Deliverable #1 Template

SE 3A04: Software Design II – Large System Design

1 Introduction

This is the Software Requirement Specification for Hacker Sim.

1.1 Purpose

The purpose of the document is to lay out a description and intention of the software that is being developed for both the client and the developers. It will convey the software and hardware requirements of the game, as well as the design constraints of the software. Furthermore, this document is used to define our product, its functionality and performance which will be used to verify the product in the testing process.

1.2 Scope

The purpose of the proposed software, Hacker Sim, will be an enjoyable, interactive game that will allow the user to grow their specimen and beat their highscores. The focus of the application is to grow a Software Engineer in their room. Both the Software Engineer and their room can be upgraded using the in-game currency. The in-game currency will be acquired through the completion of tasks. Online capabilities will give users the ability to share their rooms and chat as well as give gifts to other users to help complete tasks.

1.3 Definitions, Acronyms, and Abbreviations

SE - Software Engineer

1.4 References

Dormehl, L. (2019, May 29). The Tamagotchi Effect: How Digital Pets Shaped The Way We Use Technology. Retrieved from https://www.digitaltrends.com/cool-tech/how-tamagotchi-shaped-tech/

Technologies, U. (n.d.). Unity. Retrieved from https://unity.com/

1.5 Overview

The document is organized in the following manner: Overall Description, Use Case Diagram, Functional Requirements, Non-Functional Requirements. Overall Description focuses on providing details about how the ideas link together to enable specific functionality and showing functionality through comparison as well as outline who the users are intended to be and the constraints. Use Case Diagrams demonstrate how the stakeholder carries out business events. Functional Requirements explains the functionality of the system. Finally, the Non-Functional Requirements focuses on the required qualities of the software.

2 Overall Description

2.1 Product Perspective

The product is a system modeling application for desktop computers based on the Unity platform using C#. The product design itself is not a continuation of any predecessor products, but in the past there were many

implementations with a similar concept; raising a digital being. The trend of virtual pet games started from mid 1990s to early 2000s, popularized by the Tamagotchis. Our product will differentiate itself by giving users an experience of raising a software engineer from the very beginning.

The system's main features:

1) A server with functionalities:

   - To provide users with online access via login/logout
   - To connect a database with the implementation of a game
   - To allow different users to share and interact with each other
   - To allow users to give feedback and report bugs

2) A database or a file management system on the user's computer for storing:

   - User account information set
   - User game progress
   - Product FAQs for convenient look-up

2.2 Product Functions

2.2.1 Function Summary

The system's functionality is centralized around the user raising their digital specimen as a Software Engineer (referred to as SE below). The user will be involved with the lifecycle of specimen and attempt to promote it with a set of means and reach end-game goal. A currency system is designed for the in-game shop and the user can earn currency from initiating interactions/events with their SE. For example, working and completing coding projects. The in-game shop contains three main categories of items: interactable items (pets and exercise equipment), furniture and room upgrades and food. The specimen has a list of metrics, i.e. tiredness, happiness, age, health, etc. The user must follow certain rules to keep metrics within a rational range, otherwise it will hinder specimen growth or lead to game failure. The user will be able to view their specimen's information to check their progress and current health.

2.2.2 List of Basic Functions

1. User creates a SE.
2. Users initiate events and interact with their SE.
3. Users purchase items from the in-game shop.
4. Users utilize items.
5. Users customize specimen's room.
6. Users view specimen's information.
7. Users reset their game stage and start a new one.

2.2.3 Function Domain

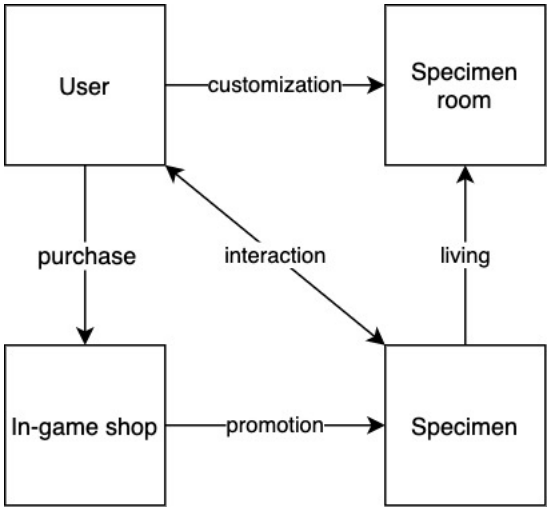


Figure 1: Function Domain

2.3 User Characteristics

Intended users do not have to meet a strict set of requirements to use our product. Users can be a PhD student or no education. They are not required to have past experience of similar types of games either. In general, it is considered that the user should:

- Have a hardware to run our product on, i.e. desktop, laptop, etc.
- Have basic knowledge of operations for computer with access to internet
- Be able to read FAQs and understand how to use the product

2.4 Constraints

Items that might limit the developer's options:

- Regulatory policies: Unity has term of service that developers need to follow
- Basic requirements expectation set by the project outline: Our product must satisfy these functionalities. Features that have a conflict with those won't be our options.
- Project deliverable deadlines and schedule: In the given limiting time, we may not be able to develop various complex functions that are not included in project outlines with complex implementations.
- Developer team's experience: Some people are using Unity platform for the first time and have finite experience with creating and designing games.

2.5 Assumptions and Dependencies

- As mentioned above in section 2.3, all requirements are based on the criteria outlined for proper use of the product. If definitions change during development, requirements will change accordingly.
- All requirements are written under the assumption that our product is able to run on users' hardware - the user's hardware supports the technology of our product.
- It is assumed that the product is running in an environment with internet connection.

2.6 Apportioning of Requirements

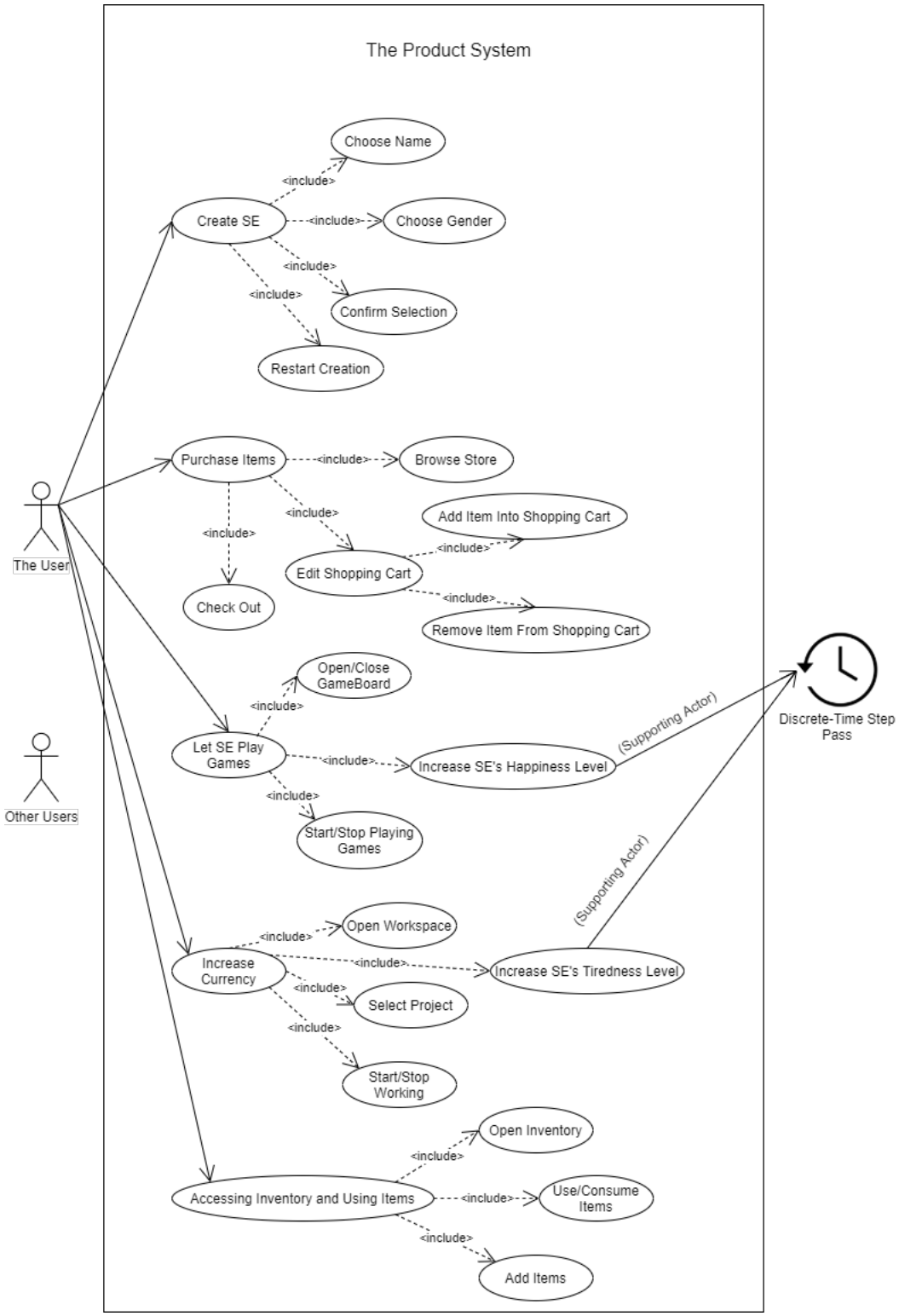
Requirements that may be delayed until future versions of the system:

- Customization of SE's room feature might be delayed as it is not a core feature to support main functions of system and it is not a requirement set by project outline
- Interaction between SEs of different users feature might be delayed if the team cannot find a proper way to achieve this function in the given time

3 Use Case Diagram

In this section, we have three Use Case diagrams to illustrate the required business events and their interactions with each other and the user.

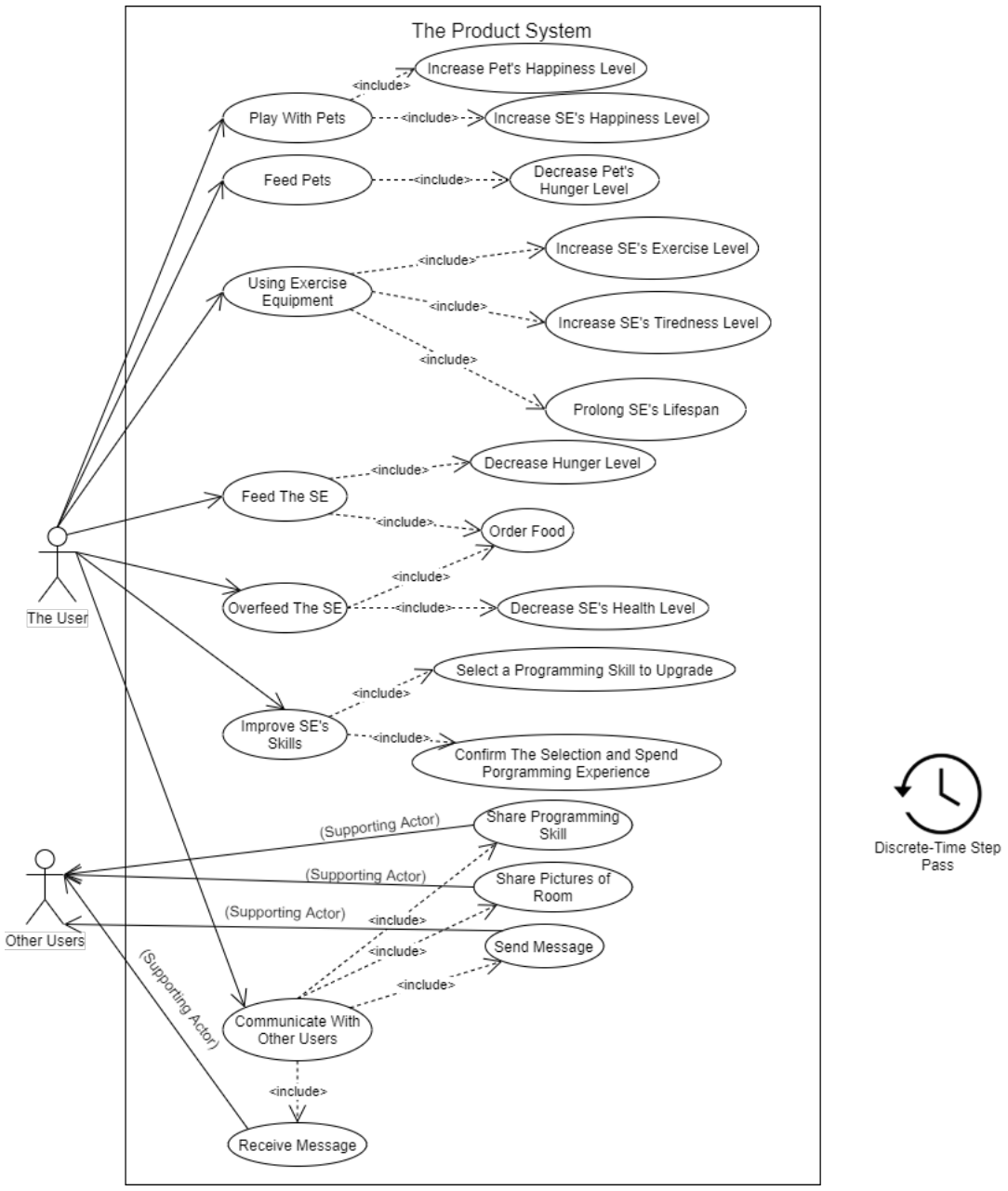
Use Case Diagram 1:



1. Create SE: The user creates his SE, by customizing the basic properties of the SE.
2. Purchase Items: User purchases in-game items for his SE with in-game currency.

3. Let the SE Play Games: The user can let his SE play games to gain happiness.
4. Increase Currency: The user increases his SE's in-game currency by letting SE work on chosen software projects which would also tired the SE.
5. Accessing Inventory and Using Items: The user accesses the SE's inventory, adds items into inventory and uses the items inside.

Use Case Diagram 2:



1. Play With Pets: The user lets SE play with its pet and gain both its and it's pet's happiness.
2. Feed Pets: The user lets SE feed its pet and decrease its pet's hunger.

3. Using Exercise Equipment: The user lets the SE take exercises with its exercise equipment. Letting the SE exercise would keep the SE fit, tired the SE, and also prolong the SE's lifespan if the exercise equipment have been used frequently.

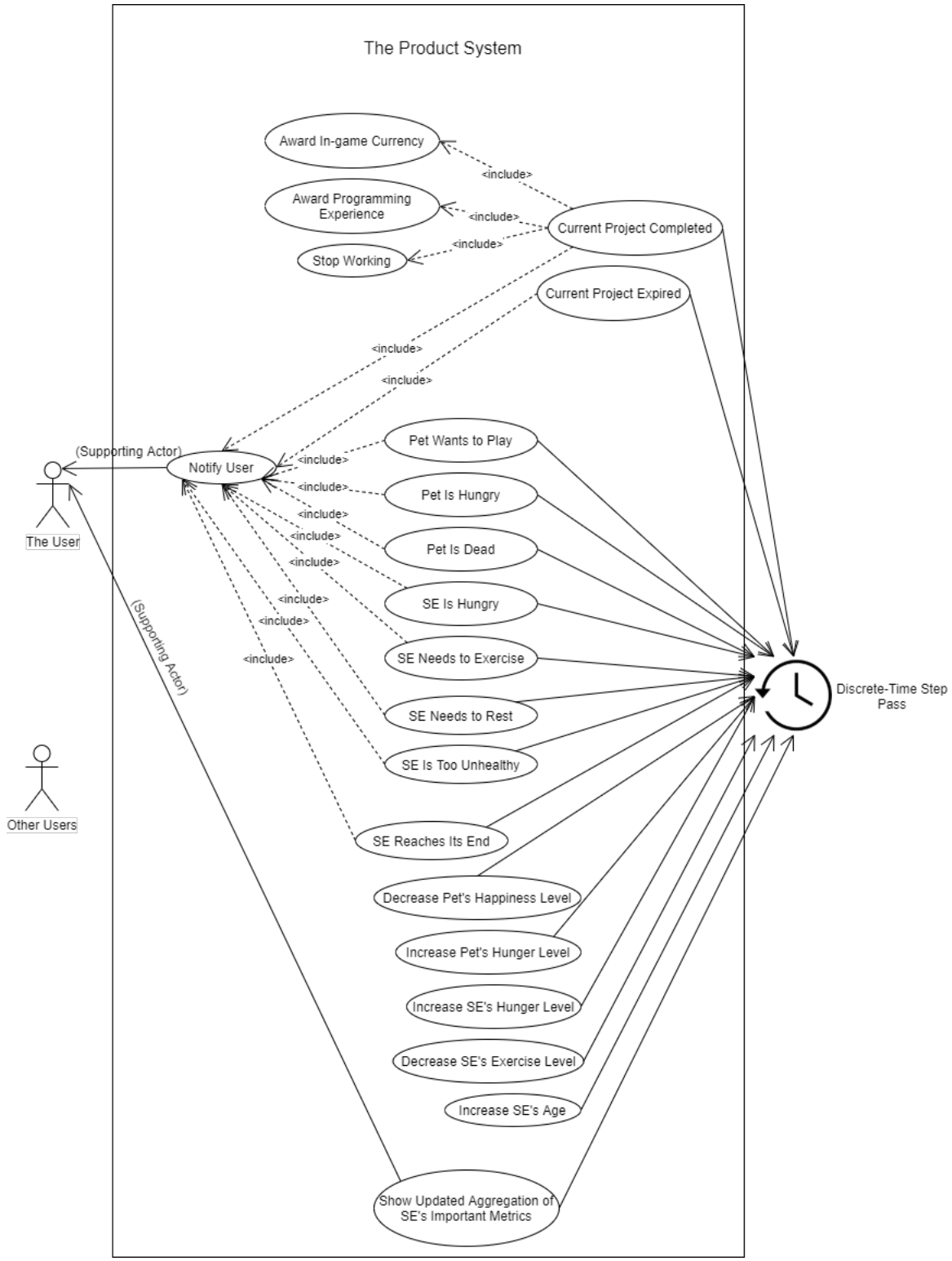
4. Feed The SE: The user shall be able to feed the SE by ordering food in-game. Feeding the SE would decrease the SE's hunger.

5. Overfeed The SE: The user overfeeds the SE by ordering food in-game when the SE is not hungry. Overfeeding the SE could make the SE's unhealthy.

6. Improve SE's skill: The user upgrades the SE's programming skills, by selecting the skill which the user wants to upgrade, confirming his selection and spending certain amount of SE's in-game programming experience to complete upgrading. Upgrading programming skills could increase SE's income.

7. Communicate With Other Users: The user shall be able to communicate through the system with other users. By communicating with other users, the user can send/receive messages to/from other users, share pictures of the SE's room and the SE's current programming skills to other users.

Use Case Diagram 3:



Use-case diagram 3 (references Business Event 5) outlines the interactions between the user and the systems reaction to a discrete amount of time passed. When a discrete amount of time has passed the system will

notify the user of an event that may have occurred. These events include:

- The SE's pet current state and if it needs attention (hungry, bored, dead)
- The SE's current state (tired, hungry, etc.)
- Relevant metrics for the SE
- The completion of a current project, and the awarded currency

4 Functional Requirements

This section of the SRS should contain all of the software requirements to a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements. Throughout this section, every stated requirement should be externally perceivable by users, operators, or other external systems. These requirements should include at a minimum a description of every input (stimulus) into the system, every output (response) from the system, and all functions performed by the system in response to an input or in support of an output.

You normally have two options for organizing your functional requirements:

1. Organize first by *business events*, then by *viewpoints*
2. Organize first by *viewpoints*, then by *business events*

Choose the one which makes the most sense.

For example, if you wish to organization by business events:

BE1. Business Event

VP1.1 Viewpoint

i. Requirement
ii. Requirement
iii. ...

VP1.2 Viewpoint

i. Requirement
ii. Requirement
iii. ...

VP1.3 ...

BE2. Business Event

VP2.1 Viewpoint

i. Requirement
ii. Requirement
iii. ...

VP2.2 Viewpoint

i. Requirement
ii. Requirement
iii. ...

VP2.3 ...

OR, if you wish to organization by viewpoints:

VP1. Viewpoint

BE1.1 Business Event

i. Requirement
ii. Requirement
iii. ...

BE1.2 Business Event

i. Requirement
ii. Requirement
iii. ...

BE1.3 ...

VP2. Viewpoint

BE2.1 Business Event

i. Requirement
ii. Requirement
iii. ...

BE2.2 Business Event

i. Requirement
ii. Requirement
iii. ...

BE2.3 ...

5 Non-Functional Requirements

5.1 Look and Feel Requirements

5.1.1 Appearance Requirements

LF1.

5.1.2 Style Requirements

LF1.

5.2 Usability and Humanity Requirements

5.2.1 Ease of Use Requirements

UH1.

5.2.2 Personalization and Internationalization Requirements

UH1.

5.2.3 Learning Requirements

UH1.

5.2.4 Understandability and Politeness Requirements

UH1.

5.2.5 Accessibility Requirements

UH1.

5.3 Performance Requirements

5.3.1 Speed and Latency Requirements

PR1.

5.3.2 Safety-Critical Requirements

PR1.

5.3.3 Precision or Accuracy Requirements

PR1.

5.3.4 Reliability and Availability Requirements

PR1.

5.3.5 Robustness or Fault-Tolerance Requirements

PR1.

5.3.6 Capacity Requirements

PR1.

5.3.7 Scalability or Extensibility Requirements

PR1.

5.3.8 Longevity Requirements

PR1.

5.4 Operational and Environmental Requirements

5.4.1 Expected Physical Environment

OE1.

5.4.2 Requirements for Interfacing with Adjacent Systems

OE1.

5.4.3 Productization Requirements

OE1.

5.4.4 Release Requirements

OE1.

5.5 Maintainability and Support Requirements

5.5.1 Maintenance Requirements

MS1.

5.5.2 Supportability Requirements

MS1.

5.5.3 Adaptability Requirements

MS1.

5.6 Security Requirements

5.6.1 Access Requirements

SR1.

5.6.2 Integrity Requirements

SR1.

5.6.3 Privacy Requirements

SR1.

5.6.4 Audit Requirements

SR1.

5.6.5 Immunity Requirements

SR1.

5.7 Cultural and Political Requirements

5.7.1 Cultural Requirements

CP1.

5.7.2 Political Requirements

CP1.

5.8 Legal Requirements

5.8.1 Compliance Requirements

LR1.

5.8.2 Standards Requirements

LR1.

A Division of Labour

Include a Division of Labour sheet which indicates the contributions of each team member. This sheet must be signed by all team members.

IMPORTANT NOTES

- Be sure to include all sections of the template in your document regardless whether you have something to write for each or not
  - If you do not have anything to write in a section, indicate this by the *N/A*, *void*, *none*, etc.
- Uniquely number each of your requirements for easy identification and cross-referencing
- Highlight terms that are defined in Section 1.3 (**Definitions, Acronyms, and Abbreviations**) with **bold**, *italic* or underline
- For Deliverable 1, please highlight, in some fashion, all (you may have more than one) creative and innovative features. Your creative and innovative features will generally be described in Section 2.2 (**Product Functions**), but it will depend on the type of creative or innovative features you are including.