many other entities within it (chit-card, cave, player token). The board has a "physical" property that defines it as an entity, beyond the culmination of attributes.

6. Volcano Card

The volcano card houses 3 separate “tiles” that a player token progresses across. It makes up the game board pieces and is a physical object, rather than a described phenomenon. This supports its classification as an entity in our Domain Model.

7. Volcano Tiles

The volcano tile houses 1 “progressive” animal (see below). 3 volcano tiles make up one volcano card and each volcano tile in a volcano card is different from the other volcano tiles in the same card (same face). This is true for a game board instantiation where a volcano card can only face one way. This supports its classification as an entity in our Domain Model.

8. Non-cut volcano card and cut-volcano cards.

These 2 are specific types of volcano cards. Because the volcano card is an entity (see the justification above), the 2 types (distinguished by the features of their own identity rather than their attributes (cut out room for cave or no cut out room for cave)), support them being considered entities themselves.

9. Progressive Animal

Like the volcano was considered an entity (with more specific ‘child’ entities), the same notion exists with the “Progressive Animal” entity in our Domain Model. In this case, a Progressive Animal represents the group of “Baby Dragon”, “Bat”, “Salamander” and “Spider”, all of which will be defined as entities. The ‘progressive animal’ is merely the collection of entities that allow a player token to move clockwise around the Volcano. We consider it to fit within the problem space, given that the Progressive Animal entity facilitates progression in this game. Without it, like many entities, the player could not proceed to try to win Fiery Dragons.

10. Baby Dragon, Bat, Salamander, Spider

These are the specific, distinct parts of the Fiery Dragons game. These entities are abstractions of the parent entity “Progressive Animal”. Despite sharing common attributes with the parent entity, they have an identity beyond this attribute (as a player can only progress forward if the “specific” animal on the chit card matches with their animal on the “volcano tile”), they have a visual representation.

11. Pirate Dragon

A Pirate Dragon is a specific entity. The Pirate Dragon has a visual representation (pirate skull) on the game board. This entity exists in single form: On Chit-Cards. The entity is fundamental to the problem space, it is the only feature in the current game of Fiery Dragons that provides a challenge to a player (acting to cause a player to retreat backwards)

12. Turn

A turn is a specific entity. While the turn is not a “noun” with physical properties, it is central to the game play and “problem space”. Without a turn, there would be no notion of players entering their decision making on selection of chit-cards. In its implementation, without a “turn” entity, we would have a completely static game board.

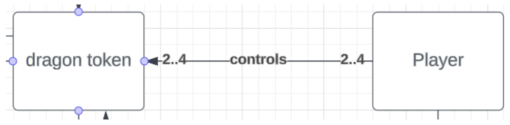
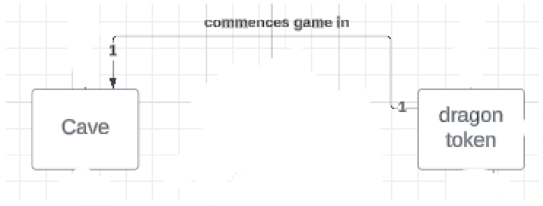
13. Sub Turn

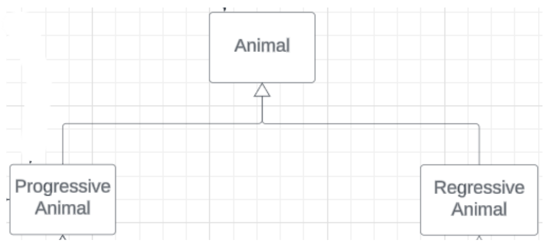
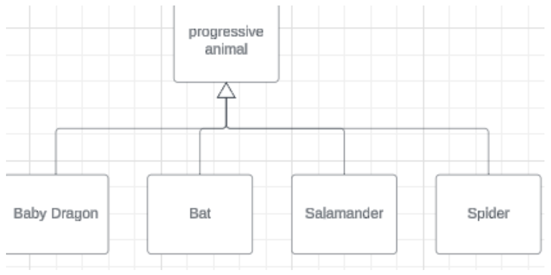
A sub-turn is also a specific entity. Just like Turn, it does not have the physical properties that help to justify its classification. However, the sub-turn is a foundation of the Fiery Dragons Game Play. If a player cannot have a sub-turn, their “turn” in Fiery Dragons is restricted to 1. They could move a maximum of 3 “tiles” forward or 2 tiles back. This would significantly reduce the competitive and interesting nature of the game

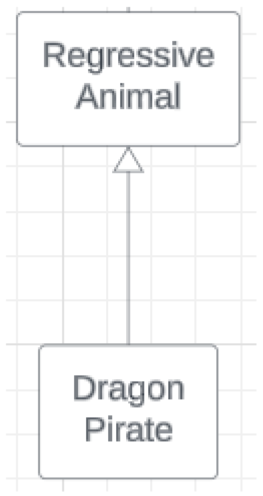
14. Game Commencement

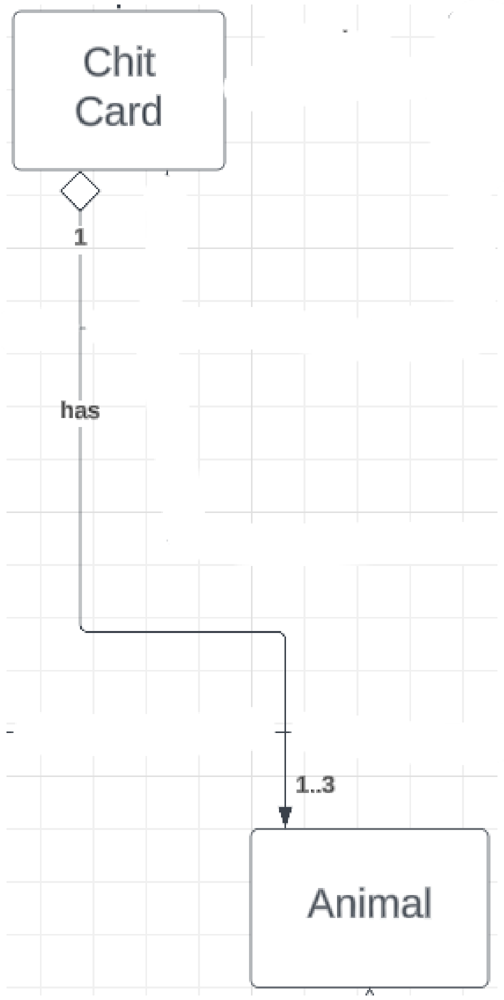
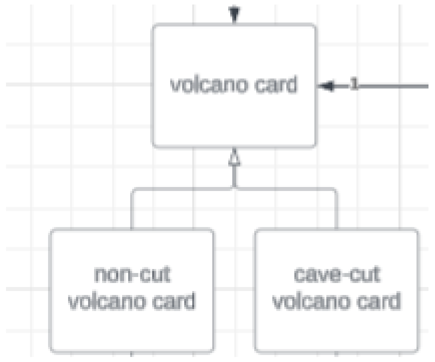
Game Commencement has been considered an entity in our Domain Model. A game commencement is not a physical object/noun. Rather it is the phenomenon of a new “instance” of a game, a new “start” of a Fiery Dragon Play, with players beginning to set up their board, and enter their first turn. Without a game commencement, we would only be able to exhibit 1 attempt at Fiery Dragons, with no ability to “reset” the board for re-play. For these reasons, its centricity to being part of the problem space, we classify “Game Commencement” an entity.

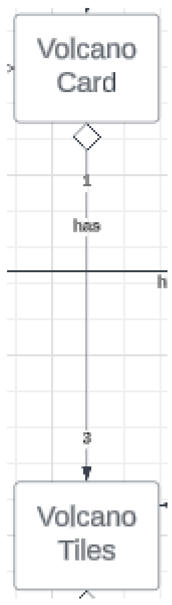
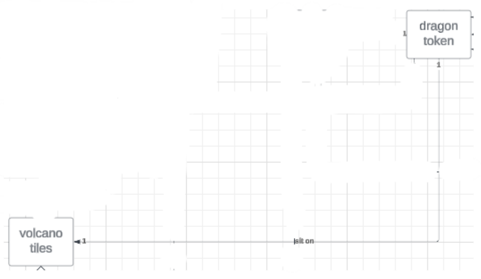
Now, the relationship/association justifications are outlined:

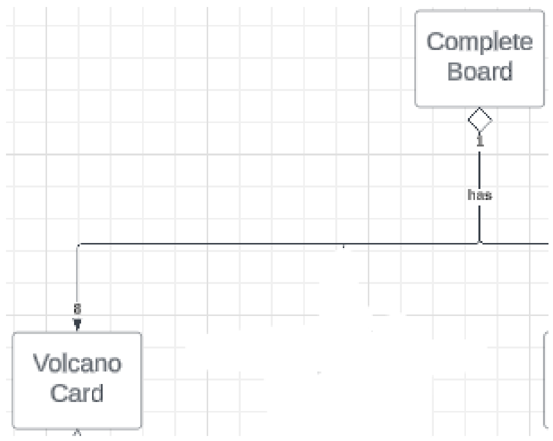
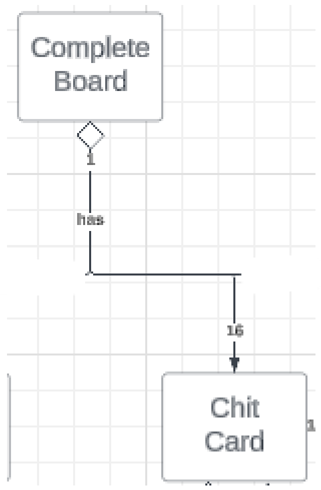
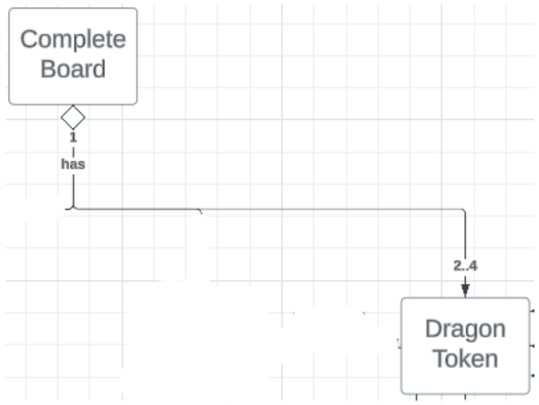
Relationship type	Justification/Rules	Reference
Association	*2 to 4 Players control between 2 and 4 Dragon Tokens individually.* <u>Cardinality Justification:</u> Given that the minimum number of players is 2 and maximum number of feasible players is 4 this cardinality is suitable. <u>Relationship Justification:</u> A player has the capacity to transfer the position of a Dragon Token, hence their direct association with it.	
Association	*Dragon tokens reside in caves at the beginning of the game.* <u>Cardinality Justification:</u> Only 1 dragon token resides in 1 cave of its respective colour at the start of the game. <u>Relationship Justification:</u> A dragon token is related in that the dragon token resides in the cave at the beginning	

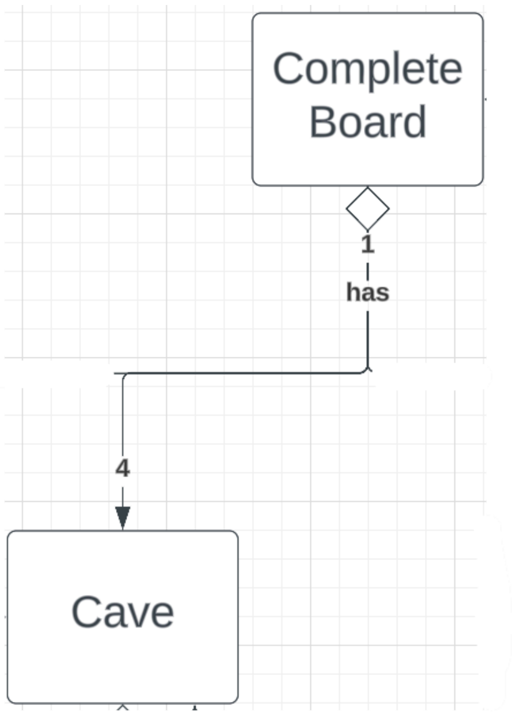
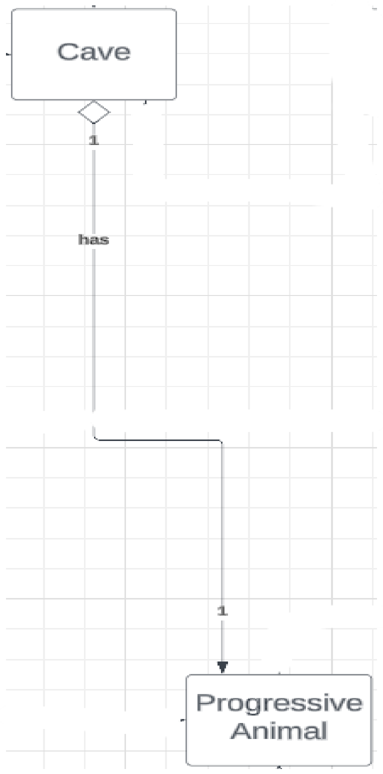
Generalisation	*There are 2 categories of animals, ones that help you progress and ones that regress* <u>Relationship Justification:</u> Despite being different types of animals, they are still animals at the end of the day that dictate the progression of the overall game.	
Generalisation	*Progressive animal will consist of one of the four animals: baby dragon, bat, salamander, spider.* <u>Relationship Justification:</u> These picture cards are what dictate the progression of the game, thus grouped together under "*progressive animal*" (We've excluded the pirate dragon picture card intentionally under this category due to the negative consequences that follow as stated in the above row)	 progressive animal Baby Dragon Bat Salamander Spider

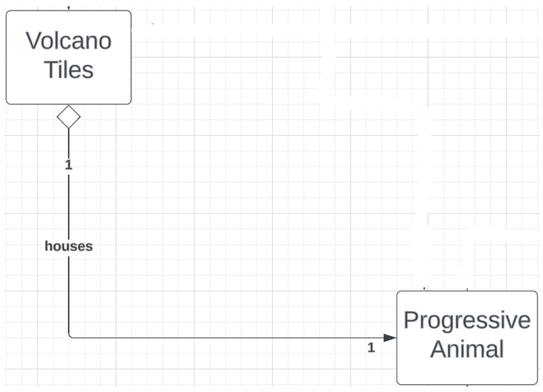
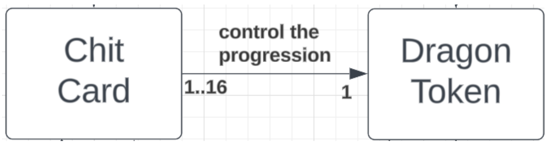
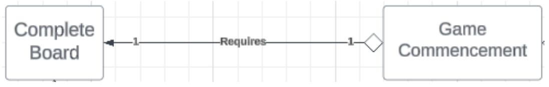
Generalisation	*A dragon pirate falls under the regressive animal category which we defined in the above row* <u>Relationship Justification:</u> Dragon pirate makes you take 1 or 2 steps back in the game, hence it falls under the category regressive animal. This design also allows for extensibility in the future with other regressive animals that hold consequences that apply to players	

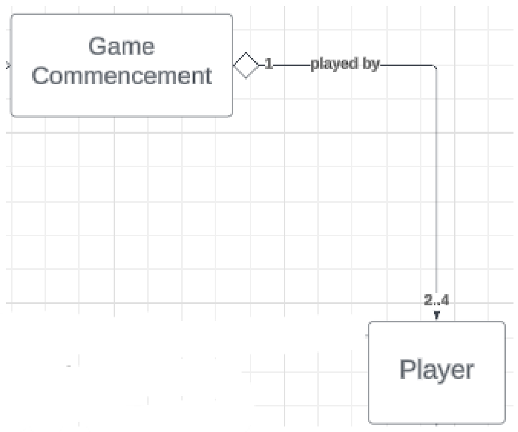
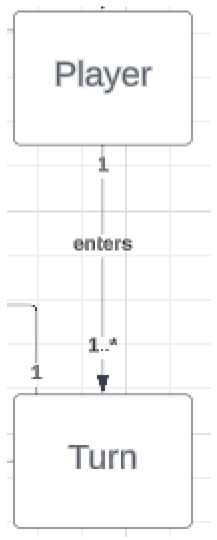
Aggregation	*A chit card will contain a picture of 1 to 3 animals* <u>Cardinality Justification:</u> A chit card can contain 1-3 pictures of either a salamander, bat, spider, baby dragon (refer to assumptions) on the same card. It could also contain 1 or 2 pictures of a dragon pirate. The cardinality 1-3 implies that there doesn't always have to be a maximum of 3 cards, but rather it can. <u>Relationship Justification:</u> A chit card MUST contain an animal on it or else it isn't a chit card - refer to game rules	
Generalisation	*Volcano cards have different types of cards. Cards that concave and cards that don't.* <u>Cardinality Justification:</u> It doesn't and shouldn't have a cardinality because this is a generalisation <u>Relationship Justification:</u> Volcano cards can be split into 2 types, one that has a cave attached to it and ones that don't. The ones that have a cave attached are concave (cave-cut) and ones that don't are non-cut hence the generalisation.	 volcano card 1 non-cut volcano card cave-cut volcano card

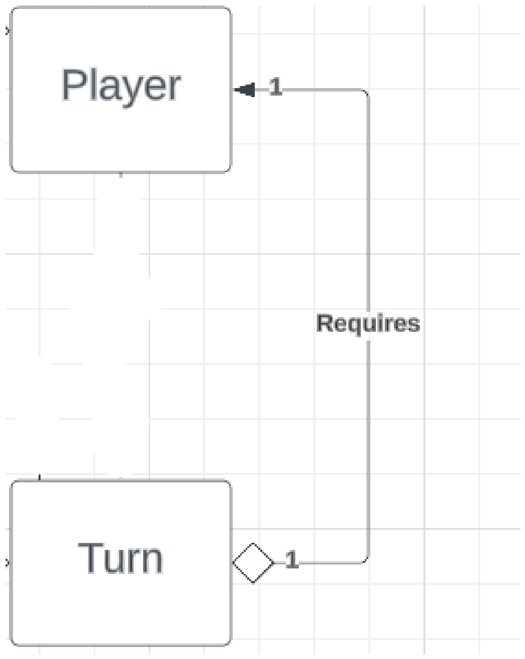
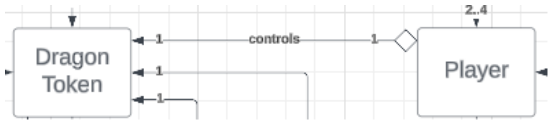
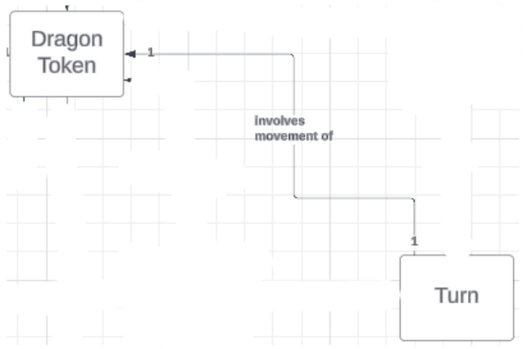
Aggregation	*Volcano card will contain 3 tile cards that a dragon token can sit on* <u>Cardinality Justification:</u> As stated by the game rule a volcano will have 3 volcano tiles per volcano card that can be used. We rejected the notion of 6 tiles on a volcano card because the other 3 cannot be used and were out of scope for our definition of a complete board that is ready to play. <u>Relationship Justification:</u> It is integral that there are 3 tiles in a volcano card for the game to function properly - refer to game rule	
Association	*A dragon token will sit on a volcano tile to progress through with the game* <u>Cardinality Justification:</u> Only one dragon can be on a square at a time - (refer to game rules), hence the 1 to 1 cardinality. <u>Relationship Justification:</u> A dragon token interacts with the volcano tile as the dragon tile will have to sit on a volcano tile at some point in time, hence the association.	

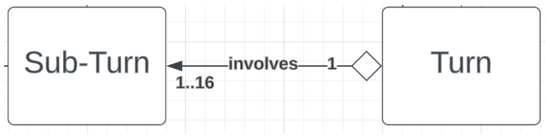
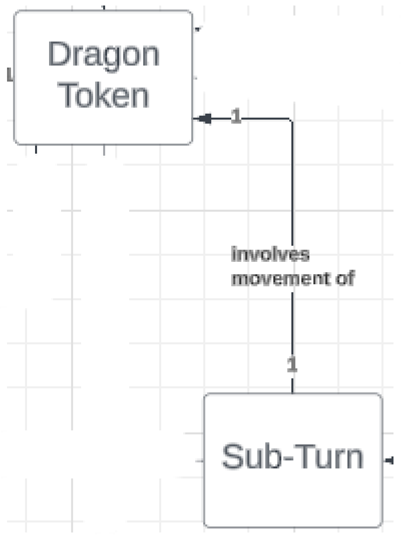
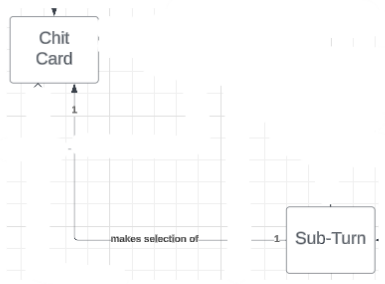
Aggregation	*A board contains 8 volcano cards* <u>Cardinality Justification:</u> For a complete board to be applicable, it will always have to have 8 volcano cards at once on the board. Hence every 1 board will have 8 volcano cards - (refer to game rules) <u>Relationship Justification:</u> A board cannot be complete without 8 volcano cards attached to it therefore the aggregation.	
Aggregation	*A board has 16 chit cards used to be flipped* <u>Cardinality Justification:</u> A complete board will have to have 16 chit cards to have a functioning game - refer to game rules <u>Relationship Justification:</u> A complete board will always have to have chit cards for it to be a functioning game. Hence the aggregation	
Aggregation	*There will be 2-4 dragon tokens on the board at a time* <u>Cardinality Justification:</u> A complete board may have anywhere from 2 - 4 players hence 2-4 dragon tokens. <u>Relationship Justification:</u> A board cannot function without dragon tokens. Dragon tokens are a part of the game, hence the aggregation.	

Aggregation	*A board has 4 caves for it to be a complete board* <u>Cardinality Justification:</u> A complete board has 4 caves for the players to start. - refer to game rules <u>Relationship Justification:</u> A complete board must contain 4 caves for it to be complete - refer to game rules	
Aggregation	*Inside the cave card will contain a picture of a "progressive animal" - define earlier* <u>Cardinality Justification:</u> For every cave, there will only be one "progressive animal" attached to it. <u>Relationship Justification:</u> A cave must consist of one of the following animals (salamander, spider, baby dragon, bat which we generalised to be a "progressive animal") - refer to game rules	

Association	*A volcano card will always consist of either a salamander, spider, bat or a baby dragon (progressive animal)* <u>Cardinality Justification:</u> Each volcano tile will have one "progressive animal" as part of the game - refer to game rules <u>Relationship Justification:</u> For every 1 tile, there should only be 1 "progressive animal" and 1 only - refer to game rules	
Association	*Chit cards determine whether the dragon token progresses to the next volcano card or not* <u>Cardinality Justification:</u> A dragon token's progression of the game is controlled by the amount of chit card the player turns over correctly and there is no limit to the amount of chit cards the player can turn correctly except for the amount of chit cards on the board. Therefore hypothetically all the chit cards on the board can be turned. <u>Relationship Justification:</u> The player interacts with the chit card in the hopes that they can progress their dragon token. The chit card will then determine whether the dragon token advances or remains stagnant.	 Chit Card control the progression 1..16 1 Dragon Token
Aggregation	*For a game to commence, it requires* <u>Cardinality Justification:</u> For a game to occur, you only require 1 complete board hence the 1 to 1	

	<u>Relationship Justification:</u> A game commencing MUST have a complete board or else the game won't be able to commence	
Aggregation	*For a game to commence, it requires 2-4 players* <u>Cardinality Justification:</u> A game must require at least 2 players to function but only a max of 4 players can play at once - refer to game rules <u>Relationship Justification:</u> A game cannot commence without a player hence the aggregation	
Association	*A player will enter a turn to progressive with the game* <u>Cardinality Justification:</u> A player can hypothetically have an infinite amount turns cause there is no limit to how many times they can turn the chit card. They could also finish the game in 1 turn assuming that they are really lucky. <u>Relationship Justification:</u> A player must have a turn to progress with the game, but player and turn are independent of each other as when the game ends, a player still exists but they do not have anymore turn hence the association.	

Aggregation	*A turn requires a player* <u>Cardinality Justification:</u> A turn requires 1 player for a turn to function and exist. For a different player, it would be a different turn and when the turn ends, it will be a completely new turn which is why the 1 to 1 relationship <u>Relationship Justification:</u> A turn requires a player to exist or else there is no turn. The turn is for that player to make decisions and progress or regress in the game.	
Association	*A player controls the movement of a dragon token* <u>Cardinality Justification:</u> 1 player controls the movement of 1 dragon token at a time since the dragon token represents a player <u>Relationship Justification:</u>	
Association	*A turn will dictate the movement or inaction of a dragon token* <u>Cardinality Justification:</u> Every turn will dictate the movement of a dragon token <u>Relationship Justification:</u> The movement of a dragon token can only occur during their turn. So without their turn, the dragon token cannot move	

Association	*A player will get another turn if the condition is met* <u>Cardinality Justification:</u> Given that the player turns over all the cards correctly, the amount of sub turns they have should be reflective of the amount of chit cards on the board <u>Relationship Justification:</u>	
Association	*A sub-turn will dictate the movement or inaction of a dragon token* <u>Cardinality Justification:</u> Every sub-turn can decide the movement of a dragon token <u>Relationship Justification:</u> The movement of a dragon token can only occur during their turn and if they get multiple turns, their sub-turn. So without their turn, the dragon token cannot move	
Association	*During a sub-turn, the player can and will flip a chit card* <u>Cardinality Justification:</u> For every sub turn, the player can flip over a chit card. If they choose the correct card, they get another sub turn. Hence for every sub turn, they can flip over a chit card <u>Relationship Justification:</u> Because during a sub-turn the player must turn over the chit card to determine what happens next in the game.	

Assumptions

A few assumptions made during the creation of this domain model include that, whilst the cardinality between chit-cards and “Dragon Pirate" (“Animal”) is 1 - 1..3 in the team’s domain model, there wouldn’t exist a scenario where a chit card contains 3 dragon pirates due to the nature of rules in the game (Only Dragon Pirate x2 and Dragon Prate exists). However the cardinality was left in the domain model due to possible future extensions where a Dragon Pirate x3 card could be added into the game.

Another assumption made during this creation of this domain model includes an assumption made that extensions of the game would contain other types of animals that would not fall under the classification of “progressive” or “regressive” such as an animal that causes a player to skip a turn. Due to such a reason, an aggregation exists where a ‘parent’ “Animal” entity is represented in the diagram.

An assumption made about the characteristics of the animals in the chit-card include that “Animal x3” (eg. Salamander x3) implies 3 animals of that same type grouped together are present in that one chit-card. This resulted in the team creating an association between the chit-card and animal with a cardinality of 1 - 1..3 respectively, implying that between 1 and 3 of the same animal type can be present in a chit-card.

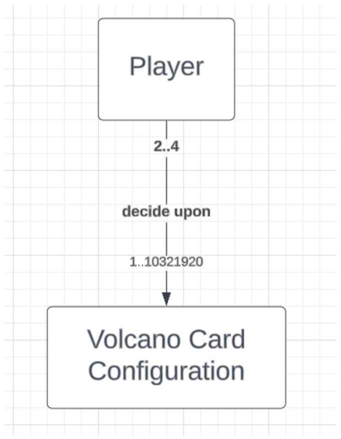
The next assumption made in the domain model was the existence of a “complete board” entity when the game ‘commences’. An assumption is made such that a ‘complete board’ contains volcano tiles that face only one way, resulting in the omission of the opposite side of a volcano card which also contains 3 extra (non-playable) volcano tiles.

The next assumption is that the game will continue until the second last player reaches the cave. This invokes the notion of the game identifying “1st, 2nd, 3rd and 4th” winning positions. This will affect the relationship between a player and its turns. A player can still exist in the game but have no turns (ie, they have “won/finished the game” and now reside in their cave”).

Discarded Domain Model Associations, Entities and Cardinalities.

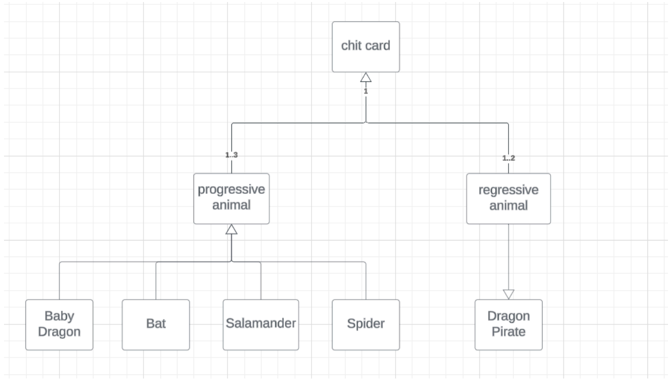
This section will depict the entities, relationships and cardinalities InfiTech has specifically chosen NOT to include in the Domain Model.
Justification of these rejections are provided for clarity of the client.

1. "Volcano Card Configuration" entity, and a relationship with Player : REJECTED

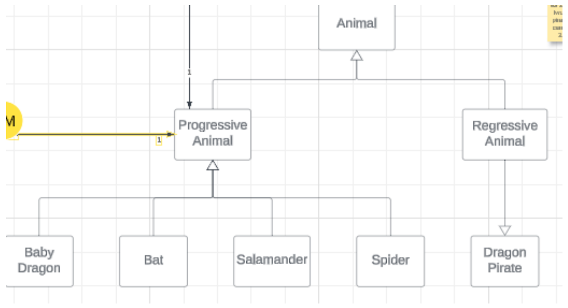


Initially, the concept of configuring the board-game by players was proposed as a feasible relationship and entity for our Domain Model. However, upon analysis, we rejected this idea. Firstly, the act of configuring a volcano card lends itself closer to an "Action" rather than an entity (given that a Volcano Card is already an entity). Secondly, the complex scope of configurations (8! x 256) make it out of the realm of "within the core problem space" of a Domain Model.

2. Chit-Card generalisation with Progressive Animal and Non-Progressive Animal Entities directly : REJECTED



This is a design, where 1 chit card shows between 1 and 3 progressive animal's and 1 to 2 regressive animals was rejected. Firstly, while this does work in the current state of the game, it does not make the addition of new extensions such as a chit-card that shows an animal that neither progresses nor regresses the player simple.
We added a generalisation of "Animal" as shown below to allow for this case. For example, if a new animal was added that makes a player simply miss his/her turn, then this could fall under a new category of animal subclass. It is a generalisation of an animal.



(Selected Design)

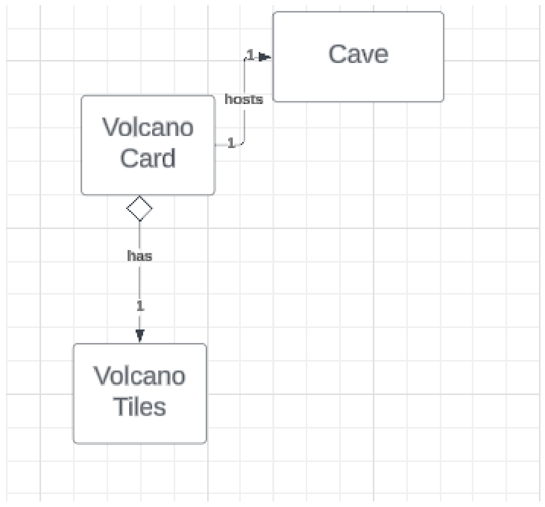
3. REJECTED: 1 type of Volcano Card, rather than a "non-cut volcano card" and "cut volcano card"

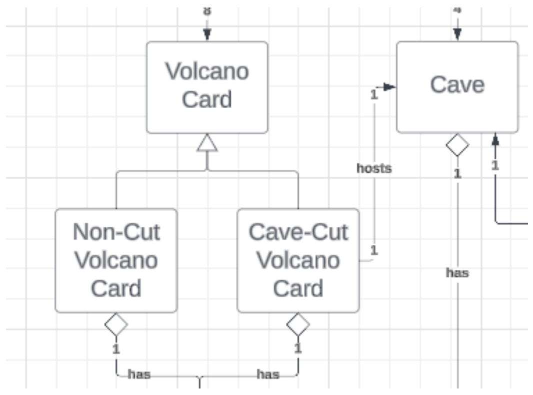
We rejected our initial design which failed to address the 2 distinct volcano card types: Volcano cards that could "host"/"position" a cave within them, and those that could not (based on the board layout specifications).

We added the generalisation in the Domain Model to allow our cardinality of "1 volcano cut card hosts 1 cave", over the incorrect relationship of "1 volcano card has between 0 and 1 cave's".

See the incorrect and correct designs below:

REJECTED: ACCEPTED:





4. REJECTED ENTITY: “Non-Visible Volcano Card”.

We have chosen to reject the notion of a “non visible volcano card”. The reason for this is the fact that our domain model does not reveal the “setup” component. Rather, it looks at the problem space being a “set game board”. This reduces the need for such entities like “non-visible volcano cards”.

5. REJECTED: Generalisation of “Dragon Token” into “Red, Green, Blue and Orange”.

We did not include further specifications of the tokens included in our Fiery Dragons game because their differences lie predominantly in their attributes, rather than identity.
For this reason, not a valid entity, and not to be included in the domain model.

Low-fi diagram

The following range of 5 pages is the basic UI, low-fidelity prototype of our proposed implementation of Fiery Dragons. The descriptions and captures follow a typical setup, gameplay and some unique cases that expose the boundary cases in the game.
The low-fidelity prototype is aimed at providing visual enhancement to further contextualise the final solution to be implemented by Infitech for you, our client.

Below are the a story-board low-fi diagrams of proposed gameplay as a player of the infi-tech implementation of "Fiery Dragons":

Please note that the player Dragon Tokens displayed in the Lo-Fi Diagrams below as "H", "Z" and "S" are NOT the tokens that will be used in the implementation itself (the detailed visual design of the tokens are redundant for clear descriptions and explanations (it is much simpler to understand a letter than a described visual token!)).
The correct Player Dragon Token Designs are presented below.

PLAYER DRAGON TOKEN #1

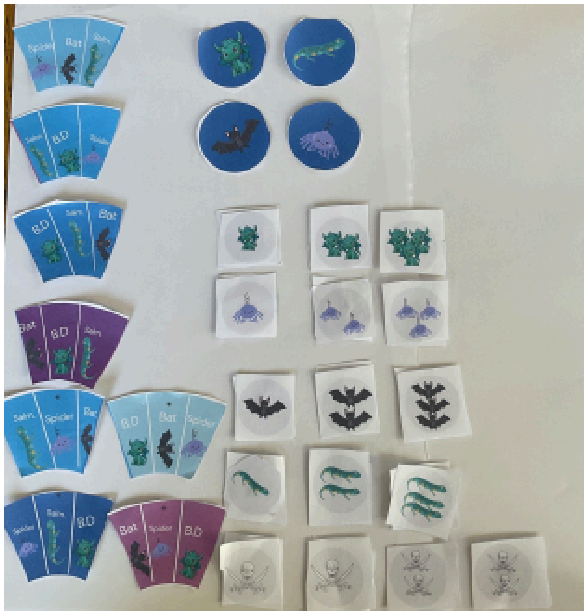
PLAYER DRAGON TOKEN #2

PLAYER DRAGON TOKEN #3

PLAYER DRAGON TOKEN #4

Clearly, each player token is distinguished by the colour of the dragon (red, blue, orange, green)

①

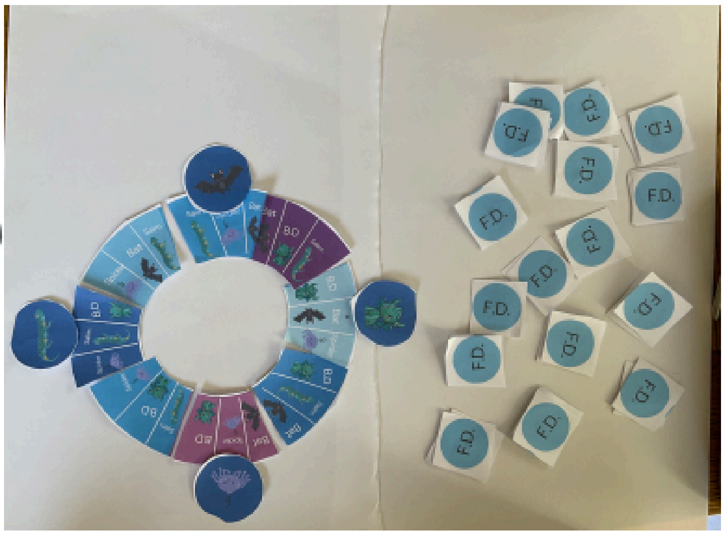


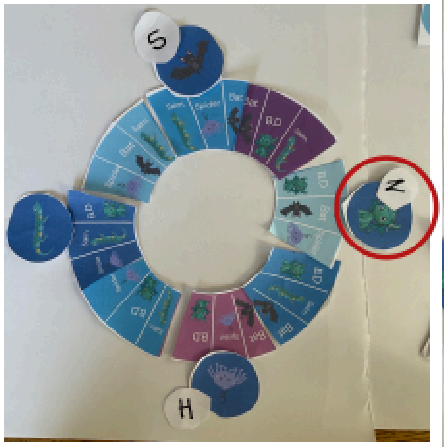
Initially, players check they have the correct inventory:

- 16 "chit cards"
- 4 "cave cards"
- 8 volcano cards
- Max 4 player tokens

②

- Game is arranged in the following way.
- Chit Cards are not visible to any player (F.D. shows)
- Caves are partioned properly

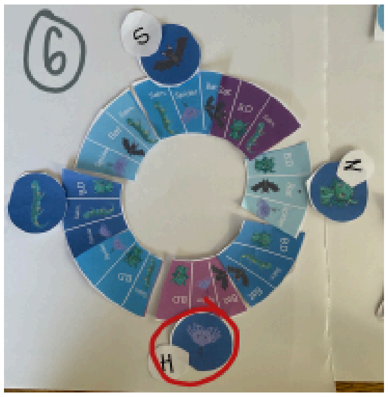


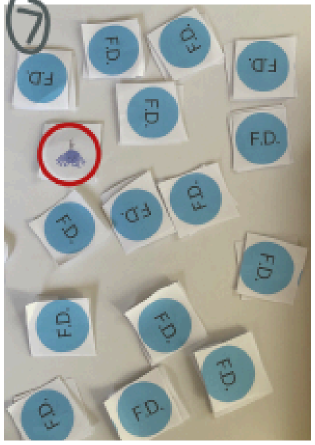
③


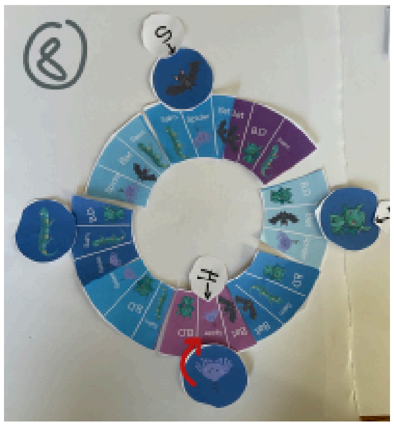
④

⑤

The youngest player, in this case, Ziheng ("Z") goes first.
He is trying to find a "Baby Dragon" chit card,
because he is currently located in the Baby Dragon cave.
He selects a bat; incorrect. He turns it face down. His turn is over.

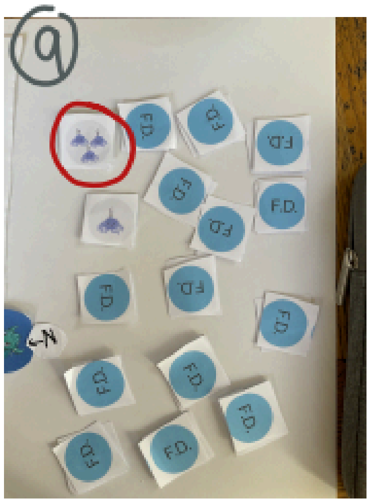
⑥


⑦


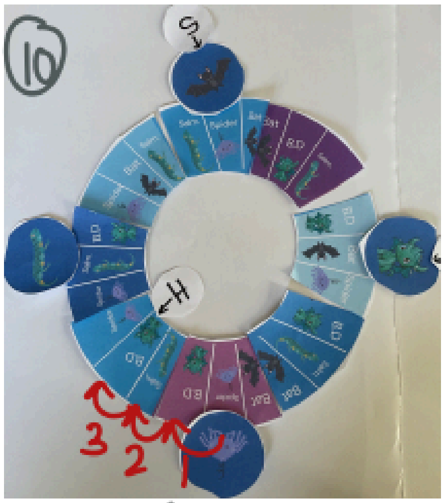
⑧


Harshath's turn now. He correctly selects spider. He advances
his token out of cave and onto Spider dragon card.

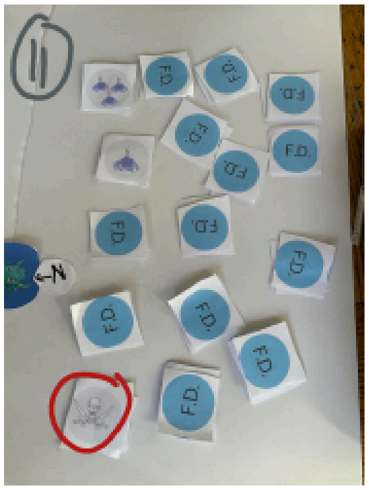
9



10

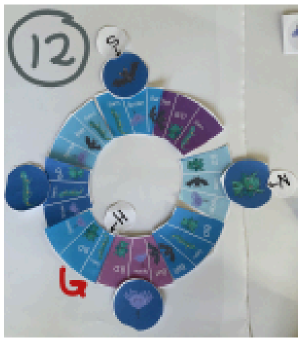


11



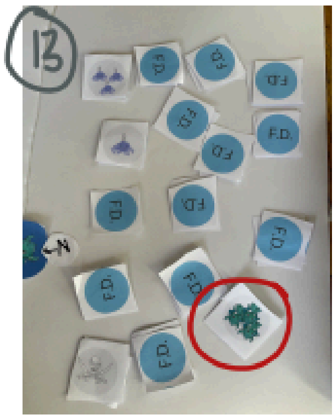
Within this same turn, Harshath chooses another correct chit card - 3 spiders. He moves 3 positions clockwise along dragon cards. Because he picked correctly, he gets another selection.
This time he selects 1 'pirate Dragon'.

12

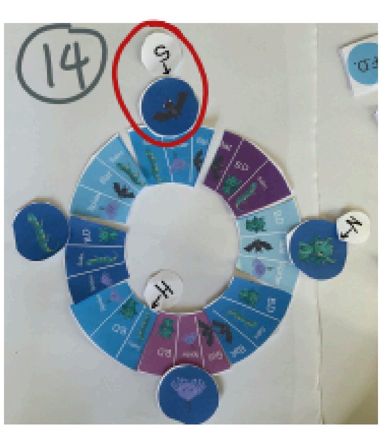
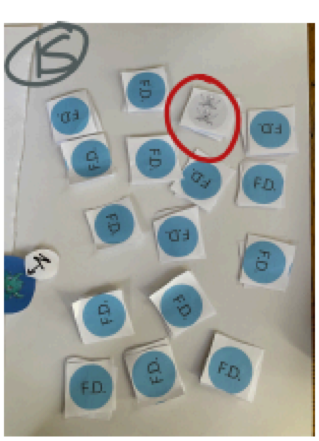
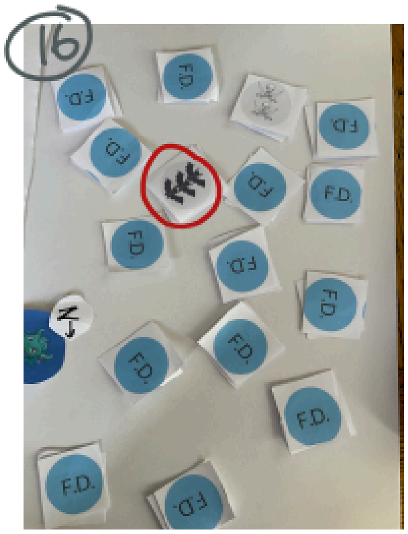


Because of the pirate dragon chosen, H moves back (anti-clockwise) 1 volcano position. He gets another turn (because he picked a pirate dragon)

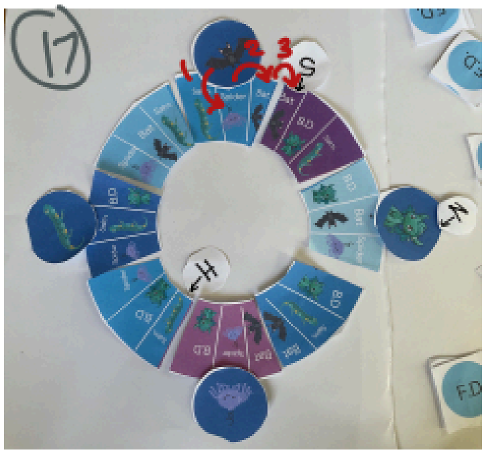
13



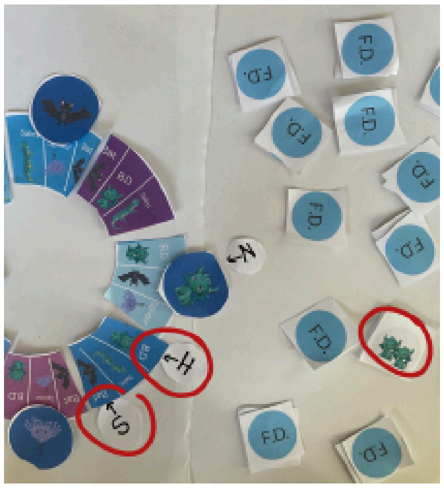
Harshath 'H' does not select a 'Salamander', he chooses a "Baby Dragon' chit card. Hence, H's turn is finally over.

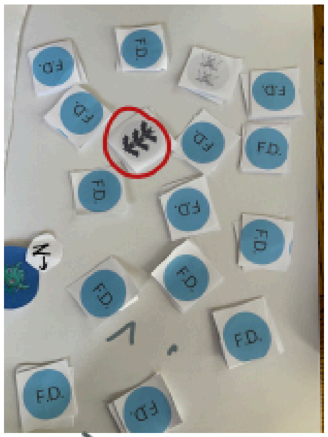
Shannon 'S' turn now. She selects a 2 pirate dragon chit card. She doesn't move as she's still in her cave. Because she chose pirate dragon, she gets another turn. This time, she correctly chooses 3 bat card. She moves out of her cave (1) and 2 places up (total 3).



INDIVIDUAL GAME CIRCUMSTANCES/EVENTS



← In this case, H selected 2 babydragon chit card. However, he does not move his token forward because another player is at the volcano position he would have rested on.



H selects a '3 bat' chit card.
He moves 3 steps into his cave.
If he had selected a chit card that sent him past his cave, he would not have moved

References

1. *Advanced topic - domain modeling* (2023) *Scaled Agile Framework*. Available at: https://www.scaledagileframework.com/domain-modeling/ (Accessed: 23 March 2024).
2. *UI and UX Design: Low-fidelity* (no date) *Codecademy*. Available at: https://www.codecademy.com/resources/docs/uiux/low-fidelity (Accessed: 19 March 2024).
3. Java vs Python – Difference Between Them (2024) www.guru99.com. Available at: https://www.guru99.com/java-vs-python.html#:~:text=Java%20is%20best%20for%20Desktop (Accessed: 27 March 2024).
4. Fiery Dragons Rule Book (https://cdn.haba.de/medias/manual/4498-drachenstark-spielanleitung-6s.pdf (Accessed 6th March 2024)
5. Domain Modelling: What you need to know before coding: Norberto Rodriguez (2021) https://www.thoughtworks.com/en-au/insights/blog/agile-project-management/domain-modeling-what-you-need-to-know-before-coding(Accessed 24th March 2024)

Contribution Log

Task	Date Commenced	Student	Time Spent	Notes
Team Information Section	13th March	Harshath, Ziheng and Shannon Wallis	2 hours	We met in a zoom call on Wednesday 13th March 4-6pm to discuss and implement the first section of the Assignment 1 Sprint (Team Information.
Team Overview	13th March	Harshath, Ziheng and Shannon Wallis	2 hours	We met in a zoom call on Wednesday 13th March 4-6pm to discuss and implement the first section of the Assignment 1 Sprint (Team Information.
Team Member Introduction	13th March	Harshath, Ziheng and Shannon Wallis	2 hours	We met in a zoom call on Wednesday 13th March 4-6pm to discuss and implement the first section of the Assignment 1 Sprint (Team Information.
Team Schedule	13th March	Harshath, Ziheng and Shannon Wallis	2 hours	We met in a zoom call on Wednesday 13th March 4-6pm to discuss and implement the first section of

				the Assignment 1 Sprint (Team Information.
Workload Management	13th March	Harshath, Ziheng and Shannon Wallis	15 minutes	We organised a “when to meet” link and each plugged our own availabilities in
Technology Justification + Decision (Python)	17th March	Harshath Ziheng Shannon Wallis	30 minutes	Had a group discussion on whether to use Python or Java for our design. Did research online to determine which was better for creating a game. Took a look at python libraries as well.
Technology Justification + Decision (Java)	17th March	Harshath Ziheng Shannon Wallis	30 minutes	A group discussion on why we didn’t choose Java. Discussed about the pros and cons as well as look up Java GUI libraries
Technology Justification + Decision (Pygame)	17th March	Harshath	30 minutes	Spent my time researching the Pygame library, researched the pros and cons of going with this approach
Technology Justification + Decision (Tkinter)	17th March	Ziheng Liao	40 minutes	Spent time googling the reasons why people used and didn’t use

				Tkinter. Went on YouTube to take a look at tutorials to get a grasp of the UI
Technology Justification + Decision (Panda3d)	17th March	Shannon Wallis	30 minutes	Spent my time researching what Panda3D was and how it would be applied for our Executable. We were in a group meeting at the time and openly discussed what tech we were going to choose.
User Stories	17th March	Ziheng Liao Shannon Wallis Harshath	2 hours	We each wrote approx ⅓ of the user stories. We did this while we were in a meeting together. We wrote the extension user stories together too. We validated each others work by “swapping 3 ways”
Proposed Extensions	17th March	Ziheng Liao Shannon Wallis	20 minutes	Ziheng and I (Shannon) wrote a descriptive paragraph for each of the extensions. IT was simple task
Domain Model Design	21st March	Ziheng Liao Shannon Wallis	4 hours	First, we each made our own

		Harshath		domain model to allow maximal ideas. We then used the collaborative tool “Lucid chat” in Shared Mode. We combined the best entities of our individual models
Entity Justification	21st March	Ziheng Liao Shannon Wallis Harshath	2 hours	We then each justified the entities that we knew the most about. We edited these together as a process of validating our domain model
Relationship + Cardinality Justification	21st March	Ziheng	2 hours	Double checked the work of the domain diagram and made justifications on them respectively. Anything wrong was brought up in the next group meeting.
Assumptions	27th March	Harshath	30 mins	Took notes on any assumptions we made during the creation of teh domain model and expanded on it
Rejected Domain Model Designs	25th March	Shannon	1 hour	I created a subsection to justify the

				domain entities and relationships we discussed throughout our meetings when designing the domain model.
Lo-Fi	25th March	Shannon	2 hours	I used Canva to mock up the tiles used in the Low-Fi model. I then printed this out. Cut it out, and glued the board together. I played the game, took photos and then wrote about it.