# Software Requirements Specification for SFWRENG 4G06 - Capstone Design Project: subtitle describing software

Team #7, Wardens of the Wild
Felix Hurst
Marcos Hernandez-Rivero
BoWen Liu
Andy Liang

October 8, 2025

# Contents

# Revision History

| Date | Version | Notes |
|---|---|---|
| Date 1 | 1.0 | Notes |
| Date 2 | 1.1 | Notes |

# 1 Purpose of the Project

## 1.1 User Business

*Insert your content here.*

## 1.2 Goals of the Project

*Insert your content here.*

# 2 Stakeholders

## 2.1 Client

*Insert your content here.*

## 2.2 Customer

*Insert your content here.*

## 2.3 Other Stakeholders

*Insert your content here.*

## 2.4 Hands-On Users of the Project

*Insert your content here.*

## 2.5 Personas

*Insert your content here.*

## 2.6 Priorities Assigned to Users

*Insert your content here.*

### 2.7 User Participation

*Insert your content here.*

### 2.8 Maintenance Users and Service Technicians

*Insert your content here.*

## 3 Mandated Constraints

### 3.1 Solution Constraints

*Insert your content here.*

### 3.2 Implementation Environment of the Current System

*Insert your content here.*

### 3.3 Partner or Collaborative Applications

*Insert your content here.*

### 3.4 Off-the-Shelf Software

*Insert your content here.*

### 3.5 Anticipated Workplace Environment

*Insert your content here.*

### 3.6 Schedule Constraints

*Insert your content here.*

### 3.7 Budget Constraints

*Insert your content here.*

### 3.8 Enterprise Constraints

*Insert your content here.*

## 4 Naming Conventions and Terminology

### 4.1 Glossary of All Terms, Including Acronyms, Used by Stakeholders involved in the Project

*Insert your content here.*

## 5 Relevant Facts And Assumptions

### 5.1 Relevant Facts

*Insert your content here.*

### 5.2 Business Rules

*Insert your content here.*

### 5.3 Assumptions

*Insert your content here.*

## 6 The Scope of the Work

### 6.1 The Current Situation

*Insert your content here.*

### 6.2 The Context of the Work

*Insert your content here.*

### 6.3 Work Partitioning

*Insert your content here.*

### 6.4 Specifying a Business Use Case (BUC)

*Insert your content here.*

## 7 Business Data Model and Data Dictionary

### 7.1 Business Data Model

*Insert your content here.*

### 7.2 Data Dictionary

*Insert your content here.*

## 8 The Scope of the Product

### 8.1 Product Boundary

*Insert your content here.*

### 8.2 Product Use Case Table

*Insert your content here.*

### 8.3 Individual Product Use Cases (PUC's)

*Insert your content here.*

## 9 Functional Requirements

### 9.1 Functional Requirements

*Insert your content here.*

# 10 Look and Feel Requirements

## 10.1 Appearance Requirements

*Insert your content here.*

## 10.2 Style Requirements

*Insert your content here.*

# 11 Usability and Humanity Requirements

## 11.1 Ease of Use Requirements

*Insert your content here.*

## 11.2 Personalization and Internationalization Requirements

*Insert your content here.*

## 11.3 Learning Requirements

*Insert your content here.*

## 11.4 Understandability and Politeness Requirements

*Insert your content here.*

## 11.5 Accessibility Requirements

*Insert your content here.*

# 12 Performance Requirements

## 12.1 Speed and Latency Requirements

*Insert your content here.*

### 12.2 Safety-Critical Requirements

*Insert your content here.*

### 12.3 Precision or Accuracy Requirements

*Insert your content here.*

### 12.4 Robustness or Fault-Tolerance Requirements

*Insert your content here.*

### 12.5 Capacity Requirements

*Insert your content here.*

### 12.6 Scalability or Extensibility Requirements

*Insert your content here.*

### 12.7 Longevity Requirements

*Insert your content here.*

## 13 Operational and Environmental Requirements

### 13.1 Expected Physical Environment

*Insert your content here.*

### 13.2 Wider Environment Requirements

*Insert your content here.*

### 13.3 Requirements for Interfacing with Adjacent Systems

*Insert your content here.*

### 13.4 Productization Requirements

*Insert your content here.*

### 13.5 Release Requirements

*Insert your content here.*

## 14 Maintainability and Support Requirements

### 14.1 Maintenance Requirements

*Insert your content here.*

### 14.2 Supportability Requirements

*Insert your content here.*

### 14.3 Adaptability Requirements

*Insert your content here.*

## 15 Security Requirements

### 15.1 Access Requirements

*Insert your content here.*

### 15.2 Integrity Requirements

*Insert your content here.*

### 15.3 Privacy Requirements

*Insert your content here.*

### 15.4 Audit Requirements

*Insert your content here.*

### 15.5 Immunity Requirements

*Insert your content here.*

## 16 Cultural Requirements

### 16.1 Cultural Requirements

*Insert your content here.*

## 17 Compliance Requirements

### 17.1 Legal Requirements

*Insert your content here.*

### 17.2 Standards Compliance Requirements

*Insert your content here.*

## 18 Open Issues

*Insert your content here.*

## 19 Off-the-Shelf Solutions

### 19.1 Ready-Made Products

*Insert your content here.*

### 19.2 Reusable Components

*Insert your content here.*

### 19.3 Products That Can Be Copied

*Insert your content here.*

## 20 New Problems

### 20.1 Effects on the Current Environment

*Insert your content here.*

### 20.2 Effects on the Installed Systems

*Insert your content here.*

### 20.3 Potential User Problems

*Insert your content here.*

### 20.4 Limitations in the Anticipated Implementation Environment That May Inhibit the New Product

*Insert your content here.*

### 20.5 Follow-Up Problems

*Insert your content here.*

## 21 Tasks

### 21.1 Project Planning

*Insert your content here.*

### 21.2 Planning of the Development Phases

*Insert your content here.*

## 22 Migration to the New Product

### 22.1 Requirements for Migration to the New Product

*Insert your content here.*

### 22.2 Data That Has to be Modified or Translated for the New System

*Insert your content here.*

## 23 Costs

*Insert your content here.*

## 24 User Documentation and Training

### 24.1 User Documentation Requirements

*Insert your content here.*

### 24.2 Training Requirements

*Insert your content here.*

## 25 Waiting Room

*Insert your content here.*

## 26 Ideas for Solution

*Insert your content here.*

# Appendix — Reflection

**Team**

3. Thankfully, our team had no disagreements during this deliverable and were all on the same page, so this will serve as a response to everyone's Q3.

**Andy Liang**

1. Creating our development plan was crucial for several reasons specific to our ambitious project. First, our game involves complex technical challenges - procedural destructible environments, intelligent slime mold traversal, and physics-based interactions that could compound into performance issues. Without a clear plan, we could easily get lost trying to solve these problems simultaneously. The plan helped us identify our main risk early: ensuring the slime mold behavior works as intended while maintaining performance when combined with our voxel-based destructible environment. By recognizing this upfront, we can focus our proof of concept demonstration on exactly this integration challenge. Additionally, with our diverse team roles (Art Director, Character Artist, Environment Artist, Programmer, Music Director, Composer), coordination is essential. The plan establishes clear communication channels through Discord and GitHub, defines our workflow using pull requests and code reviews, and sets expectations for CI/CD implementation. Without this structure, our different specializations could easily work in isolation and create integration nightmares later. The scheduling aspect also forces us to think realistically about deliverable deadlines and break down our complex technical goals into manageable milestones.

2. Early Issue Detection: Given our concern about compounding errors between procedural systems and physics, automated testing can catch integration problems before they become major headaches. Team Coordination: With multiple people working on different systems (art, code, audio), CI/CD ensures everyone's work integrates properly and nobody breaks someone else's features. Code Quality Assurance: Our plan includes unit testing, security checks, and formatting verification, which is essential when working with C# and Unit.y Performance Monitoring: Since performance is a key risk with our procedural and physics systems, automated performance testing can flag issues early.

**BoWen Liu**

1. Creating a development plan prior to starting the project is essential in aligning the team's goal, and workflow in order to have an realistic and feasible starting point and roadmap on how to proceed in this project.

2. CI/CD improves traceability and accountability in one's work both in terms of intra/inter team development as well as for upper management in a business context. The disadvantages to using CI/CD could be low quality of work to meet rigorous and sometimes unrealistic weekly milestones as well as adding unnecessary overhead when committing deliverables.

**Felix Hurst**

1. Creating a development plan prior to starting a project ensures many aspects of proper organization. Everyone in the team knows what tasks they are responsible for, so different team members do not end up trying to do the same work, and know who to contact to ask questions about specific modules. The team has expectations set, including activity, quality, self-imposed deadlines, and meeting schedules. The team is ultimately guided by the development plan in nearly everything they do while working on the project. Without this kind of structure, team members would be spending a lot more time asking questions, causing delays in development. Or, they may underperform compared to the other team members' internal expectations. It is important that everyone is on the same page to minimize the need for future questions and minimize the possibility of conflict within the team.

2. The advantages of using CI/CD include:

    - Pull requests could be verified to meet specified tests. This ensures poorly written code is not accepted into the repository.
    - New code could be automatically built into a testable version of the project, making it faster to test.

    The disadvantages of using CI/CD include:

    - It takes time to set it up and write tests, especially those that are intended to be universal across all newly accepted code.

- It may slow down the process of merging pull requests for minor changes that don't need extra testing.

**Marcos Hernandez-Rivero**

1. Creating a development plan before starting a software engineering group project is essential because it provides a clear roadmap for the team, defining goals, scope, roles, and timelines to keep everyone aligned. It helps prevent confusion, overlap, or missed tasks by assigning responsibilities, establishes coding and documentation standards for consistency, and outlines milestones to manage time effectively. A development plan also anticipates risks to project success, and ideally sets strategies to address them.

2. CI/CD allows teams to integrate code frequently, and catch many errors early and automatically, which in the long run improves software quality and reduces the amount of bugs either on release or later down the production workflow. Some notable disadvantages though, are that it requires setting up and maintaining the CI/CD system, which can be time-consuming and/or confusing to many individuals. Additionally, any automated tests need to be thorough so that they act as a reliable tool to ensure code quality.

1. What went well while writing this deliverable?

2. What pain points did you experience during this deliverable, and how did you resolve them?

3. How many of your requirements were inspired by speaking to your client(s) or their proxies (e.g. your peers, stakeholders, potential users)?

4. Which of the courses you have taken, or are currently taking, will help your team to be successful with your capstone project.

5. What knowledge and skills will the team collectively need to acquire to successfully complete this capstone project? Examples of possible knowledge to acquire include domain specific knowledge from the domain of your application, or software engineering knowledge, mechatronics knowledge or computer science knowledge. Skills may be related to technology, or writing, or presentation, or team management, etc. You should look to identify at least one item for each team member.

6. For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?