Requirements Documentation

Main Page		
Task	**Contributors**	**Notes**
Introduction	Aryan Baria | Mohammed Bayoumi | Maher Fawzi Rammal	Mohammed worked on the overview, Aryan worked on the objectives and Maher did the references.
Team Contract	Dilraj Deogan | Rayan Amir | Aryan Baria | Mohammed Bayoumi | Maher Fawzi Rammal	Drafted the team contract and decided on roles to work on during the project.
Domain Analysis	Maher Fawzi Rammal	Wrote the Domain analysis of the project
Functional Requirements: List of requirements	Mohammed Bayoumi	Wrote functional requirements of the project
Functional Requirements: Scenario Model	Mohammed Bayoumi	Created the Scenario Model for the functional requirements.
Functional Requirements: Use Case tables	Maher Fawzi Rammal| Aryan Baria | Dilraj Deogan	Created the use case tables of the functional requirements
Functional Requirements: Use Case activity	Dilraj Deogan | Rayan Amir | Aryan Baria | Mohammed Bayoumi | Maher Fawzi	Divided the use cases amongst each other and each person did a few use cases.

diagrams	Rammal	
Functional Requirements: Use Case diagram	Aryan Baria	Created the use case diagram of the project
Non-Functional Requirements	Maher Fawzi Rammal	Wrote the non-functional requirements of the project
Summary	Mohammed Bayoumi | Aryan Baria | Maher Fawzi Rammal	Wrote a summary of the document describing it in a paragraph

Introduction	
Overview	Virtual pets are digital animals that simulate caring for a real pet by assigning users a range of responsibilities such as feeding, grooming, and interacting with their pets. Popular games such as Tamagotchi, Neopets, Nintendogs, and Bitzee are examples of virtual pets. The reason why they were popularized was their ability to engage users through simple, repetitive tasks that grow more rewarding over time as the pet's well-being reflects the user's care. Considering this, with the use of Java and other Computer science tools, our goal is to design and implement this game style. The player will have to take responsibility for the pet while encouraging the users to have constant engagement with the game. The game will also feature a reward system for completing mini-games and an emotional meter that will increase when a mini-game is completed. Although the main aim is to create an engaging and interactive game, there is an educational element where it teaches players simple life skills like taking responsibility and keeping a routine.
Objectives	● Develop a virtual pet game that is both structured and maintainable by applying object-oriented programming strategies. ● Implement requirements such as pet interactions, pet emotions, animations, feeding mechanics, and user experience. ● Implement our design into Java code successfully to capture our decisions into machine code. ● Create a graphical user interface that is both functional and user-friendly. ● Work successfully as a team prioritizing team chemistry and collaboration to push the project success further. ● Write efficient, clean, well-documented Java code that adheres to best practices. ● Reflect and work on decisions made throughout the project to enhance the gameplay experience.
References	● Servos, Daniel. CS2212B Group Project Specification. Version 1.1.0, Winter Session 2025. Owl Brightspace, 2025. ● Draw.io program ● Servos, Daniel. CS2212: Introduction to Software Engineering, Test resources. Quiz 1 resource. Winter Session 2025, Owl. ● Servos, Daniel. CS2212: Introduction to Software Engineering, Week 2 Unified Modeling Language. Winter Session 2025, Owl. ● **"**MB**."** *Cambridge Dictionary*, Cambridge University Press,

https://dictionary.cambridge.org/dictionary/english/mb
- Interaction Design Foundation - IxDF. “What is User Experience (UX) Design?” Interaction Design Foundation - IxDF, 8 Feb. 2025, https://www.interaction-design.org/literature/topics/ux-design.
- W3Schools. “What is JSON?” W3Schools, https://www.w3schools.com/whatis/whatis_json.asp.
- GeeksforGeeks. “Introduction to JUnit 5.” GeeksforGeeks, 4 Jan. 2025, https://www.geeksforgeeks.org/introduction-to-junit-5/.
- GeeksforGeeks. “What is Graphical User Interface?” GeeksforGeeks, 29 Jan. 2024, https://www.geeksforgeeks.org/what-is-graphical-user-interface/.
- **"GitLab."** *TechTarget*, October 2020, https://www.techtarget.com/whatis/definition/GitLab.

Domain Analysis	
What is the software domain for this project?	● The software domain for this project falls under the simulation Java application because users learn how to care for a pet by feeding, dressing, and grooming it. While the game also serves as an educational tool by teaching responsibility, its primary focus on real-life scenarios makes it a simulation game.
Target Audience	● The game is designed for all ages 7 and up, as mentioned in 3.2.3 of the non-functional requirements. While anyone can enjoy the game, the primary audience is children, as they will benefit most from the responsibility-learning aspect. Considering this, the UI and messages prompted will be designed to be simple and easy to follow to respect the age range
Does this project fall into a specific subdomain or category within this software domain?	● This project falls into two categories: ○ Strategy game where players must make strategic decisions to keep their pet healthy and happy. ○ Educational tool, where the game teaches responsibility and simulates real-life pet ownership, helping players prepare for taking care of an actual pet.
What do we know about the domain?	● In most strategy games leaderboards where players compete with friends, motivating them to keep playing. ● In-game purchases & customization, where there are exclusive items (clothes for pets) help to keep interest ● Simulation games are usually fun to play as you progress through and experience the different scenarios ● In overall educational games, what is being played can be implemented in real-life
What are the common issues encountered in this domain (list at least	● Lack of engagement & repetitive gameplay, so for example, if the game lacks variety, it can become boring quickly, especially for children who usually have low attention spans ● There could be a confusing UI or possibly wordy messages that a young audience might not

3)?	understand or might get frustrated. ● There needs to also be an appropriate way of dealing with situations for example the death of a pet. ● .There could be no beta testing for new modes or no tutorial that will help guide the player
What are the common solutions to the above issues in this domain?	● Frequent updates such as introducing new features and challenges keep players engaged. ● User feedback and beta testing, which can help in collecting feedback and running beta tests help refine game mechanics. ● As explained above, the UI and messages prompted will be designed to be simple and easy to follow to respect the age range ● If an action is implemented to ask the age of the user, then based on this age, we can change the story mode slightly so that when the death of a pet happens it is not as tragic for young users ● In terms of following best practices, a great example to follow is The Sims, which has successfully maintained long-term player engagement with updates, personalization, and goal-based gameplay.
How can we use this domain understanding to improve or accelerate development of this project?	● By using this domain as a whole, we can understand what is suitable for the range of the audience, what the main goal/life lessons we want them to learn, and how the progression will be made. By utilizing this, we will be able to integrate different aspects smoothly and not have to worry about leaving out scenarios which can impact the player negatively. Or possibly including scenarios specific to an age group for example as mentioned above, the idea of the pet's death would be handled differently for a younger audience (maybe something like sending the dog away to a vet school). Overall this will limit the amount of mistakes, obstacles, and negative user feedback.

Functional Requirements		
3.1.1	User Interface	● Application needs to have a GUI that is easy to navigate. ● The game must have at least 5 different screens including the main menu screen, gameplay screen, tutorial screen, load/new game screen and a parental control screen. ● Interface must support mouse-based interactions enabling players to navigate through the game's screens, make selections, and control game elements primarily with a mouse ● Interface should have keyboard shortcuts and commands for common actions to make it easy to play.
3.1.2	Main Menu	● Game must have a main menu screen that displays the title of the game and options to the user to choose from. ● Main menu should also display a visual that is representative of the game, list the names of the developers, group number, etc.
3.1.3	Instructions/Tutorials	● Must include at least one screen with detailed instructions on how to play the game and use this application This can be text based, include pictures, or be an interactive tutorial. In all cases it must be comprehensive enough to inform players with no previous experience, how to play our game.
3.1.4	New Game and Pet selection	● When a new game is selected, the player is presented with a choice of picking from at least 3 different virtual pet types. An image representing each pet type should be displayed on the screen and well as some basic information about each pet type. The user should be able to select one of the pets and give it a name. ● Each pet should have slightly different characteristics that impact gameplay, for example one pet might get hungrier quicker or may get happier with less gifts/games
3.1.5	Save/Load game	● The game must feature a Save Game State mechanism that specifically captures the player's progression and the current state of the pet. ● Vital statistics about the pet such as the type of pet being used, its name, its health state, etc. must be included to allow players to resume their game play when they exit out the game.

		● Users could either manually save the game by adding a feature in the GUI or automatically save it once certain check points are reached. A clear message should be displayed to confirm that the game has been saved. ● A player can have multiple pets that they can switch between.
3.1.6	Vital statistics and Rules	● Health, sleep, fullness and happiness should all be displayed on the gameplay screen. - Health should be a positive numerical value that shows how healthy the pet is. - Sleep should also be a numerical value that shows how tired a pet is - Fullness is also a numerical value thar shows how hungry a pet is - Happiness is also a numerical value that shows how happy the pet is ● Each statistic should have a maximum value that could be different for each pet. All of the statistics except health should slowly decline at a set rate during game play. This rate may be different for each type of pet and for each statistic ● Once a statistic is less than 25% of the maximum value, a warning should be visually indicated to the user such as changing the color of the text or progress bar to red, flashing the text or progress bar, making a sound, etc. ● If a statistic reaches 0 the following would happen for each statistic: - For Health, the pet dies and the game would be over, and user will only have 2 options to either load or start a new game. - For Sleep, a health penalty will be applied and the pet will fall asleep and can no longer be interacted with. Other statistics will decline normally. - Fullness, the pet enters the hungry state and the rate that happiness declines should be increased. Health points should also start decreasing until the pet exits the hungry state. - Lastly, Happiness, the pet will enter the angry state. In the angry state the pet will refuse all commands of the player except ones that increases happiness until the bar is at least half full.
3.1.7	Commands	● The player interacts with the pet by issuing commands and they are dependent on the pet's state. At minimum, the following commands should be included: go to bed, feed, give gift, take to the vet, play and exercise.
3.1.8	Player Inventory	● The game should keep track of food items and gift items. Three items from each category should be included. ● The inventory should display a list of the items currently in the player's inventory including a count of how many they have of each item type

		● The game should include a mechanic that allows the player to obtain items
3.1.9	Keeping score	● The game should keep track of the player's score. The score should start at zero and increase when the player does a positive action such as feeding the pet, giving it a gift, or playing with it.
3.1.10	Pet Sprite	● The pet should be represented visually by a sprite. The sprite can be a premade sprite that can be found online. ● The sprite is not required to be fully animated but should switch periodically on a timer or randomly with another sprite to give the illusion of movement
3.1.11	Parental Controls	● The game should provide a separate area accessible via the main menu that provides controls and statistics for the parent of the player. The screen should be password protected. (it can be hardcoded no need for an account)
3.1.11.1	Parental Limitations	● Parents should be able to set certain times of the day where the player is allowed to play the game. If the feature is enabled and the restrictions are active, the player should get a message about not currently being allowed to play
3.1.11.2	Parental statistics	● Game should keep track of the players total play time as well as the average play time per session and display this information in the parental controls screen. ● These statistics should be saved and persist if the application is closed and reloaded. Any time the application is running should count towards the total play time. ● The parent should be able to reset the play times
3.1.11.3	Revive Pet	● The parent should be able to select a save file or save slot and revive the pet back to normal state with full values in each statistics.
3.1.12	Housekeeping & Error Handling	● Users must be able to exit the app cleanly and data must be saved correctly such that it is available the next time the app is opened. ● If the user is able to minimize and maximize the window, UI elements should scale correctly. ● Any errors should be handled and clear messages in simple english should be shown to the user to help them understand and correct the error. ● There should be a way to navigate through each screen and user should never get stuck on one screen

3.1.13	Additional Feature	● Players will be able to play mini games with the pet which will allow the user to earn awards for the pet. - Different mini games will allow users win different rewards ● For each mini game completed, the happiness bar would increase. - Each mini game will have different amount of happiness on top of the rewards

Non-Functional Requirements	
3.2.1	● GUI application program must be developed and ran in the latest java version and minimum java 23
3.2.2	● Object-oriented programming and design patterns must be implemented to create the game
3.2.3	● Game must be rated e (For everyone), and should not include any form of offensive information or visuals
(Additional requirement)	● Game must not have any political views
3.24	● The application must have a user-friendly GUI with clear labeling and consistent design for easy navigation. The interface should follow best UX practices to ensure a smooth user experience.
3.2.5	● The GUI application program must be able to run locally and require no internet connection to play. In terms of the file format, JSON will be used.
3.2.6	● All tools and libraries must be free and implemented in an effective way so that it is easy to obtain and install. Instructions are needed for the any user to be able to run the program with the respected libraries
3.2.7	● GitLab repository must be used to store/contain all code and files. The work must also be progressively added to the repository.
3.2.8	● GitLab wiki must store/contain the diagrams and other design work
3.2.9	● Any issues or tasks must be informed and tracked using GitLab, and should be worked on progressively
3.2.10	● Code must be documented using Javadoc. Every function/method must have a comment, and each file must have a header comment explaining its purpose, author, etc. If any sources were used from the class, youtube, stack overflow, it should be referenced.
3.2.11	● Most of the code should be unit tested using JUnit 5, except for code that requires GUI interactions
3.2.12	● A consistent coding style and naming conventions (variables, function names, etc.) must be applied across all files
3.2.13	● The application must be executable on Windows 11 with a standard Java installation (Java 23 or newer

Non-Functional Requirements	
	as mentioned above). All team members must be able to compile and run it using the same agreed-upon development environment.
3.2.14	● The application must only modify files within its installation directory and not alter or delete external files.
3.2.15	● Every user action must trigger a visible response. Errors must be handled with clear, professional messages (proofing the user input)
3.2.16	● Total project file size must be under 500 MB
3.2.17	● The application must run efficiently (without excessive resource usage). The interface must also remain responsive without lag or freezing.
3.2.18	● The UI must consider accessibility, allowing keyboard and mouse interaction, logical tab order, and color choices that accommodate colorblind users.
3.2.19	● Code should be maintainable and reusable, structured for easy updates and future modifications.
3.2.20	● All game content, documentation, design work, comments, and communication must be in English. Also the work/communication needs to be clear and free of grammar
3.2.21	● From lectures to quizzes to the textbook, the application must follow sound software engineering principles covered in these.

Scenario Model	
Actor	Player
Description	The player is the primary actor of the game, responsible for taking care of the virtual pet. They would feed, play and monitor the pet's statistics such as its health and happiness. The player will be able to also engage in mini games and earn rewards that can be added in the inventory.
Aliases	User
Inherits	None
Actor Type	Person
Active/Passive	Active
Scenario Model	
Actor	Parent
Description	The parent has the same abilities as the player but with extra control over the game settings. They will have a password protected page where they will be able to set game restrictions on when the player is able to play, and monitor gameplay statistics such as the average time and full gameplay time. The parent will also be able to revive the pet/and or edit the pets stats
Aliases	None
Inherits	Player
Actor Type	Person
Active/Passive	Active

Use Case	
Name	Navigate Game User Interface
Primary actor	Player
Secondary actors	Parent, Game System
Goal in context	The player interacts with the graphical user interface to navigate through the various game screens (main menu screen, gameplay screen, tutorial screen, load/new game screen, and parental control screen) allowing control through a mouse and keyboard. They should be able to use shortcuts and commands to improve gameplay.
Preconditions	The game is running and the player has access to the GUI.
Trigger	The player launches the game and interacts with the GUI.
Scenario	1. On launch, the main menu screen appears allowing for navigation through a keyboard and mouse. The player can start a new game, load an existing saved game, and access settings from this screen. 2. The system will respond to these options respectively displaying the screen that corresponds to the option chosen. 3. On the new game screen, the player will be able to choose a new pet, name the pet etc. 4. On the loaded existing game screen, the user will be able to choose their save file. 5. Both screens will lead to the gameplay screen where the player can interact with their pet. 6. The system will constantly process the player inputs and respond to them by changing the interface.
Alternatives	1. Shortcuts and commands can be used to easily navigate the user interface. 2. The parent can access the parental control screen using a password to set restrictions and monitor statistics. 3. From the main menu, the player has an option to play the tutorial which will open a new screen that teaches the user about the game. 4. The settings menu can be accessed at any time to adjust settings.

Exceptions	1. Unrecognized keyboard inputs will be ignored.
Priority	Highest

Use Case	
Name	Access Main Menu
Primary actor	Player
Secondary actors	Parent, Game System
Goal in context	The player interacts with the Main menu to access different game options. This main menu must be representative of the game containing a visual that captures what the game is about as well as containing general information such as the game name, list of developers, and more.
Preconditions	The game is running and the main menu screen is displayed.
Trigger	The game is launched or the player returns to the main menu screen from another screen.
Scenario	1. The game launches or the player returns to the main menu prompting the main menu screen to appear. 2. The title of the game, visual representation of the game, developer names, and options are displayed. 3. The options displayed consist of start new game, load game, tutorial, parental controls, settings, and exit. 4. These options can be selected using the mouse or keyboard shortcuts and the game system will promptly display the corresponding information.
Alternatives	1. In parental controls, a password will be prompted before accessing. 2. When exiting, a confirmation will be asked for to ensure the user wants to close the game.
Exceptions	1. Unrecognized keyboard inputs will be ignored. 2. Parental control password must be correctly imputed otherwise access will not be granted.
Priority	High

Use Case	
Name	View Instructions/Tutorial
Primary actor	Player
Secondary actors	Parent, Game System
Goal in context	The player accesses the tutorial to learn how to play the game and understand the controls. Must be user-friendly to ensure that anyone can understand and get a grasp on how the game plays.
Preconditions	The game must be running and the player must be in the main menu screen.
Trigger	The player selects the tutorial option from the main menu using mouse or keyboard shortcuts.
Scenario	1. The player selects the tutorial option from the main menu screen. 2. The tutorial screen is loaded and a short simulation of how the game plays is shown to the user allowing them to interact with the game while showing them all the controls and keyboard shortcuts that are applicable in different situations. 3. This tutorial allows the player to progress at their own pace ensuring that all skills are acquired to operate the full game. 4. Once the tutorial is completed the player is taken back to the main menu screen.
Alternatives	1. The player may exit the tutorial before completing it by choosing the exit tutorial option.
Exceptions	1. Any errors that occur within the tutorial will be handled by either resuming the tutorial or returning the user to the main menu screen.
Priority	Medium

Use Case	
Name	Start new game
Primary actor	Player

Secondary actors	Parent, Game System
Goal in context	The player starts a new game allowing them to select a pet from at least 3 options displayed through an image, and a short description about the pet displaying basic information. The player will be able to give their chosen pet a name.
Preconditions	The game must be running and the player must be in the main menu screen.
Trigger	The player selects the new game option from the main menu using mouse or keyboard shortcuts.
Scenario	1. The player selects the new game option from the main menu screen. 2. A screen with different pet options is displayed showing an image to represent each pet as well as basic information on each pet. 3. The player can select one of the available pets from which they will be prompted to enter a name for their chosen pet. 4. The player must confirm their pet and then the game will be started with the selected pet.
Alternatives	1. Before confirming their pet the player will have the option to go back and select a different pet. 2. The player may go back to the main menu using an option that will be displayed.
Exceptions	1. Invalid names such as empty, too long, or invalid characters will show an error message and prompt the user to enter a valid name.
Priority	High

Use Case	
Name	Save/Load Game
Primary actor	Player
Secondary actors	Parent, Game System
Goal in context	Allow players to save their game progress and pet’s state, ensuring they can resume gameplay later, keeping all pet statistics intact.

Preconditions	- Player must have an active game session - A pet must already be selected or created - The save system must be properly implemented and easily accessible
Trigger	- Player manually clicks “Save Game” button in the GUI - System automatically triggers a save at certain checkpoints
Scenario	1. Player selects “Save Game” from the game menu (OR automatic checkpoint is reached) 2. The system gathers all the pet data of the pet (pet type, name, health, hunger, happiness, inventory, game progress, etc.) 3. Data is stored in a specific file 4. Confirmation message is shown to the user 5. When the player chooses “Load Game”, they can choose from saved pets 6. The system loads the saved data and restores the game state 7. Player resumes the game from where they left off
Alternatives	- Instead of a manual save button, the game could rely on autosave - The game could supposed multiple save spots instead of a few
Exceptions	- If the player tries to save the game but they encounter a storage issue, the game should notify them by displaying an error message - If no saved game exists when the player attempts to load, the system should display an error message and prompt the player to start a new game
Priority	Extremely High – This feature is crucial for player retention and ensures game continuity

Use Case	
Name	View and Manage Pet Vital Statistics
Primary actor	Player
Secondary actors	Parent, Game System
Goal in context	Once on the game screen, the player can view and maintain their pet’s health, sleep, fullness, and

	happiness. All statistics except for health should decline over time and when a stat drops below 25%, a warning will be shown. Different negative effects will be applied once a stat reaches 0%. The user should be able to complete tasks to maintain these stats and ensure their pet's well-being.
Preconditions	The game must be running and the player must be in the gameplay screen with their selected pet.
Trigger	The player must have selected their pet and this use case will constantly trigger updating the pet's vital statistics.
Scenario	1. The player is on the gameplay screen with their selected pet. 2. The system will display the pet's stats including their health, sleep, fullness, and happiness on the screen along with the pet. 3. With time the pet's sleep, fullness and happiness will decrease. 4. When sleep reaches 0, the pet will fall asleep and cannot be interacted with. This will also cause the pet to lose some health. During this time the pet's other stats will also decrease. 5. When fullness reaches 0, the pet will lose happiness much faster and will also begin to lose health over time. 6. When happiness reaches 0, the pet will enter an angry state where the pet will refuse all commands from the user except those that will increase happiness until the happiness bar is at least half full. 7. When health reaches 0, the pet will die, causing the current game to be over taking the user back to the main menu screen. 8. The colour of each bar will reflect the current state of the statistic.
Alternatives	1. The player can complete tasks such as feeding, putting their pet to sleep, and playing with their pet to increase their respective stats. 2. When a stat decreases under 25% a warning will be displayed.
Exceptions	1. Interacting with the pet when in a restrictive state (Such as trying to feed the pet when angry) will display a message to the user notifying them of the pet's current state and what they can do to help the pet reach normal vital statistics.
Priority	High

Use Case	
Name	Commands
Primary actor	Player
Secondary actors	Parent, Game System
Goal in context	The player interacts with their pet using different commands that affect the pet's behaviour. Through the use of commands, the user is allowed interactivity in maintaining the pet's progress in the game. (For example, a command to feed when hungry)
Preconditions	The game must be running in an active session and the player must have at least one pet (alive) in the game.
Trigger	With the cursor the player selects a command button from the game menu, and will be prompted to follow instructions to execute that command
Scenario	1. The player is on the gameplay screen with their respected pet 2. The player selects a command, for example, "Play" 3. The game system retrieves the current state of the pet For example, The pet is bored and has low happiness. 4. The system checks if the command is valid based on the pet's current state. Valid example would be that the pet is bored, so it accepts the play command. Invalid example would be that the pet is already tired, so it refuses to play. 5. If the command is valid, The system executes the command and an animation may allow the user to play with their pet. As a result, the pet's state is updated (For example, happiness increases, but energy decreases slightly). Then we will let the user know, for example saying, "Your pet enjoyed playing! Happiness +10, Energy -5" 6. If the command is not valid, we display an error message, for example, "Your pet is too tired to play! Let it rest first." 7. If the command has lasting effects, those effects are gradually applied over time, for example, If the pet is overfed, it may gain weight and become sluggish. 8. The player can pick a new command.

Alternatives	In the scenario we handled cases where the state of the pet might not match the command (go to bed, when not tired, or feed when full or take to vet when already healthy) then we display a message to let the user know
Exceptions	There will be a cooldown (lock commands) if the user keeps pressing commands that will not activate. Another Exception could be if anything goes wrong when loading in the pet data/vital stats
Priority	Highest

Use Case	
Name	Player Inventory
Primary actor	Player
Secondary actors	Parent, Game System
Goal in context	The player has an inventory system that keeps track of food items and gift items. The inventory should allow the player to view, manage, and obtain items as they progress in the game. For example, the player buys food to feed their pet or gifts to give
Preconditions	The game must be running in an active session, and the player must have an inventory available
Trigger	The player accesses the inventory menu from the game interface or obtains a new item.
Scenario	1. The player is on the gameplay screen and selects the inventory option. 2. The inventory menu is displayed, showing a list of available items and their quantities. For example, 3 Apples, 2 Bones, 1 Teddy Bear. 3. The player selects an item to use or inspect. For example, selecting "Apple" shows details such as "A healthy snack for your pet." 4. If the player uses an item, the system updates the inventory count. For example, using "Apple" decreases the count from 3 to 2.

	5. If the player has no more of a selected item, the system removes it from the inventory list. 6. The game includes mechanics that allow the player to obtain new items. For example, receiving food as a reward or purchasing a gift from a shop. 7. When an item is obtained, the system updates the inventory and displays a confirmation message. For example, "You received 1 Bone! It has been added to your inventory." 8. The player can exit the inventory menu and return to the game.
Alternatives	If the player tries to use an item that they do not have, the system displays a message to notify them
Exceptions	If an error occurs while obtaining or using an item, the system prevents the action and displays an error message
Priority	Highest

Use Case	
Name	Keeping Score
Primary actor	Player
Secondary actors	Parent, Game System
Goal in context	The game tracks the player's score, which starts at zero and increases when the player performs positive actions. The scoring system provides feedback on the player's engagement with their pet. For example, feeding the pet increases the score.
Preconditions	The game must be running in an active session, and the player must have an active pet
Trigger	The player performs a positive action, such as feeding, playing, or giving a gift to their pet
Scenario	1. The player is on the gameplay screen with their pet. 2. The player performs a positive action, such as feeding the pet, giving a gift, or playing with

	it. 3. The game system detects the action and verifies if it qualifies for a score increase. For example, feeding the pet when it is hungry is valid. 4. The system updates the player's score based on the action performed. For example, "+10 points for playing with your pet!" 5. A confirmation message is displayed to notify the player of the score change. For example, "Your pet had fun! Score +10." 6. The updated score is reflected in the game's UI. 7. The player can continue interacting with the pet, accumulating more points through additional positive actions.
Alternatives	If the player repeats the same action excessively within a short time, the game may limit score increases to prevent exploitation, for example, "You have already fed your pet, Wait before feeding again."
Exceptions	If an error occurs in the score calculation or display, the system prevents incorrect scoring and may display an error message, for example, "Error updating score. Please restart the game."
Priority	Medium to High

Use Case	
Name	Pet Sprite
Primary actor	Player
Secondary actors	Parent, Game System
Goal in context	The pet is represented visually by a sprite that changes periodically to create the illusion of movement. The sprite enhances player engagement by providing a visual representation of the pet. For example, the pet sprite may switch between an idle and a blinking pose.
Preconditions	The game must be running in an active session, and the player must have an active pet.

Trigger	The pet sprite updates either on a timer or randomly during gameplay.
Scenario	1. The player is on the gameplay screen, and their pet is visible. 2. The game system loads and displays the pet's assigned sprite. 3. A timer runs in the background or a random event occurs, triggering a sprite change. For example, every few seconds, the sprite updates. 4. The system selects a new sprite variation to display. For example, switching from a neutral face to a blinking face. The pet's appearance updates on the screen, creating the illusion of movement. 5. This process continues periodically while the game is running. 6. If the pet interacts with the player (such as playing or eating), the system may also update the sprite to reflect that action. For example, a happy expression when receiving food.
Alternatives	If the player adopts a new pet or switches pets, the system loads the new pet's corresponding sprite set.
Exceptions	If the pet sprite fails to load, the system displays a placeholder or error message. For example, "Error loading pet sprite. Restart the game to fix this issue."
Priority	Medium to high

Use Case	
Name	Parental Controls
Primary actor	Parent
Secondary actors	Game System
Goal in context	The game provides a parental control section where parents can access controls and statistics related to their child's gameplay. This ensures that parents can monitor and manage gameplay activity. For example, a parent can view how long their child has played and adjust certain settings.

Preconditions	The game must be running in an active session, and the parental controls section must be accessible from the main menu.
Trigger	The parent selects the parental controls option from the main menu.
Scenario	1. The parent is on the main menu and selects the parental controls option. 2. The system prompts for a password before granting access. 3. The parent enters the password, and the system verifies if it is correct. 4. If the password is correct, the parental control screen is displayed. 5. The screen provides various statistics and settings. For example, the parent can view the total playtime, how frequently the child interacts with the pet, or adjust volume and screen time limits. 6. The parent can navigate through the options and make changes if applicable. 7. Once finished, the parent exits the parental control screen, returning to the main menu.
Alternatives	If the parent enters an incorrect password, the system displays a message and allows retry attempts.
Exceptions	If an error occurs while loading the parental control screen, the system prevents access and displays an error message. For example, "Error loading parental controls. Please restart the game."
Priority	Medium to high

Use Case	
Name	Parental Limitations
Primary actor	Parent
Secondary actors	Game System
Goal in context	The game allows parents to set time restrictions on when the player can access the game. If

	restrictions are active, the player is notified that they are not allowed to play. For example, a parent can set playtime from 3:00 PM to 6:00 PM, and outside of this window, the game will block access.
Preconditions	The game must be running in an active session, and the parental control settings must be configured with active time restrictions.
Trigger	The player attempts to start the game during a restricted time period.
Scenario	1. The parent accesses the parental control section from the main menu. 2. The system prompts for a password, and the parent enters it. 3. The system verifies the password and grants access to the parental settings. 4. The parent selects the "Set Playtime Restrictions" option. 5. The system provides an interface for setting allowed play hours. For example, the parent selects 3:00 PM to 6:00 PM as the permitted time window. 6. The system saves the settings and enforces them automatically. 7. The player later tries to launch the game outside of the allowed time period. 8. The system checks the current time and determines that the player is not allowed to play. 9. A message appears on the screen notifying the player. For example, "You are not allowed to play at this time. Please try again later." 10. The game prevents access until the allowed playtime begins.
Alternatives	If the parent wants to temporarily override the restrictions, they can enter the parental control menu and disable the limitation
Exceptions	If an error occurs while enforcing time restrictions, the system prevents access and displays an error message. For example, "Error enforcing parental limitations. Please restart the game."
Priority	Medium

Use Case	
Name	Parental statistics
Primary actor	Player

Secondary actors	Game System
Goal in context	The game tracks the players total playtime and average playtime per session and displays this data in the parental controls screen. This allows parents to monitor gameplay habits and manage screen time. For example, a parent can check that their child has played for an average of 45 minutes per session
Preconditions	The game must be running in an active session, and parental statics must be accessible from the parental controls menu
Trigger	The player selects the parental controls option from the main menu and navigates to the statistics section
Scenario	1. The parent accesses the parental control section from the main menu. 2. The system prompts for a password, and the parent enters it. 3. The system verifies the password and grants access. 4. The parent navigates to the statistics section. 5. The system retrieves and displays the player's total playtime and average playtime per session. For example, "Total Playtime: 10 hours, Average Session: 45 minutes." 6. The statistics update dynamically as the player continues playing the game. 7. If the application is closed and reopened, the statistics persist and continue tracking. 8. The parent has an option to reset the playtime statistics. 9. If the parent chooses to reset, the system confirms the action and resets the stored playtime values to zero.
Alternatives	If the parent does not rest the statistics, the system continues tracking
Exceptions	If an error occurs while saving playtime data, display an error message. For example, “Error retrieving playtime statistics”
Priority	Medium

Use Case	
Name	Revive Pet
Primary actor	Parent
Secondary actors	Game System
Goal in context	Allow parents to restore a pet that has died or has low stats by selecting a save file or save slot, bringing the pet back to its normal state with full values in all statistics.
Preconditions	- Parent must have access to the parents control screen - There must be an existing saved game with a per in non-normal state - System must support save file selection and pet revival functionality
Trigger	- The parent access the Parental Control Menu - The parent selects the revive pet option and chooses a save file or save spot
Scenario	1. The parent navigates to the Parental Controls Menu 2. The system prompts the parent for authentication such as password or PIN 3. The parent selects a save file or a save slot which contains the pet that is dead or in low health 4. The system restores the pet to a normal state, and it sets all of the pets statistics to their maximum (including health, happiness, fullness, and sleep) 5. A confirmation message gets displayed saying “Pet has successfully been revived!” 6. The parent now exits the menu, and the player can interact with the pet as normal
Alternatives	- Instead of reviving pets instantly, the system could implement revival cooldown, this means you can only revive a pet once every 24 hours. - The game could require the parent to spend In-game currency or items for the revival instead of it being free.
Exceptions	- If there is no save file or no pet that needs to be revived, the system should display an error message where it says “There are no pets available for revival” - If the incorrect PIN or password is given, the access to parental controls are denied - If the system cannot retrieve the file, then it should provide an error message saying “Saved files could not be loaded, Please try again.”

Priority	Medium - Not too critical, but still important for Parental Control Functionality and also retention of players

Use Case	
Name	Housekeeping and error handling
Primary actor	Player
Secondary actors	Game System
Goal in context	Ensuring smooth application management, allowing users to exit cleanly, save data properly and navigate through all screens without getting stuck. Error messages are in english (not too wordy) to help the user understand and resolve issues.
Preconditions	The game must be running in an active session
Trigger	The player attempts to exit the game, minimise/maximise the window, or navigate between screen
Scenario	1. The player chooses to exit the game. 2. The system ensures all progress is saved correctly. 3. The game closes cleanly, and data is available the next time the app is opened. 4. If the player minimizes and maximizes the game, the UI elements scale correctly to fit the window size. 5. The player navigates between screens, such as moving from the main menu to gameplay or parental controls. 6. If an error occurs, the system detects the issue and displays a clear, understandable message. For example, if a save fails, the system shows "Error saving game. Please try again." 7. If the player attempts to navigate to a broken or unavailable screen, the system prevents the action and provides an alternative. For example, "This screen is not available right now. Returning to the main menu." 8. The player is always able to return to the main menu or another functional screen and

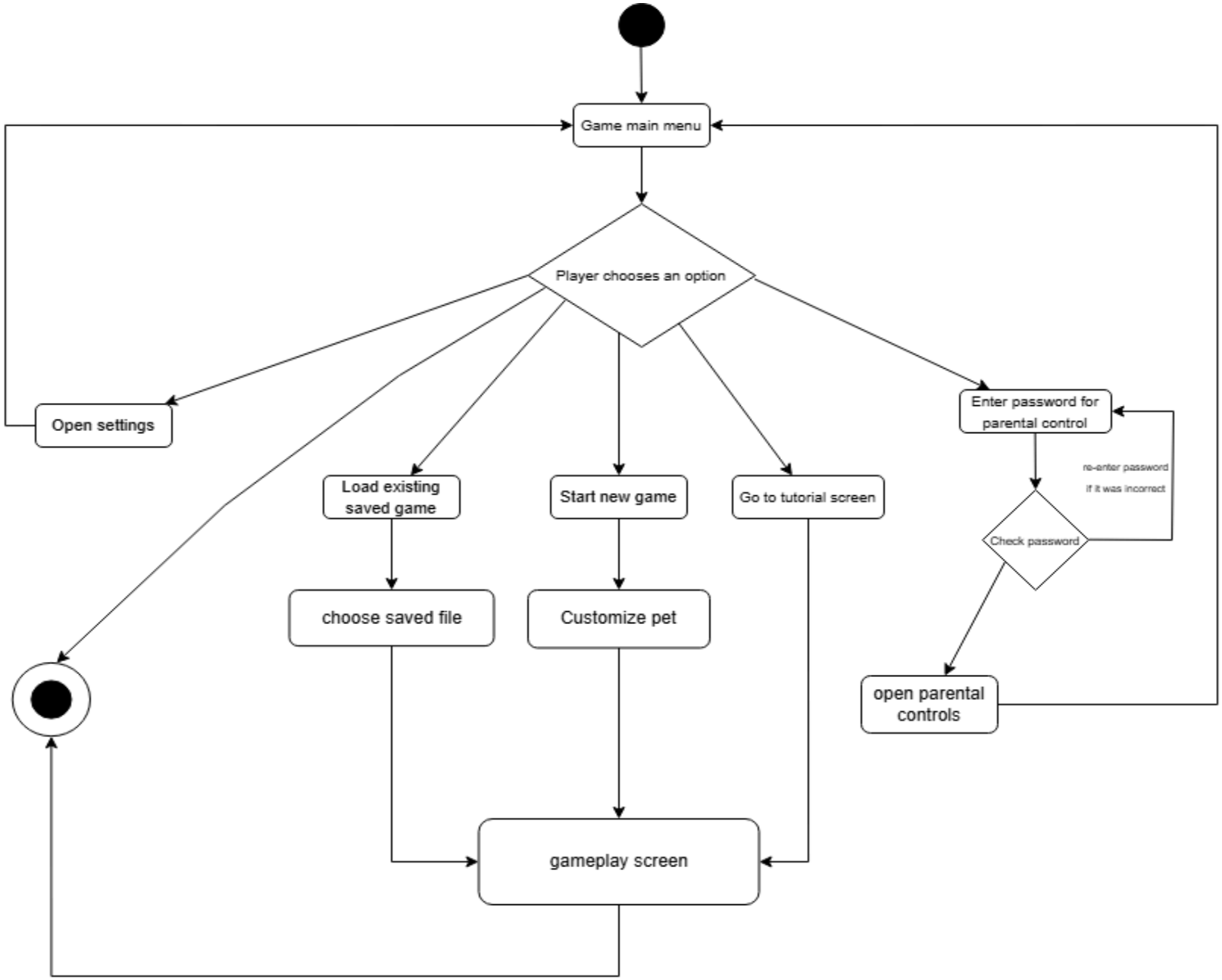
	never gets stuck in a non-interactive state.
Alternatives	If the player force closes the game without using the exit option, the system will try to auto save the info to reduce the risk of losing user data
Exceptions	If an error occurs while saving, such as UI elements not loading properly (or even after minimising), then a backup needs to be checked.
Priority	High

Use Case	
Name	Additional Features - Mini-Games for Pet Rewards
Primary actor	Player
Secondary actors	System / GUI Interface
Goal in context	Allow players to play mini-games with their pet which allows them to earn rewards and increase the pet's happiness level. Different games will offer different rewards and different happiness boosts.
Preconditions	- The player must have an active game session with a pet selected - The mini-game feature has to be accessible from the game's UI - The system must track the happiness of the pet, levels and rewards
Trigger	- The player selects the mini-game option from the game menu or an activity button
Scenario	1. The player selects mini-game menu from the game interface 2. The system presents a list of available mini-games to play 3. The player chooses a mini-game 4. The system launches the mini-game and tracks the player's performance 5. After the completion, the system calculates the earned rewards based on performance 6. The player receives a happiness boost for their pet, higher success in mini-game means

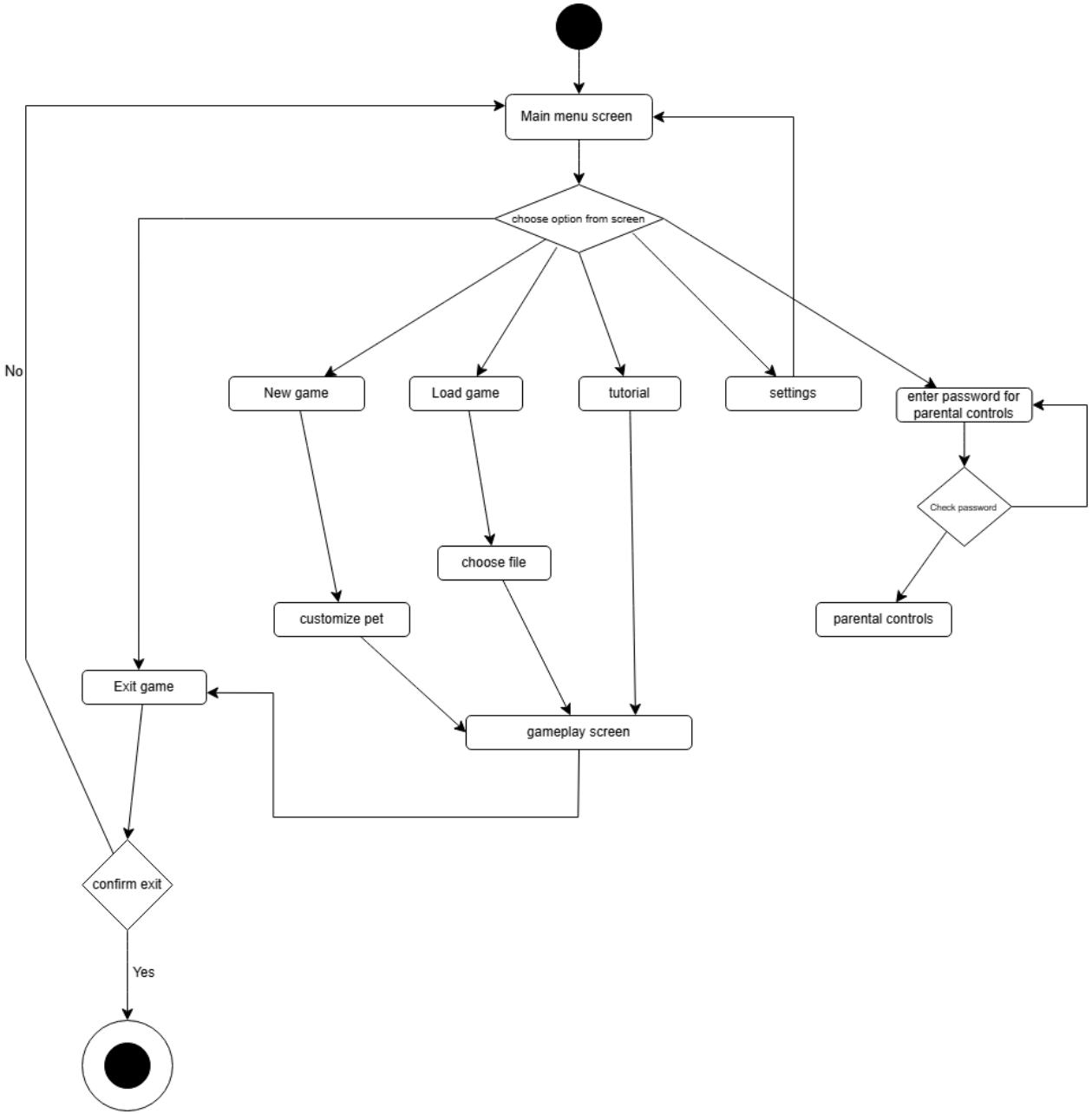
	higher happiness boost 7. A confirmation message is displayed saying “Mini-game complete! You earned (reward) and you pet gained (X) happiness points” 8. The player reuters to the main gameplay screen
Alternatives	- Instead of the instant rewards, mini-games could require a cool down period before they can be played again - The system could introduce special events where mini-games offer bonus rewards - Players could also unlock a new mini-game as they progress and level up in the game
Exceptions	- If the player quits the mini-game, there is not reward or boost of happiness - If a technical issue or error occurs, the system will display an error message saying “The mini-games could not be completed, please try again later” - If the pet is either unhappy or sick, access to mini-games could be restricted until the pet is taken care of
Priority	High - This feature is ideal for engagement of players and replay value, gives a good incentive for players to interact with their pet more often

Use Case - Activity diagrams

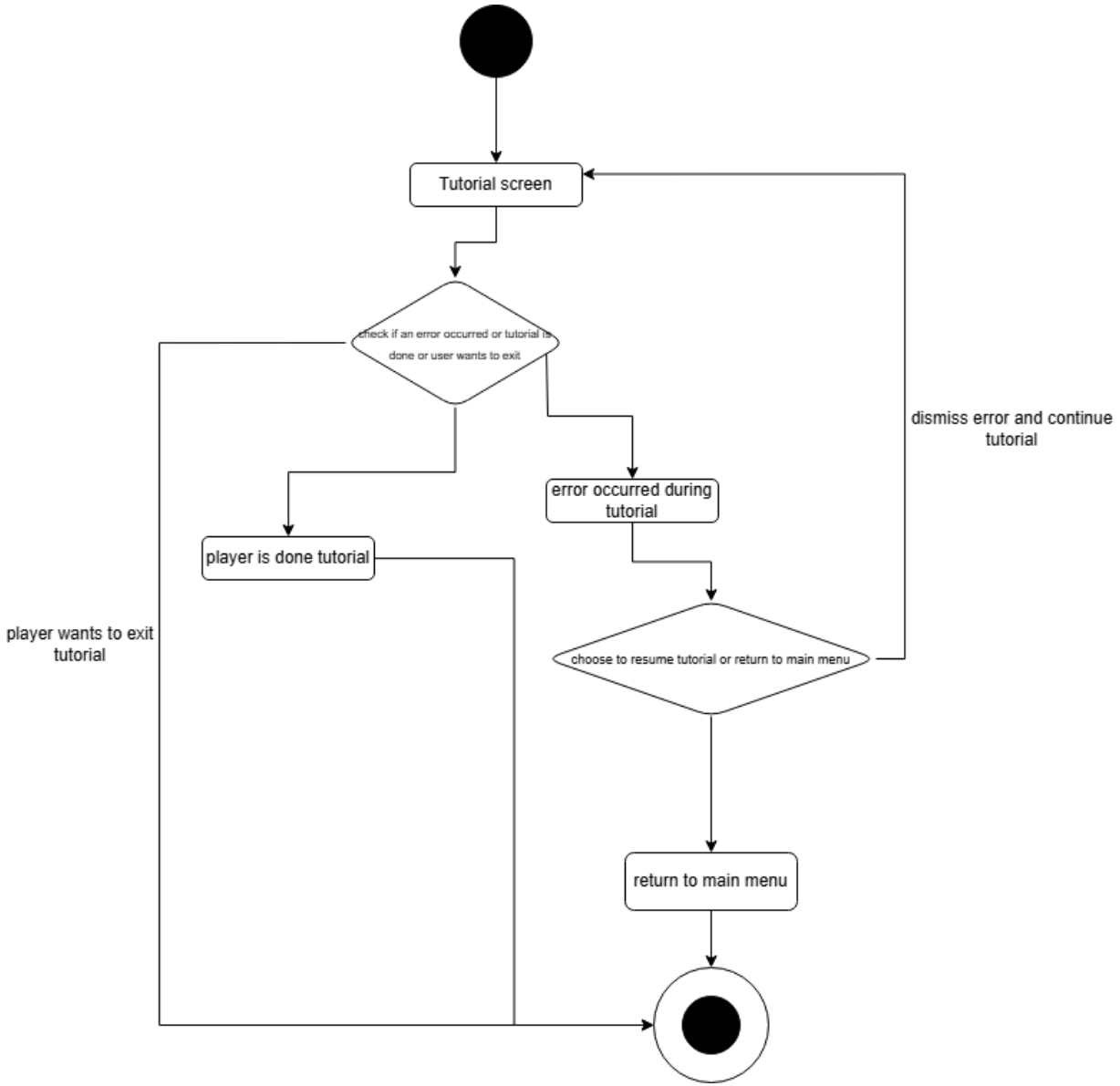
Game interface activity diagram



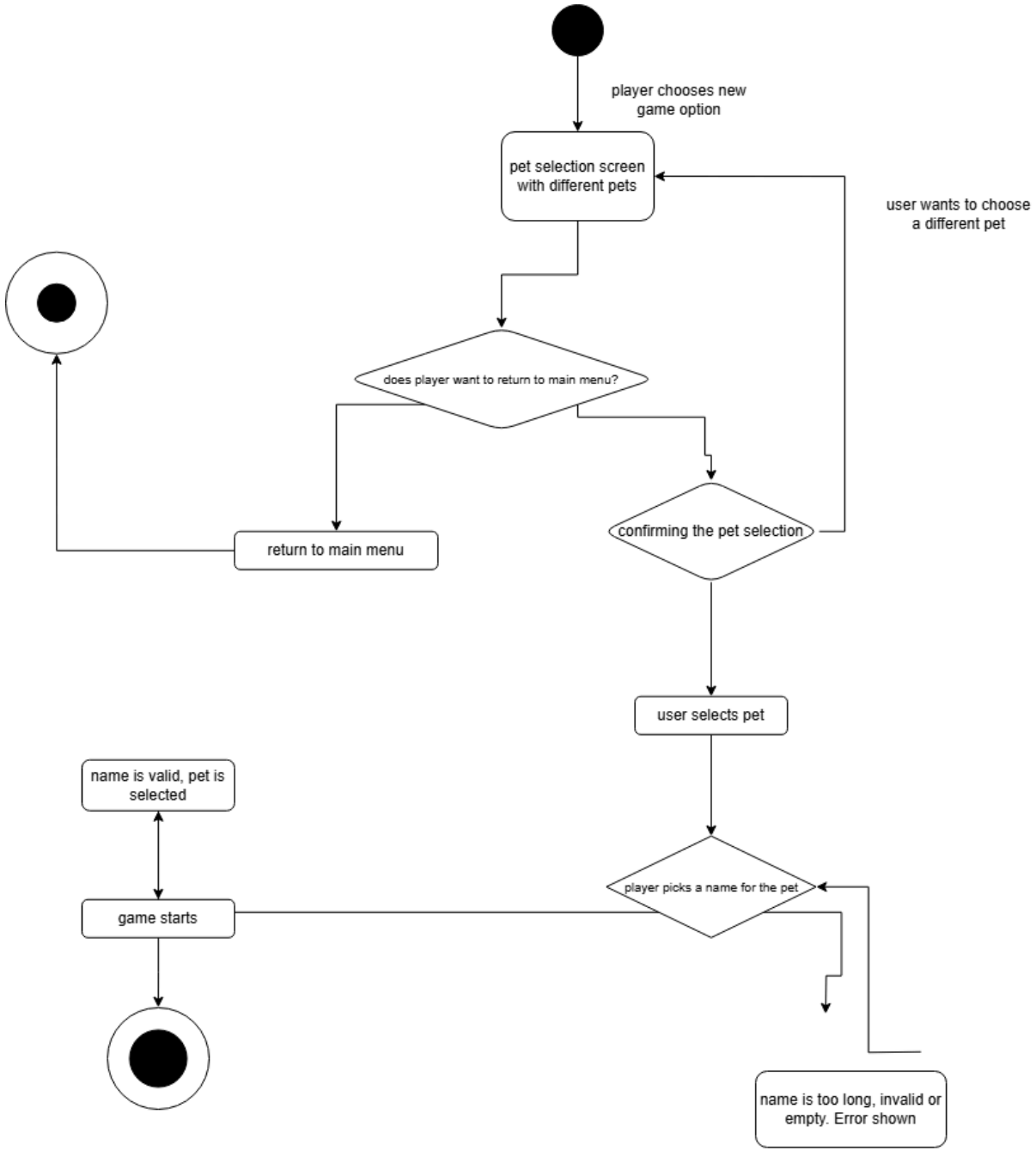
Accessing Main Menu activity diagram



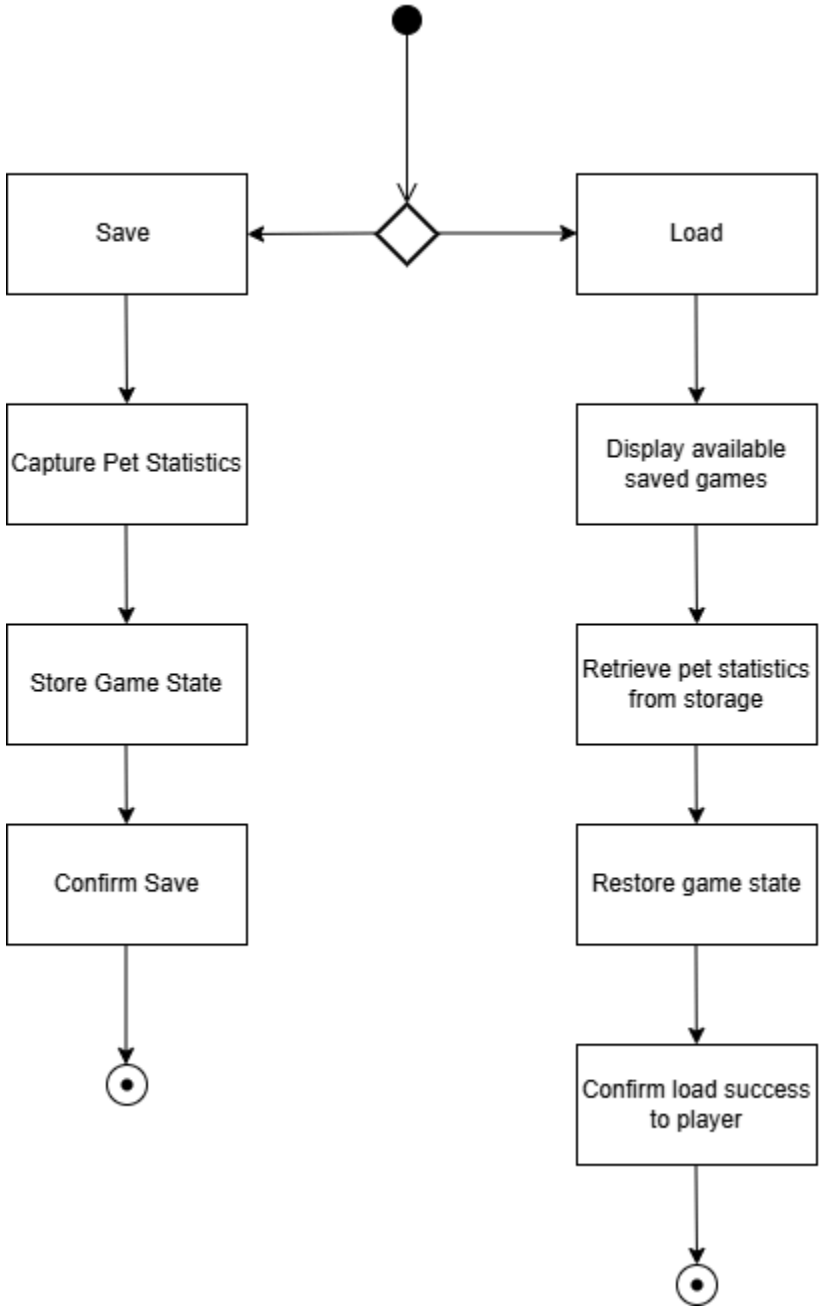
View Instructions/Tutorial Activity Diagram



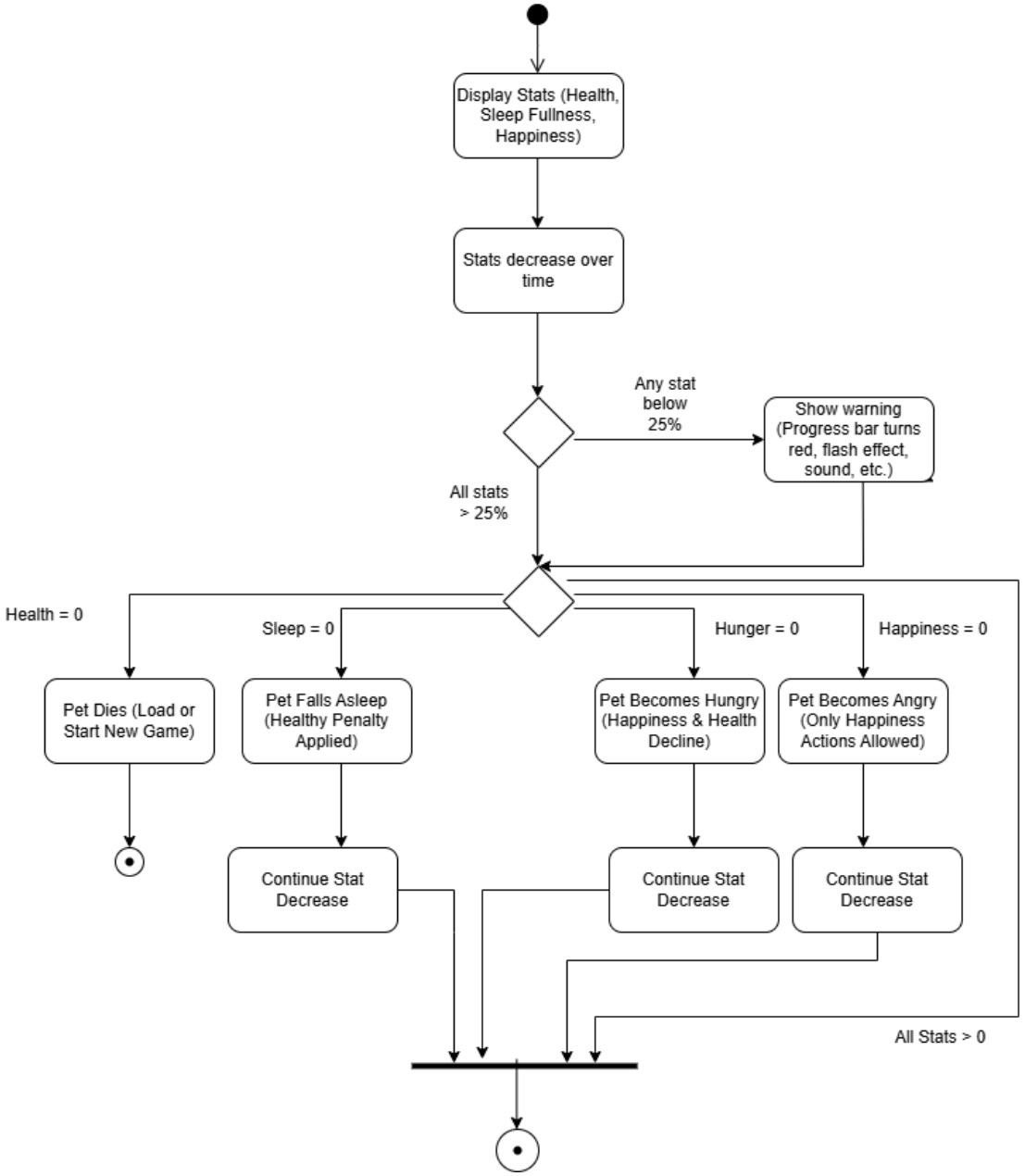
Start new game activity diagram



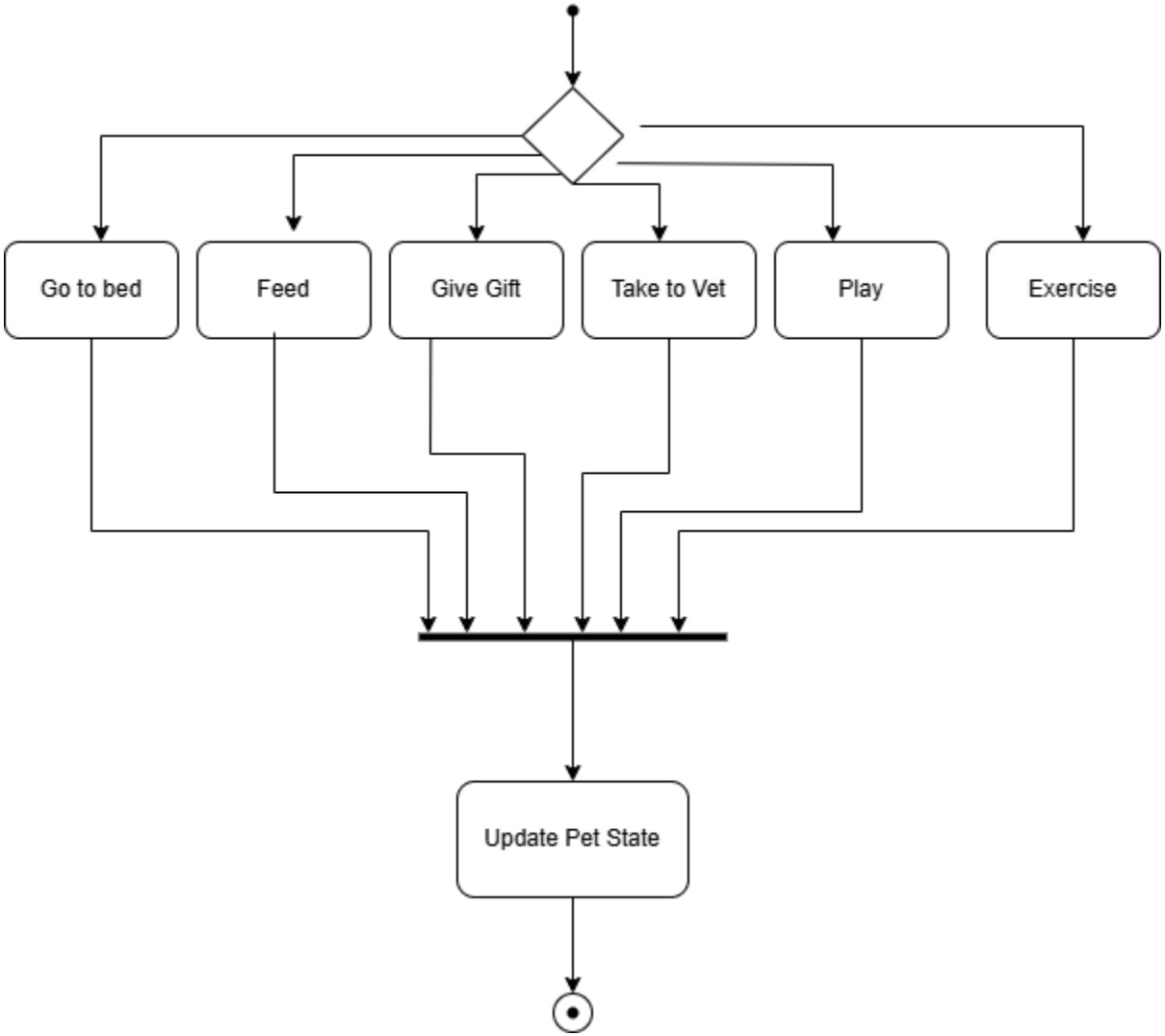
Save/Load Game Activity Diagram



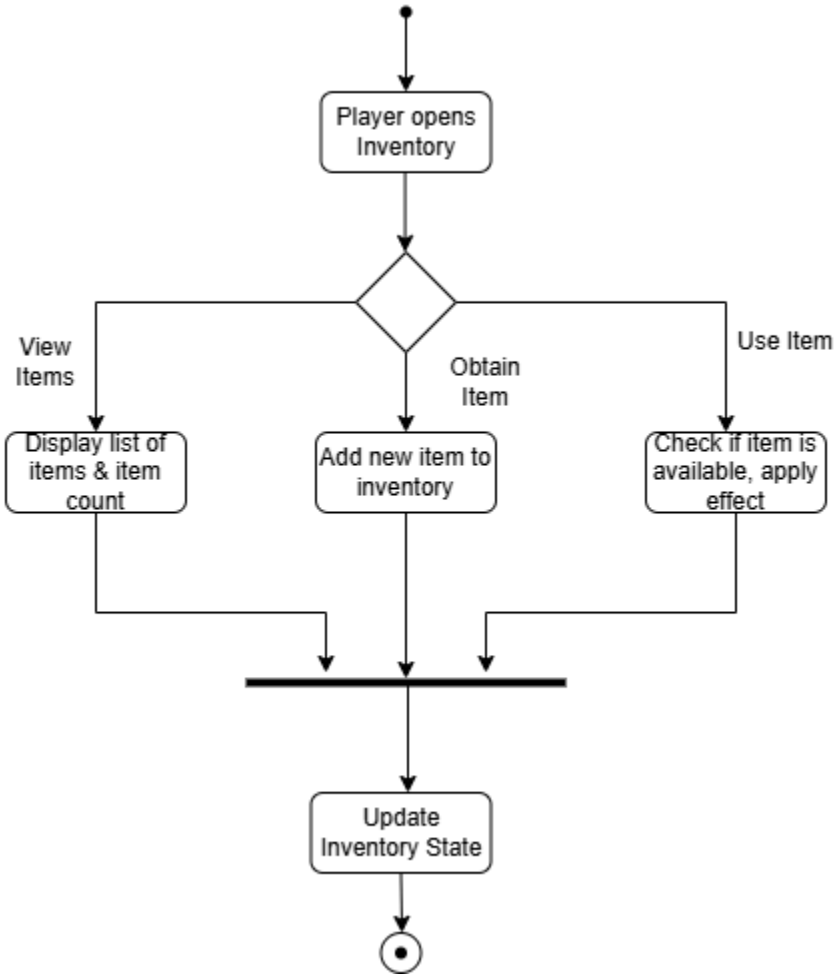
View and Manage Pet Vital Statistics



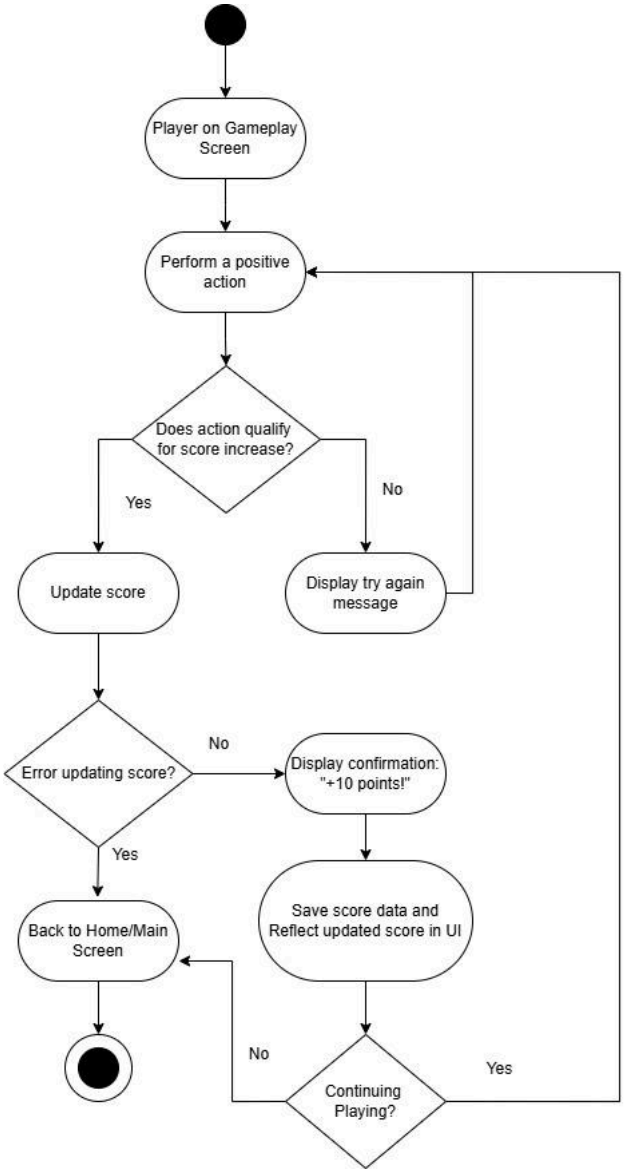
Commands Activity Diagram



Player Inventory Activity Diagram



Keeping score diagram

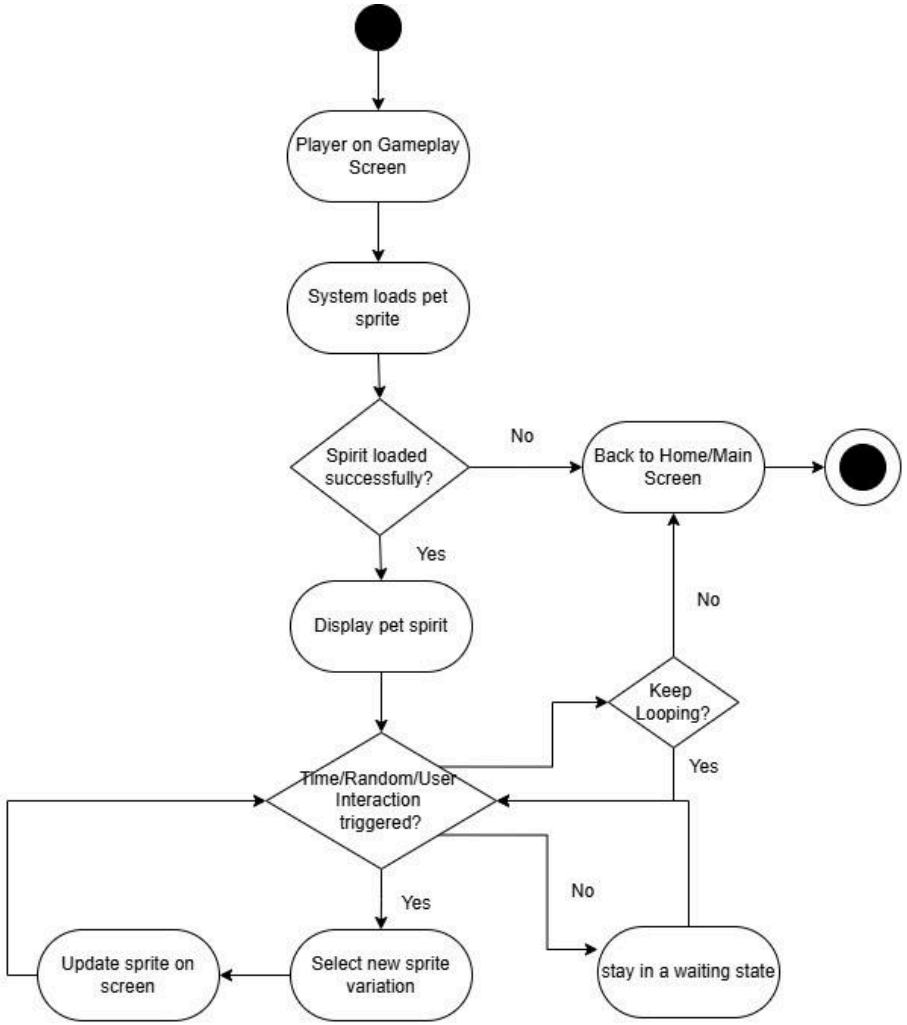


FOR PET SPIRIT USE CASE DIAGRAM

Explanation of looping as it might look confusing on the diagram:

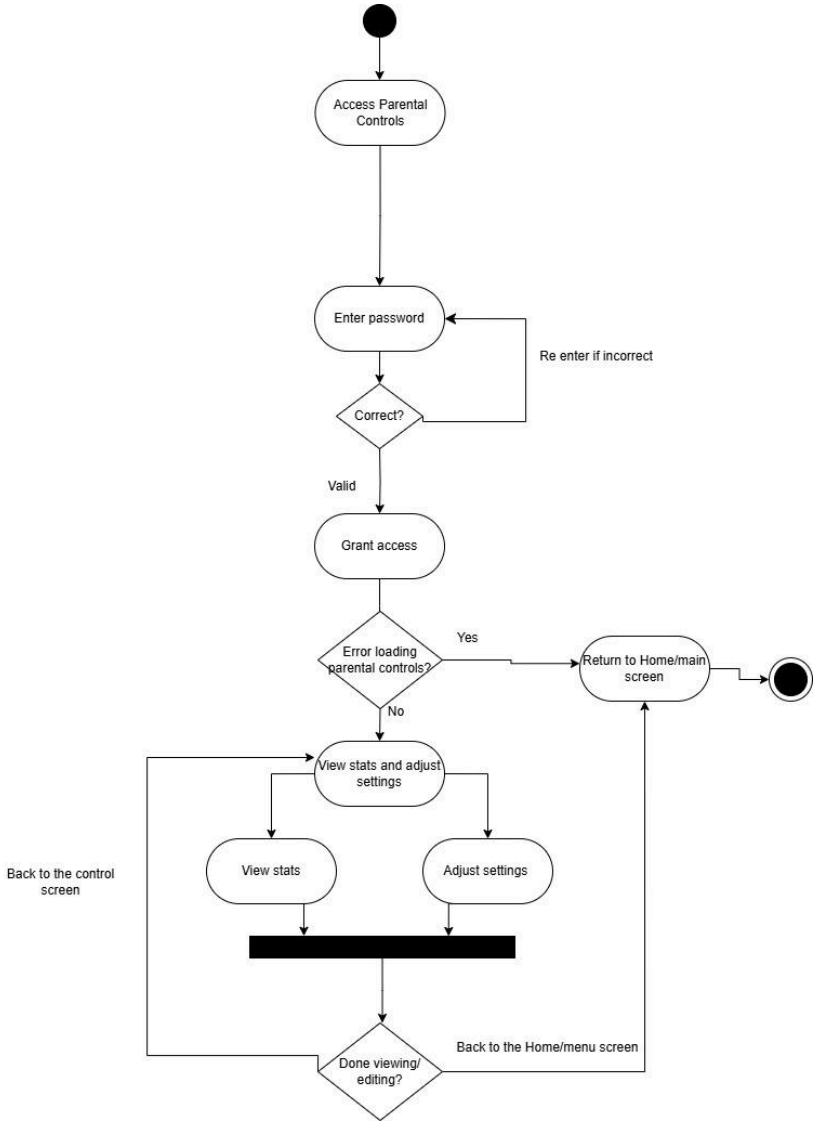
In the case of the pet's sprite, it updates at regular or random intervals and whenever the player interacts with it. We can create a decision node, "Time/Random/User Interaction triggered?" which can determine if a new sprite variation should be displayed, such as blinking or a happy face. If triggered, the system selects a new sprite variation, updates the sprite, and loops back to check for future updates. If not, it continues monitoring until a change occurs. In this case a second decision node, "Keep looping or go to the main menu?" is needed to ensure the player can either stay in the game, allowing the sprite updates to continue, or exit to the main menu at any time.

Pet Sprite Activity Diagram

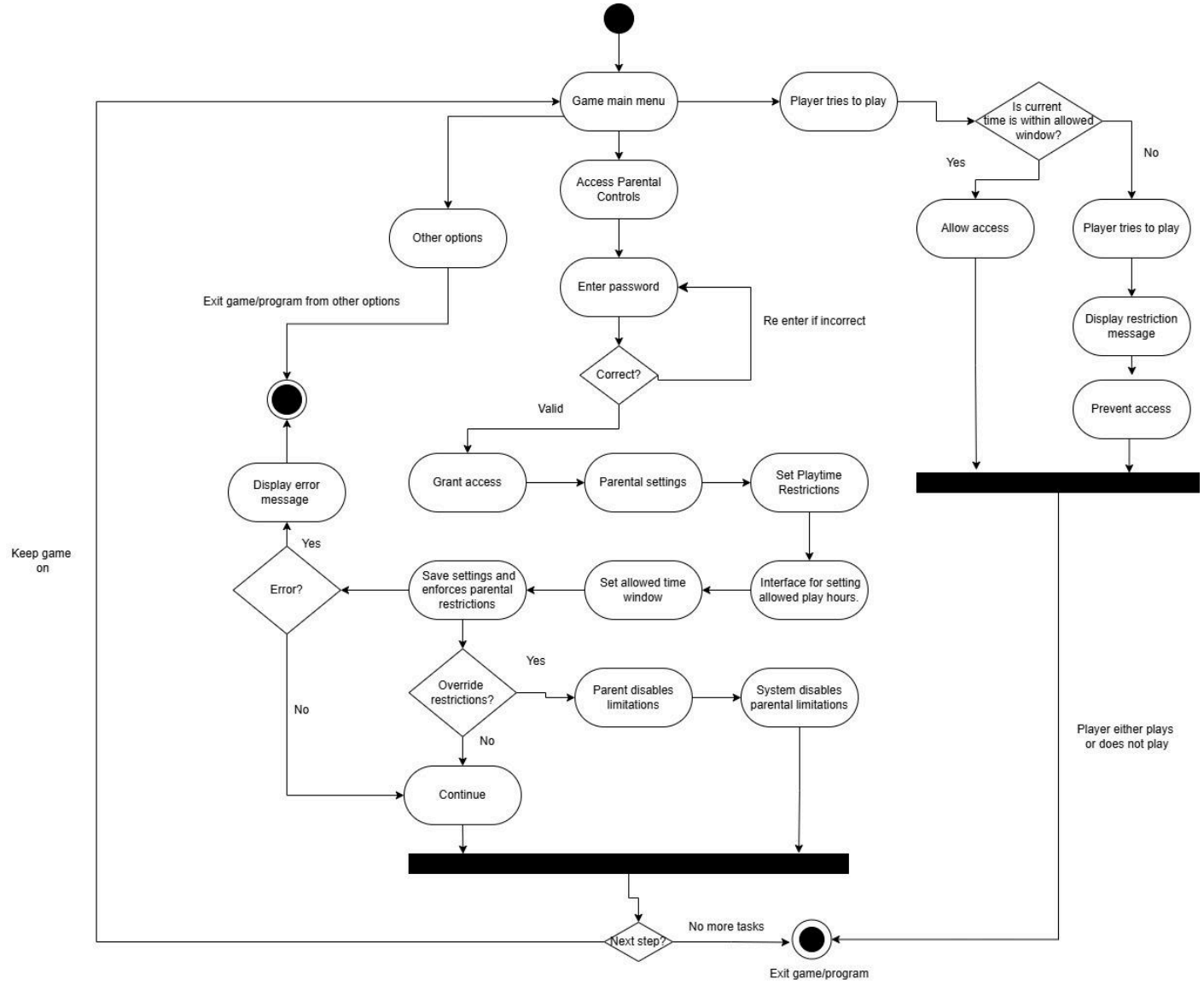


Update spirit.. & stay in the waiting.. Loop back to decision

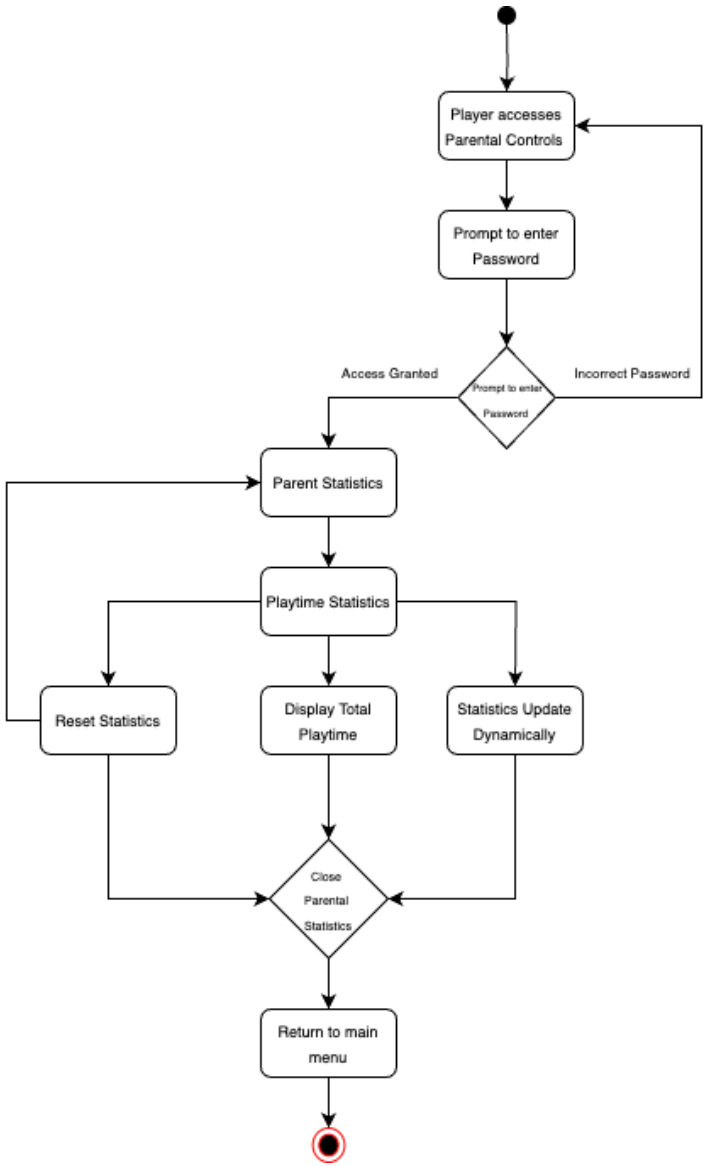
Parental Controls Activity Diagram



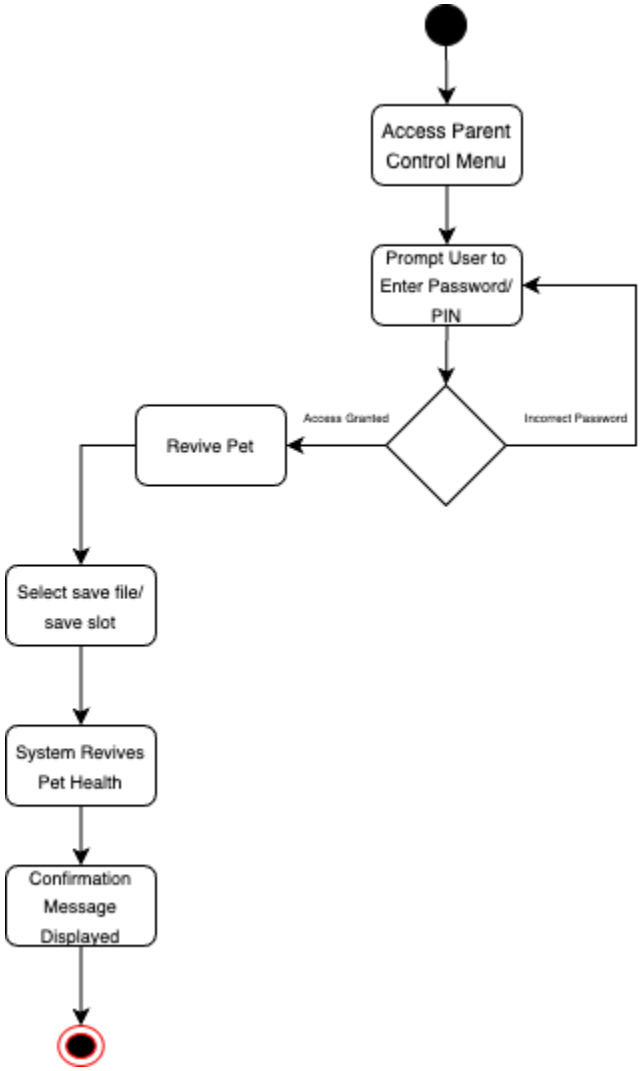
Parental Limitations Activity diagram



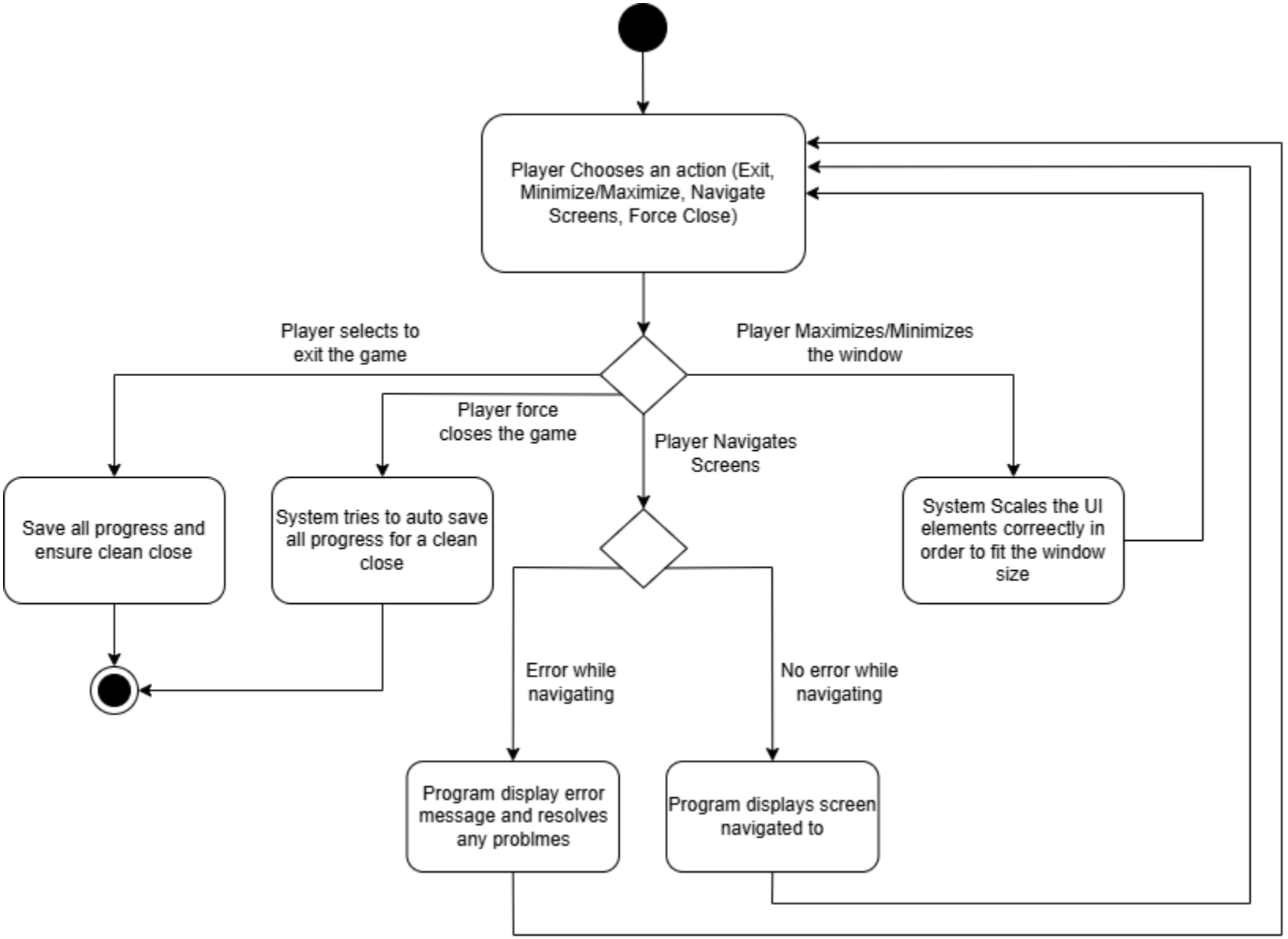
Parental Statistics Activity Diagram



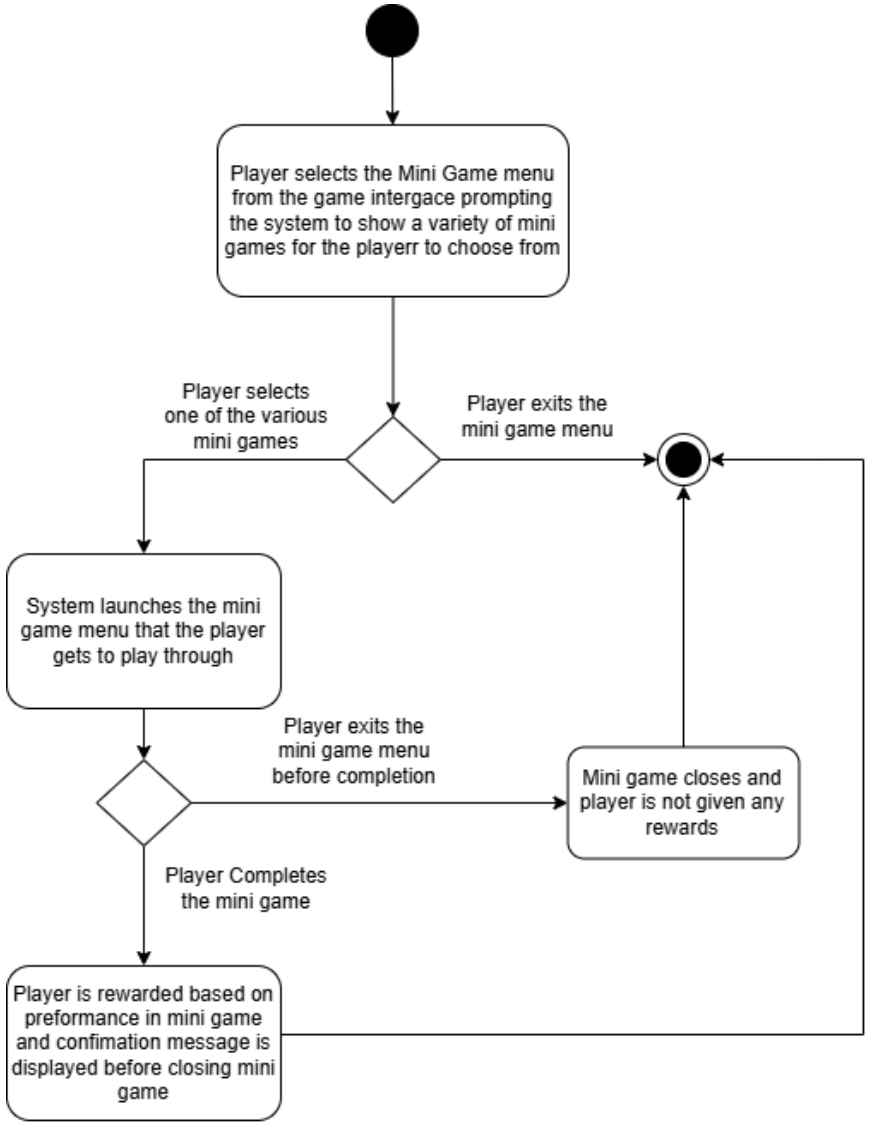
Revive Pet Activity Diagram



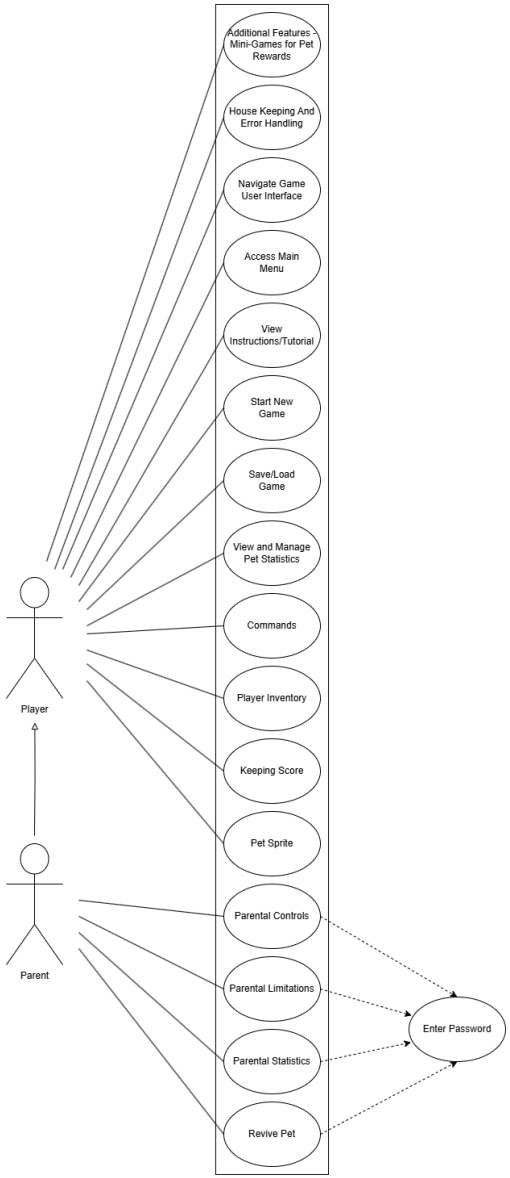
Housekeeping And Error Handling Activity Diagram



Additional Features - Mini-Games for Pet Rewards Activity Diagram



Use Case Diagram (May need to zoom in to read text)



Summary

To summarize, this document outlines the team contract detailing how the project will be managed and the roles of each team member that will be fulfilled. Furthermore, an overview of the project was given explaining what it is and its objectives, the functional and non-functional requirements were stated as well as a domain analysis with its use cases and activity diagrams for developing a virtual pet game. The main objective of the game is to create a game that is similar to the popularized games such as Tamagotchi, Neopets, Nintendogs, and Bitzee which offer an engaging experience that could be playable by all ages. Our game will be created using Java and the player will have to take responsibility for the pet while encouraging the users to have constant engagement with the game. The game will also feature a reward system for completing mini-games and an emotional meter that will increase when a mini-game is completed. The main actors described in the outline of our game would be the player and the parent. The player has access to all use cases that include playing the game while the parent would have access to these as well as additional use cases to manage game usage. The use cases we introduced will all be implemented into our game efficiently allowing for smooth gameplay while having various features that provide the users with a fun interactive game. Overall this document provides a concrete outline of how our virtual pet game will be managed and produced.

Table of Terms, Notations, Acronyms (Citations are found in references section of Introduction table)	
Terms, Notations, Acronyms	**Description**
UML	Unified Modeling Language which is a way to visually depict various aspects of the overall design of a solution.
JSON	Short for JavaScript Object Notation, is a self-describing format of storing and transporting data
JUnit 5	Is a testing framework
GUI	Short for Graphical unit interface, displays a combined system of visual components for s system software
UX	Short for User experience design, is used by design teams to create products that the user can learn and benefit from
XML	Short for eXtensible Markup Language, is a markup language, also similar to jason, for storing

	and transporting data
MB	Abbreviation for megabyte which is a unit of measure
TA	Teaching assistant, helps with marking work, meeting with group, and answering any questions/concerns
GitLab	Collaborative software development software that helps teams organise their projects.