2212 Virtual Pet Project

Requirements Documentation

Team 75: Yonas Asmelash, Julia Norrish, Jessica Wang

Table of Contents

Task Breakdown

Task	**Contributors**	**Description/Notes**
Team Contract	Jessica, Julia, Yonas	The team contract was completed together during a weekly group meeting.
Introduction	Julia	
Domain Analysis	Jessica	
Functional Requirements	Yonas, Jessica	Yonas worked on scenario model and activity diagrams, Jessica assisted with use case diagram
Non-Functional Requirements	Julia	
Summary	Jessica, Julia, Yonas	

2.2 Introduction

Overview
During this project we will develop a virtual pet game in the Java programming language. The game will simulate taking care of a real pet by encouraging players to take care of their virtual pet's needs on a regular basis. Players will learn about responsibility and routine management through an engaging and interactive game.

We will design our virtual pet and its personality to be engaging for the player. Players will have to interact with their virtual pet regularly, engaging in tasks such as feeding, playing, and maintaining the pet's health. We will implement a simple graphical user interface to provide the player with a visual representation of their pet and the aspects associated with caring for it.

This document details the requirements for the virtual pet project, including domain analysis, functional requirements, and non-functional requirements. Throughout this project we will use our knowledge on software engineering principles to create a functional, engaging virtual pet game.

Objectives
The main objective of this project is to apply software engineering principles to develop a virtual pet game. While doing this we will gain experience in:

- Understanding and adhering to project specifications.
- Creating requirements and design models based on the specifications.
- Implementing the design in Java and managing the effect of earlier decisions.
- Developing a graphical user interface.
- Writing efficient, clean, well-documented code following Java best practices.
- Working effectively with a team throughout the project.
- Reflecting on and learning from decisions made while designing the project.

For the game itself, we will aim to develop a virtual pet simulation that is both functional and engaging while being educational. Feeding, playing with, and maintaining the pet's health will be core functions of the game. It will be designed with children between the ages of 7 and 12 in mind and provide a fun experience while promoting responsibility and life skills.

References
Servos, Daniel. "CS2212B Group Project Specification", 2025

2.3 Domain Analysis

Introduction to the domain
This project falls within the Computer Gaming Software domain, which encompasses the development of interactive entertainment applications. The gaming software domain is characterized by real-time user interactions, state management, and graphics. There exists a substantial variety of software of many different formats that fall under this domain, including educational games, action games, simulation games, and casual games, each with their own specific requirements and design patterns. Computer Gaming Software can be a mobile application, desktop application, website, and more.

Target Audience
Based on the project specifications (Section 3.2.3), the primary target audience is children aged 7 and up. The target audience has been further refined to children between the ages of 7-12, particularly girls.

Specific Subdomain
This project falls under the subdomain of Virtual Pet Care Simulation Games, which is a popular type of game itself. This domain became popular in the mid-1990s with the introduction of Tamagotchi devices, and now represents an intersection between entertainment and educational gaming.

Domain Details
The virtual pet care simulation domain is characterized by several key elements, including:

- Cyclical gameplay loops centered around meeting the pet's needs
- Time-based mechanics, where certain interactions are necessitated after a certain amount of time passes
- Emotional investment through personalization (ie. names, colours, accessories)
- Simple but engaging interactions and gameplay, to facilitate pet care (ie. feeding the pet, cleaning the pet, playing games with the pet)
- Non-violent, family friendly gameplay

Common Issues in this Domain
Player retention: Once the novelty wears off, many players become bored of the game
Limited interaction: Games often become repetitive due to a limited number of actions and interactions that the player can make. After the user has completed all actions (ie. feeding the pet, cleaning it) they must wait until more time passes.

Common Solutions to Above Issues
Player Retention: Implementing rewards systems and achievements, or progressive unlocking of new features and pet types encourages the user to return. Implementing social or competitive elements, where players can compete with others on the software, also increases player retention.

Limited interaction: Including mini-games associated with each pet care task to vary gameplay (ie. must play a minigame to feed the pet). Special or randomized events (known as gacha) can also make the game more interesting.

Domain Implication on Project
Having a thorough understanding of the virtual pet care simulation subdomain will help us:

- Leverage established gaming industry practices and patterns
- Use proven UI/UX patterns that games are familiar with (menus, settings, save systems)
- Follow standard gaming interface conventions (keyboard shortcuts, mouse interactions)
- Apply existing successful virtual pet mechanics such as health bars, different levels (happiness, hungriness, cleanliness, etc)
- Add engagement features that exist in current games (achievements, sound effects and music, pet customization)
- Use established game development libraries and frameworks in Java

2.4 Functional Requirements

Functionality to be delivered
User Interface System

- Multiple required screens (main menu, gameplay, tutorial, load/new game, parental controls)
- Mouse-based interaction support
- Keyboard shortcuts for common actions
- Visual/auditory feedback system

Main Menu System

- Display options: new game, load game, tutorial, parental controls, exit
- Show game visuals and team information
- Navigation between screens

Pet Selection System

- Minimum 3 different pet types with unique characteristics
- Name input functionality
- Pet type information display
- Save file creation upon selection

Save/Load System

- Save current game state (pet stats, inventory, etc.)
- Multiple save slot support (minimum 3)
- Load previous game states
- Save confirmation messaging

Statistics Management System

- Track and display health, sleep, fullness, happiness
- State management (dead, sleeping, hungry, angry)

- Visual indicators for low stats
- State-specific behavior implementation

Command System
- Support for basic commands (feed, sleep, play, etc.)
- State-dependent command availability
- Command cooldown management
- Visual feedback for commands

Inventory System
- Track food and gift items
- Display current inventory counts
- Item acquisition mechanics
- Item usage tracking

Score System
- Track player score
- Score modification based on actions
- Score display
- Score persistence

Pet Visualization System
- State-dependent sprite display
- Basic animation support
- Different sprites for each pet type
- State indication through visuals

Parental Control System
- Password protection
- Time restriction settings
- Play time statistics
- Pet revival functionality

Error Handling System
- User input validation
- Error message display
- Graceful error recovery
- Data integrity protection

Extra Functional Requirement (may not have to do since team has been reduced to 3 members):
- Include sound effects and visual feedback
    - Actions that take place during the game will have sound effects
    - Buttons that the user interacts with via the GUI will have sound effects
    - Actions that take place during the game will have animations/movements of sorts
- Different minigames for different pet types
    - User will interact with different minigames depending on their pet type
        - The more pet types they own the more minigames they can play

Scenario model

Actors

Player

Description	Primary user who interacts with the virtual pet, responsible for pet care and maintenance
Aliases	User, Game Player
Inherits	N/A
Actor Type	Person
Active/Passive	Active

Parent

Description	Administrative user with access to additional controls and statistics
Aliases	Administrator, Guardian
Inherits	Player
Actor Type	Person
Active/Passive	Active

Timer System

Description	External system that manages time-based events and stat decay
Aliases	Clock, Time Manager
Inherits	None
Actor Type	External System
Active/Passive	Active

Use Cases

Name	Main Menu Navigation
Primary Actor	Player
Secondary Actors	N/A
Goal in context	Allow player to access different sections of the game through a central menu interface
Preconditions	Game is launched and running with main menu presented to user
Trigger	Player starts the game or returns to main menu
Scenario	1. The system displays main menu screen with title and options 2. System shows game graphics and developer information 3. Player selects one of the available options (new game, load game, tutorial, parental controls, or exit) 4. System validates selection 5. System transitions to selected screen
Alternatives	● N/A
Exceptions	● System fails to load menu graphics ● System fails to transition to selected screen
Priority	Highest

Name	Pet Selection
Primary Actor	Player
Secondary Actors	N/A
Goal in context	Allow player to choose and name their virtual pet from available options
Preconditions	Player has selected "New Game" from main menu
Trigger	Player enters pet selection screen

Scenario	1. 1. System displays available pet types (minimum 3) 2. System shows characteristics of each pet type 3. Player selects desired pet type 4. System prompts for pet name 5. Player enters pet name 6. System creates new save file 7. System transitions to main game screen
Alternatives	● Player cancels selection and returns to main menu
Exceptions	● Invalid pet name entered ● Save file creation fails
Priority	High (less so that starting the game)

Name	Save Game Progress
Primary Actor	Player
Secondary Actors	N/A
Goal in context	Save current game state including pet statistics, inventory, and progress
Preconditions	Game is in progress and player has active pet
Trigger	Player selects save option or automatic save trigger
Scenario	1. System prepares current game state data 2. System validates save file location 3. System writes game state to file 4. System confirms successful save to player 5. Game continues
Alternatives	● Automatic save at checkpoints ● Save and quit to main menu
Exceptions	● Save file write fails ● Insufficient disk space ● File permission issues

Priority	High

Name	Manage Pet Statistics
Primary Actor	System
Secondary Actors	Player
Goal in context	Track and update pet's vital statistics (health, sleep, fullness, happiness)
Preconditions	Game is running with active pet
Trigger	Continuous monitoring during gameplay
Scenario	1. System continuously monitors vital statistics 2. System updates statistics based on time and actions 3. System checks for critical thresholds 4. System displays warnings when stats are low 5. System applies state changes based on statistics 6. System updates visual representation of pet
Alternatives	● N/A
Exceptions	● Calculation errors ● Display update failures
Priority	Highest

Name	Execute Pet Commands
Primary Actor	Player
Secondary Actors	N/A
Goal in context	Allow player to interact with pet through various commands
Preconditions	Pet is in a state that allows commands

Trigger	Player selects a command
Scenario	1. 1. System displays available commands based on pet state 2. Player selects command 3. System validates command availability 4. System checks required resources (inventory items, cooldowns) 5. System executes command 6. System updates pet statistics 7. System provides feedback animation/sound
Alternatives	● Command unavailable due to pet state
Exceptions	● Invalid command for current state ● Insufficient resources ● Command on cooldown
Priority	High

Name	Inventory Management
Primary Actor	Player
Secondary Actors	N/A
Goal in context	Track and manage player's inventory of food and gift items
Preconditions	Game is active
Trigger	Player views inventory or uses item
Scenario	1. System displays current inventory items 2. System shows item counts 3. Player selects item to use 4. System validates item availability 5. System applies item effects 6. System updates inventory counts
Alternatives	● Obtain new items through game mechanics

	● View only mode
Exceptions	● Invalid item selection ● Item count tracking errors
Priority	High

Name	Score Management
Primary Actor	System
Secondary Actors	Player
Goal in context	Track and update player's score based on actions and events
Preconditions	Active game session
Trigger	Player performs scoreable action
Scenario	1. System monitors for score-affecting actions 2. System calculates score changes 3. System updates total score 4. System displays updated score 5. System saves score with game state
Alternatives	● N/A
Exceptions	● Score calculation errors ● Display update failures
Priority	Medium

Name	Parental Control Management
Primary Actor	Parent
Secondary Actors	N/A
Goal in context	Configure and manage parental control settings
Preconditions	Valid parent password entered

Trigger	Access parental control menu
Scenario	1. System requests password 2. Parent enters password 3. System validates password 4. System displays parental control options 5. Parent configures time restrictions 6. Parent views play statistics 7. Parent manages pet revival options 8. System saves settings
Alternatives	● Invalid password entry
Exceptions	● Password validation fails ● Settings save fails
Priority	Medium

Name	Pet Visualization System
Primary Actor	System
Secondary Actors	N/A
Goal in context	Display appropriate pet sprites and animations based on current state
Preconditions	Pet exists and game is running
Trigger	Pet state changes or animation timer triggers
Scenario	1. System monitors pet state changes 2. System selects appropriate sprite set for current state 3. System manages basic animation timer 4. System updates sprite display 5. System adds state indication icons if needed 6. System ensures smooth transition between states
Alternatives	● Different sprite sets for different pet types
Exceptions	● Missing sprite assets

	● Animation timing errors ● Display update failures
Priority	High

Name	Error Handling System
Primary Actor	System
Secondary Actors	Player
Goal in context	Handle errors and provide clear feedback to users
Preconditions	System is running
Trigger	Error condition occurs
Scenario	1. System detects error condition 2. System logs error details 3. System determines error severity 4. System creates user-friendly error message 5. System displays error to user 6. System attempts recovery if possible 7. System maintains data integrity
Alternatives	● Automatic recovery for minor errors
Exceptions	● Critical system failures ● Unrecoverable data corruption
Priority	High

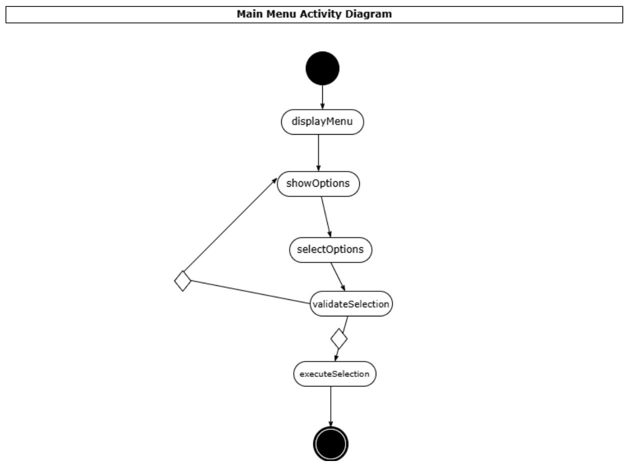
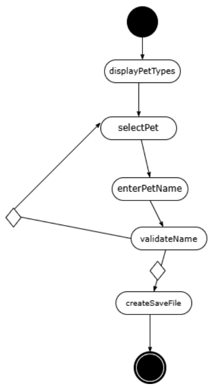
Name	UI System
Primary Actor	System
Secondary Actors	Player
Goal in context	Provide accessible and responsive user interface across all screens
Preconditions	Game is running

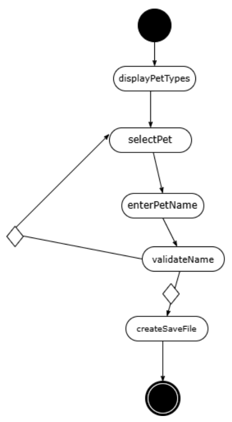
Trigger	Game being started and any UI interaction
Scenario	1. System initializes UI components 2. System monitors for mouse input 3. System monitors for keyboard shortcuts 4. System processes input 5. System provides feedback 6. System updates UI state 7. System manages screen transitions
Alternatives	● Keyboard-only navigation
Exceptions	● UI component initialization failures ● Input processing errors
Priority	Highest

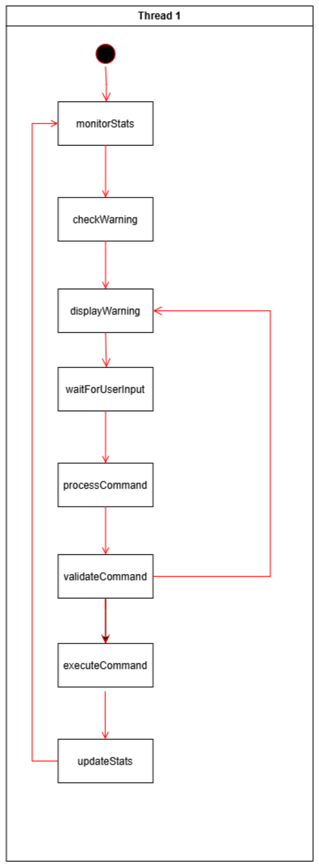
Name	Sound System
Primary Actor	System
Secondary Actors	Player
Goal in context	Provide enhanced gameplay through sound effects and pet-specific minigames
Preconditions	Game is running and pet is active
Trigger	Player initiates an action requiring sound
Scenario	1. System monitors for sound trigger events 2. System plays appropriate sound effects 3. System checks pet type for available minigames 4. System loads appropriate minigame 5. System manages minigame state 6. System tracks minigame progress 7. System provides rewards based on performance
Alternatives	● Different minigames for different pet types

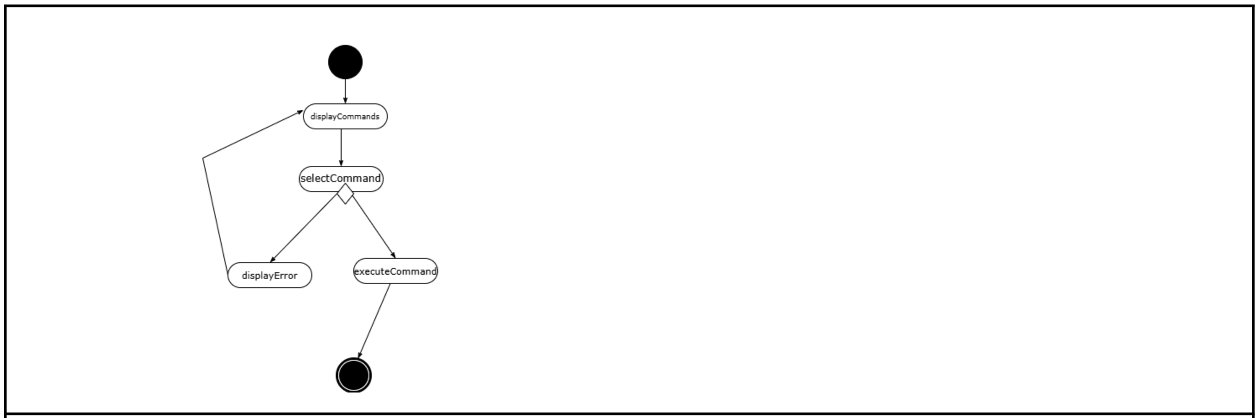
Exceptions	• Sound playback failures • Minigame loading errors
Priority	Medium

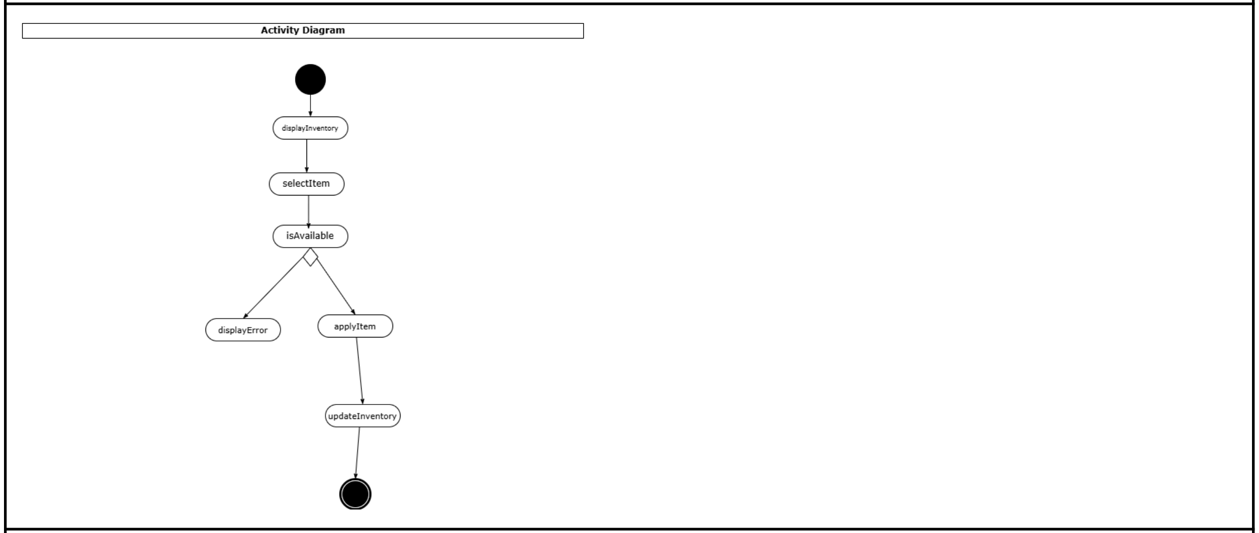
Activity Diagrams

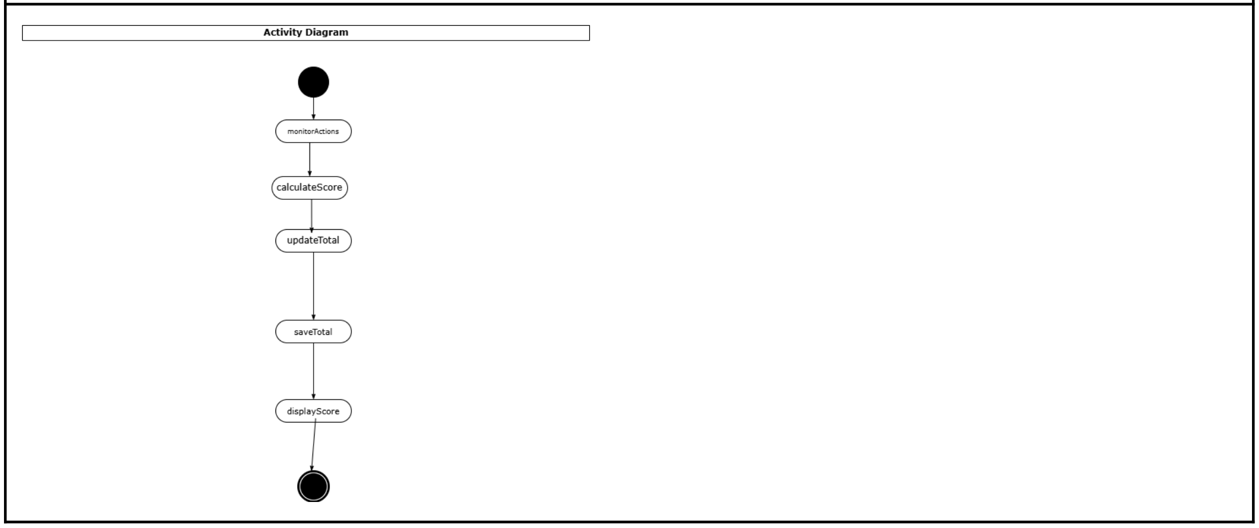
 Main Menu Activity Diagram displayMenu showOptions selectOptions validateSelection executeSelection
 displayPetTypes selectPet enterPetName validateName createSaveFile

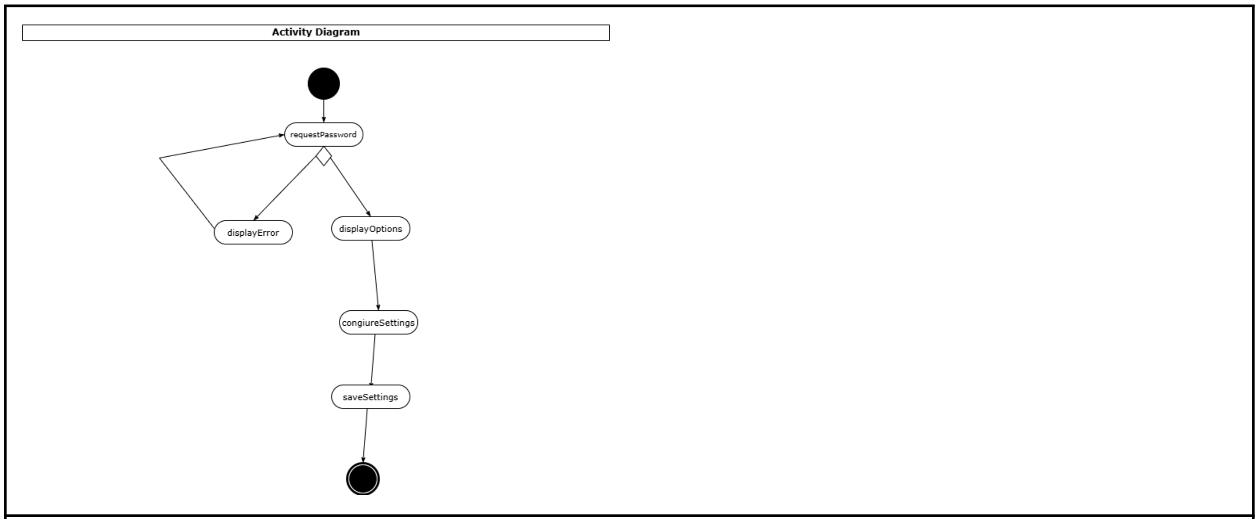

displayPetTypes
selectPet
enterPetName
validateName
createSaveFile

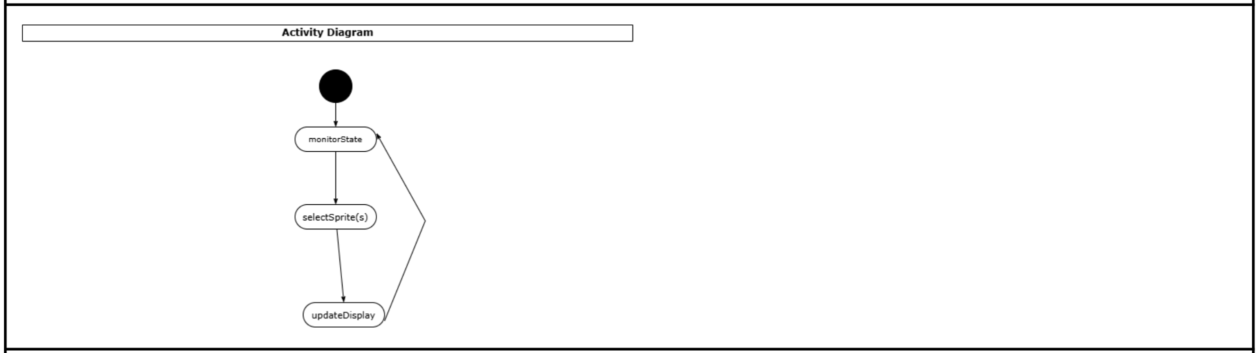

Thread 1
monitorStats
checkWarning
displayWarning
waitForUserInput
processCommand
validateCommand
executeCommand
updateStats

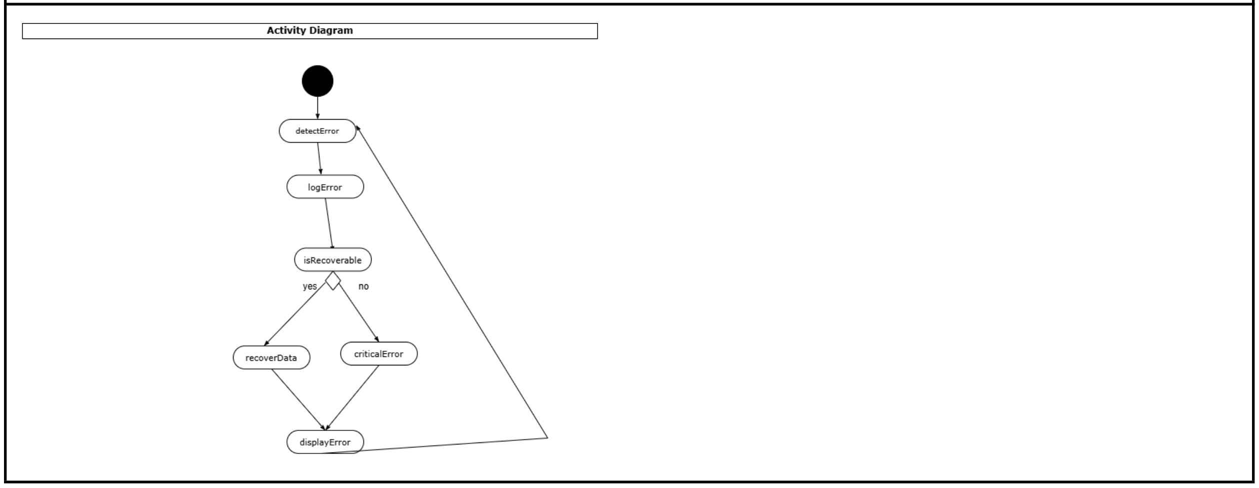

displayCommands
selectCommand
displayError
executeCommand

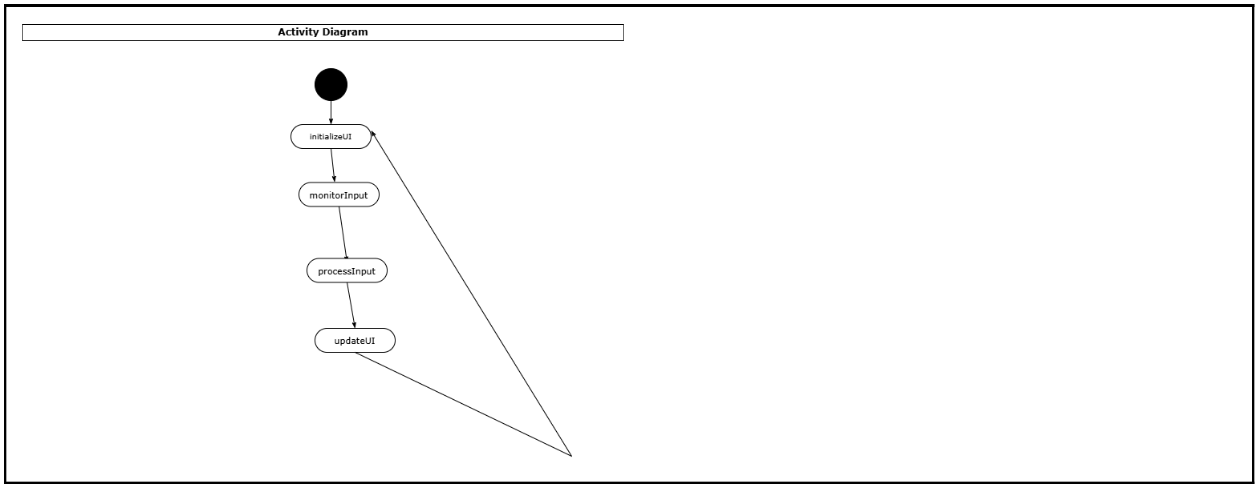

Activity Diagram
displayInventory
selectItem
isAvailable
displayError
applyItem
updateInventory

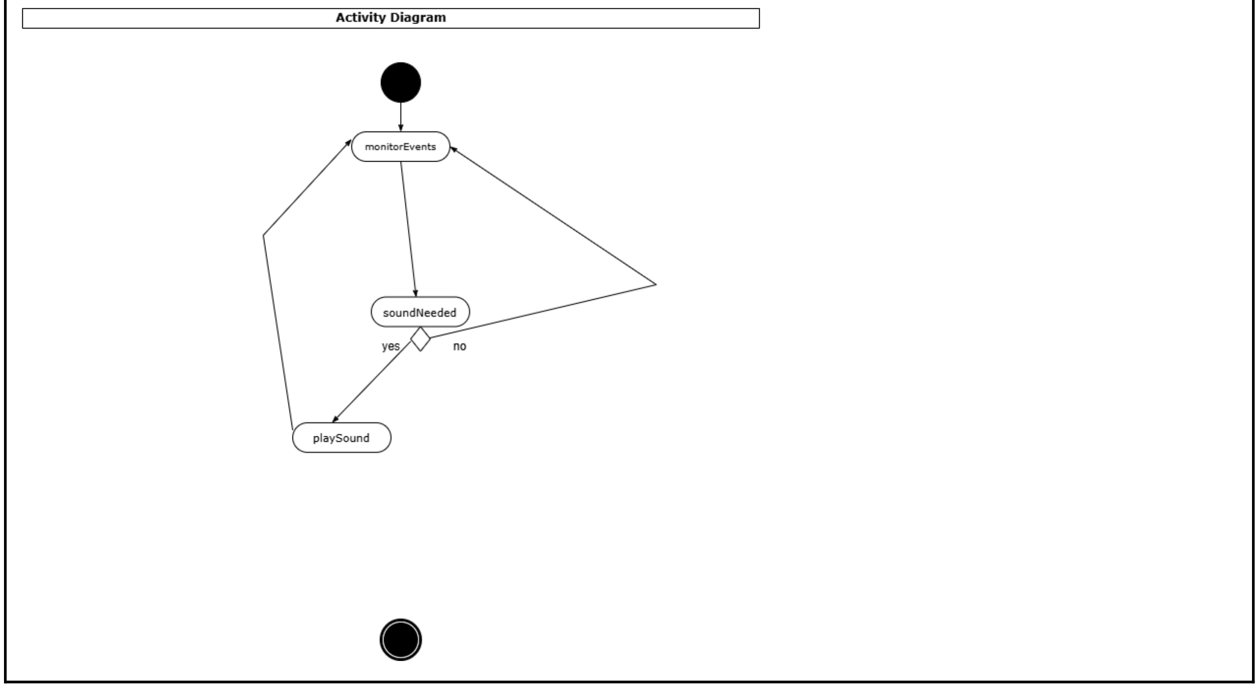

Activity Diagram
monitorActions
calculateScore
updateTotal
saveTotal
displayScore


Activity Diagram
requestPassword
displayError
displayOptions
congiureSettings
saveSettings

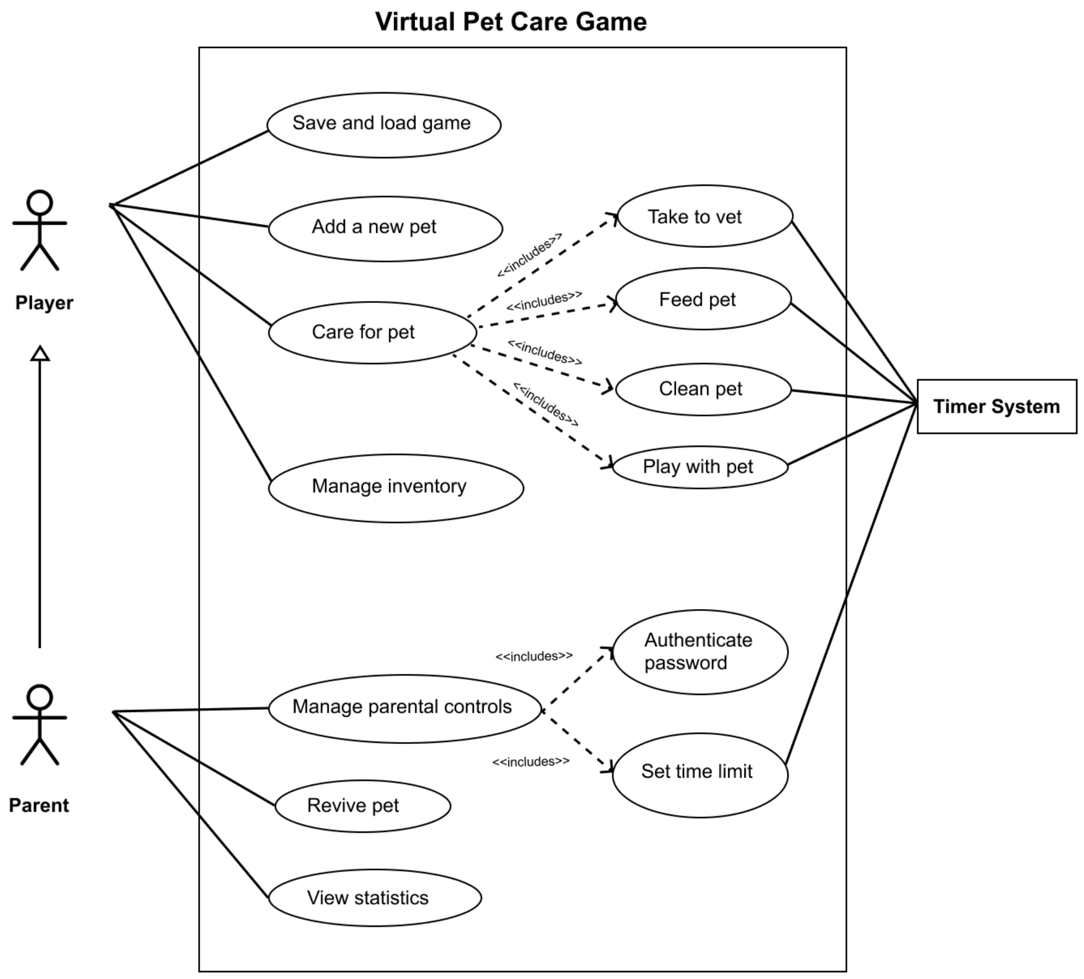

Activity Diagram
monitorState
selectSprite(s)
updateDisplay


Activity Diagram
detectError
logError
isRecoverable
yes
no
recoverData
criticalError
displayError


Activity Diagram
initializeUI
monitorInput
processInput
updateUI


Activity Diagram
monitorEvents
soundNeeded
yes
no
playSound

Use Case Diagram



2.5 Non-Functional Requirements

1. The application will be developed in Java 23 and will function properly in Java 23 or newer.
2. The application will apply object-oriented concepts.
3. The game will be suitable for children aged 7 and older and not include any offensive material.
4. A well-designed Graphical User Interface with a focus on user experience will be created.
5. A JSON file will be used to store the game data locally, without an internet connection.
6. Any libraries or tools used in the application will be included in the application.
7. GitLab will be used to store all the files for the application, and it will be used regularly throughout the project.
8. The Wiki on GitLab will be used to store the designs and diagrams for the project.
9. GitLab will be used regularly to keep track of the tasks and issues of the project.
10. Each file and function will be documented using Javadoc comments. Any code acquired from outside sources will be properly cited.
11. Junit 5 will be used to test the code that does not require the GUI to be tested.
12. Our team will be consistent in our coding styles throughout the project.
13. All team members will use VSCode or Eclipse to develop the game. The project will run on Windows 11 with a standard Java installation.
14. The program will not alter any files that were not made by the program itself.
15. There will be a reasonable response for every action taken by users interacting with the application. Any necessary error messages will be displayed professionally.
16. The file size of the project will not exceed 500 megabytes.
17. The application will be developed efficiently to prevent it from being non-responsive.
18. Accessibility will be considered during development, specifically regarding colours and how the user interacts with the game.
19. The application will be developed with a focus on code reusability to ensure the application will be easily maintainable.
20. All work during the project including the game’s content, documentation, designs, and communication between team members will be in English.
21. The software engineering techniques discussed in the course will be used throughout the course to create a high-quality product.

2.6 Summary

This requirements documentation outlines the specifications for our virtual pet care simulation game, targeted at children aged 7-12. The document covers the project's analysis of its domain (Computer Gaming Software, which can be further specified to the Virtual Pet Care Simulation Game subdomain), detailed functional requirements for core features like pet care mechanics and parental controls, and non-functional requirements that ensure code quality and user experience. This documentation outlines technical requirements in three ways: the scenario model defines primary actors and use cases, the activity diagrams identify two main threads of application activity, and the use case diagrams identify three users (player, parent, and the game's timing system), demonstrating how they will interact with the system. Our additional features of sound effects and background music will enhance gameplay engagement. These requirements provide the foundation for developing an educational and entertaining virtual pet simulation that meets both technical specifications and user needs.

Terms, notations, and acronym

Term	**Definition**
IDE	Integrated Development Environment (the application used to write the code for our software)
GitLab	The version control system (a system that lets us save work and collaborate) that we will use for this project
GUI	Graphical User Interface (the part of the software application that the user sees and directly interacts with)
UI	User Interface - the visual components and interactive elements through which users interact with the software (buttons, menus, displays, etc.)
UX	User Experience - the overall effectiveness, efficiency, and satisfaction with which users can accomplish their intended tasks when interacting with software