Software Requirements Specification

for

PetQuest

Version <0.1>

Prepared by

Group Name: GROUP C2

Lee Shing Hei	1155183712	lshcyrus@link.cuhk.edu.hk
Sin Cheuk Yin	1155193665	1155193665@link.cuhk.edu.hk
Tong King Laam	1155194266	1155194266@link.cuhk.edu.hk
Wong Po Hing	1155183715	1155183715@link.cuhk.edu.hk
Zhang Chak Fung	1155192302	1155192302@link.cuhk.edu.hk

Instructor: Dr. LAM Tak Kei

Course: CSCI3100 Software Engineering

Lab Section:

Teaching Assistant:

Date: 10/2/2025

1 Introduction

1.1 Document Purpose

This document specifies the software requirements for initial release of PetQuest, as well as providing a detailed description of the game's functionality, target audience and its design constraints. This document also serves as an overall guide for developers and stakeholders of this game to ensure the same understanding of the game's requirements.

1.2 Product Scope

PetQuest is a web-based application that combines both pet simulation and roguelike adventure, allowing players to create pets, train pets and explore dungeons.

Besides entertainment, this game also focuses on deliver meaningful morals using the story and gameplay. For example, players can learn the values of responsibility, empathy and more while taking care of their own pets and tackling challenging dungeon at the same time.

The games not only enhance player engagement through this unique pet bonding experience, but also inspires players to make thoughtful decisions. During the games, we hope this promotes positive values and life lessons that players can carry into their daily life.

1.3 Intended Audience and Document Overview

This document is mainly made for developers, testers and end-users. Detailed information such as the game's requirements, design constraints and functionality are included. In addition, the SRS document is organized into sections that cover the overall description of the game, functionality, design and implementation details. Readers are encouraged to begin by watching the overview section and proceed through other sections.

1.4 Definitions, Acronyms and Abbreviations

CS - Cascading Style Sheets
DB - Database
SRS - Software Requirements Specification
UI - User Interface
UML - Unified Modeling Language

1.5 Document Conventions

In general, this document follows IEEE formatting requirements. The text is written in Arial font with size 11 and 12, single-spaced, and maintains 1'' margins.

1.6 References and Acknowledgments

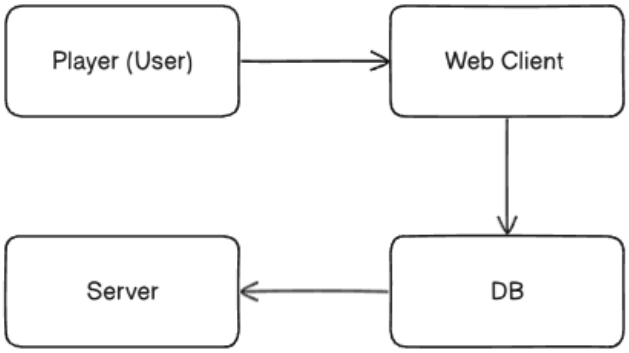
https://cs.gmu.edu/~rpettit/files/project/SRS-template.doc

2 Overall Description

2.1 Product Overview

The Pet-Raising Roguelike Game is a standalone web-based application that merges the nurturing part of pet simulation with the procedural generation and challenge of a roguelike adventure. The product does not depend on any pre-existing game components; However, it fits into the larger domain of interactive online experiences aimed at providing both casual and strategic gameplay.

Below is a conceptual diagram illustrating how different components interact:



1. Player interacts with the game's user interface through the web client.
2. The server processes requests, enforces game logic, and validates data.
3. The database stores and retrieves persistent information such as pet stats, user accounts, and dungeon records.

2.2 Product Functionality

2.2.1 2.2 Product Functions

At a high level, the game must provide:

1. Pet Creation and Adoption
    - Players adopt a new pet from a set of species, each with distinct base attributes.
    - A custom naming feature that fosters player-pet bonding.
2. Pet Raising and Training
    - Variety of training sessions or mini-games to increase specific stats such as Strength, Agility, Intelligence, and Bond Level.
    - Feeding mechanics to regulate pet health and bolster certain attributes.
3. Roguelike Adventures
    - Procedurally generated battle scenes and levels for exploration, featuring diverse enemies, traps, and loot.
    - Turn-based combat where a pet's trained stats and skills determine outcomes.
4. Progression and Resource Management

- Skill tree or perk system that unlocks unique abilities and evolutions for the pet.
- Currency and item inventory offering gear upgrades, consumables, and cosmetic options.

5. Player Feedback and Progress Tracking
    - User interface elements (notifications, progress bars, dashboards) to display real-time pet stats and dungeon states.
    - Persistent profile that saves pet progress, achievements, and unlocked features.

2.2.2 2.3 Product Functions

The target audience for this game includes:

- Casual Players: Interested in nurturing their pet daily and slowly discovering new features.
- Core Gamers: Seeking deeper roguelike mechanics and strategic gameplay.
- Completionists: Motivated by collecting all pet types, completing all dungeons, and achieving maximum stats.

While the game is primarily designed for individual play, future iterations could include social or multiplayer elements.

2.2.3 2.4 Product Functions

- Main Platform: Modern desktop or laptop browsers (e.g., Chrome, Firefox, Edge).
- Server Environment: Cloud-hosted or on-premise server running Node.js (or similar) for game logic, with a relational or NoSQL database.
- Network Requirements: Stable internet connection for real-time synchronization of pet stats and roguelike events.

2.3 Design and Implementation Constraints

- Use the for software design and UML for architectural modeling.
- Developed using web standards (HTML5, CSS, JavaScript) and other modern frontend frameworks
- Target device profiles primarily include desktops; mobile optimization is a lower priority for the initial release.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

A home screen that contains the pet in the center and basic attributes on the side, such as the pet's name, level, mood, statistics and more (customizable by the user). There are also large quick-action buttons near the bottom center for feeding, adventure and similar actions. The navigation bar will be at the bottom of the screen, allowing users to switch to different tabs easily, like inventory, shop, dungeon and profile.

Inventory and shop interface will show items in a matrix and they can be sorted by type and filtered, price and cost of items will be shown clearly below the item.

Dungeon lobby will show an overview of available dungeons and their difficulty levels (e.g., Tier 1, Tier 2), as well as potential rewards or loot. Users can select their companion and equipment at this stage before entering the dungeon.

3.1.2 Hardware Interfaces

Supports any generic keyboard, mouse or touchscreen for input and game navigation, no specialized hardware or sensors required.
The game server will be hosted on a cloud server to handle database queries and game logic.

3.1.3 Software Interfaces

Server-side APIs like RESTful to authenticate players, save/load data, manage inventory, generate dungeons and handle interaction events. Endpoint security is enforced with tokens or sessions.
A database responsible for storing user accounts, pet statistics, inventory and histories
A payment gateway like Paypal or Stripe (Optional)

3.2 Functional Requirements

3.2.1 Pet Creation and Adoption

F1.1: The system shall allow players to create a new pet by specifying a species (or choosing randomly) and a pet name.

F1.2: The system shall initialize the adopted pet with base stats and assign it a unique database record linked to the user's account.

3.2.2 Pet Raising and Training

F2.1: The system shall provide training activities (e.g., obstacle course, puzzle) that increase pet attributes (Strength, Agility, Intelligence).

F2.2: The system shall allow players to feed their pet daily, which can have positive or negative effects (e.g., boosting health, raising bond level).

F2.3: The system shall enforce a daily limit or cooldown for training activities to prevent infinite stat gains within a short period.

3.2.3 Roguelike Adventures

F3.1: The system shall offer either turn-based or semi-real-time combat actions (Attack, Defend, Use Skill, Use Item).

F3.2: The system shall populate each dungeon with random enemy types, puzzles, or traps based on difficulty tiers.

F3.3: The system shall calculate outcomes using pet attributes, enemy stats, and any relevant equipment or buffs.

3.2.4 Progression and Resource Management

F4.1: The system shall allow players to manage an inventory of items (healing potions, equipment, rare artifacts).

F4.2: The system shall enable the purchase, sale, or crafting of items via an in-game shop or crafting station.

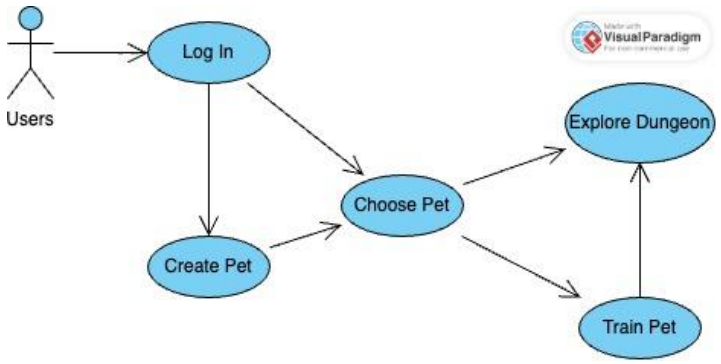
F4.3: The system shall grant currency, items, and experience points upon completing a dungeon run or defeating enemies.

3.2.5 Player Feedback and Progress Tracking

F5.1: The system shall store all user and pet data (stats, items, achievements) on a secure server. As well as provide regular data backups and recovery options in case of server outage or data corruption.

F5.2: The system shall provide and display real-time pet actions and stats to the player through UI.

3.3 Use Case Model



further use case will be create after discussion

3.3.1 Use Case #1 - U1: Create Pet

Author – Jack, Wong Po Hing

Purpose - Starting points of the game, allow player to create a new pet by specifying a species and its name.

Requirements Traceability – Systems initialize and assign the pet to a unique database record linked to Player account

Priority - High, as player need a pet to start the game

Preconditions - Player must create and log in to their account

Post conditions - A new pet is created, added and linked to the player's account. The pet's base stats are initialized

Actors – Player, system

Extends – None

Flow of Events

1. **Basic Flow**
    1. The player selects the option to create a new pet.
    2. The system prompts the player to choose a species and enter a pet name.
    3. The player selects a species and enters a name.
    4. The system creates the pet with the chosen species and name, initializes its base stats, and links it to the player's account.
    5. The system confirms the creation of the pet and displays its details to the player.
2. **Alternative Flow**
    1. If the player chooses the random option for species, the system selects a species randomly.
3. **Exceptions**
    1. If the player has reached the maximum number of pets, the system displays an error message and does not proceed with pet creation.

Includes - None

Notes/Issues - Make sure the pet creation process is user-friendly, maybe adding a help section for new players

3.3.2 Use Case #2 - U2: Train Pet

Author – Jack, Wong Po Hing

Purpose - Allow player to train their new pet to increase specifying attributes such as strength/energy

Requirements Traceability – Systems provide training activities that can increase attributes, and also enforce a daily limit or cooldown for training activities to prevent infinite stat gains within a short period.

Priority - High

Preconditions - Player must log in to their account and select the active pet

Post conditions - After different training activities, attributes are increased. System updates the pet's stats and enforce a daily limit or cooldown on further training

Actors – Player, system

Extends – None

Flow of Events

4. **Basic Flow**

    1. The player selects their new pet.
    2. The system displays available training activities
    3. The player selects the training activities
    4. The system processes the training activities and updates the attributes based on the result
    5. The system enforces a daily limit or cooldown for training activities to prevent infinite stat gains within a short period
    6. The player end the training activities
    7. The system display the updated pet stats to player

5. **Alternative Flow**
    1. If the player chooses a pet that already reached the daily limit, the system will display a message that the pet is in cooldown period
6. **Exceptions**

    1. If the player does not select a pet, the system will force the player to choose a pet for training activities

Includes - None

Notes/Issues - Make sure the training activities are interesting and meaningful, which can attract the players interest.

4 Other Non-functional Requirements

4.1 Performance Requirements

4.1.1 Loading time

Under normal broadband connections, the game website should be able to load within 5 seconds and finish loading all essential information about the game such as the accessed account's past data and default visual elements. After the user starts the game, it should be ready for interaction in less than 10 seconds to keep a smooth gaming process, a brief animation about the game (e.g. showing the pet) can be shown during the loading screen to make the process less blend.

4.1.2 Display quality

Our game is running at 30 fps for most situations on default and can be adjusted to 60 fps for users with higher standards. Stutters can be further reduced by allowing users to toggle between high performance mode or high quality mode. (for example the particle effect will be reduced, transitions will be removed, the monsters' move set will be reduced)

4.1.3 Device compatibility

Our game should be able to run on any device with a monitor and online service, the game's user interface should be adaptive and functional across a reasonable range of screen sizes from desktops to smartphones. The graphics and UI elements can also be scaled to appropriate sizes for clarity and clearness of the system.

4.1.4 Resource management

The game will both efficiently use the CPU, GPU and the memory of the device. Lazy loading and background preloading are implemented to optimize performance. Previously used resources are selectively deleted if they are not being used in the future anymore to save execution time.

4.2 Safety and Security Requirements

4.2.1 User authentication

Our game will use the users' personal accounts to store and record game data, users can choose using information like Google account, Facebook, or phone number to authentiate, a return verification message will then be sent back to the users to check their accounts validity. Https connection will be used and all sensitive information will be encrypted on both ends before transmission to protect users' privacy.

4.2.2 Game Integrity

All processes about the game logic will be done on the server-side to prevent malicious users from exploiting certain game logic or loopholes on their devices. Users' game records will be examined to ensure that no abnormal item collections or in-game transactions have occurred, this can help keep the game fair for every user.

4.2.3 Server Security

All connections between the users' personal devices and the server side will be guarded by firewalls, users cannot directly access the server information or alter settings on the server side, to protect the server from malicious attacks. Security checks will also be carried out regularly to eliminate bugs or loopholes by technicians, potential security upgrades can be done based on the test results.

4.2.4 Maintain safety on unknown connections

A warning will be given to mobile users who are connected to unknown public Wifi since these internet connections lack protection. Services like Cloudflare will be used to block away non-human access to the website, this can prevent the use of scripted robots and unauthorized access.

4.3 Software Quality Attributes

4.3.1 Reliability

The game should have a server uptime of 95% or higher with regular downtime for maintenance checking every 2 weeks for a few hours. Updates to the game will also be done during the server down time such that the game can have a more regular improvement with a given schedule which also helps the users plan when to access our game. There will be an autosave function implemented to the game such that the game data will be recorded after the completion of each stage of every level and after progression of levels. Sudden shutdown of the game leaves a notice to the account such that the next login can check for a recovery plan to allow users to resume their journey.

4.3.2 Usability

The user interface will be intuitive and designed with a simple style to allow clearer and faster understanding of the game flow. Button displayed will mostly in a symbol form, hovering above the symbol with cursor or pressing on the button enough time on touchscreen will have a text box popped out showing more detailed information about the button. Tutorial levels are given to provide step by step guidance to

newcomers and they can be revisited from the menu. A small catalog of the unique names in the game will also be included for the users' reference.

4.3.3 Maintainability

The code written should be modular and well-documented to facilitate any basic updates and even debugging, since pets and dungeons are the gimmicks of our games, it is important that these elements can be updated and added frequently and conveniently. A comprehensive report about the testing of the game should be submitted and stored for future use after each regular maintenance check. A reporting system will be implemented inside the game such that users can express their opinions about the game to boost the connections between developers and gamers, and newly discovered bugs can be reported to officials immediately.