

SE 3XA3: Software Requirements  
Specification  
*Snake 2.0*

Team 30, VUA  
Andy Hameed — hameea1  
Usman Irfan — irfanm7  
Vaibhav Chadah — chadhav

December 3, 2018

# Contents

<b>1</b>	<b>Project Drivers</b>	<b>1</b>
1.1	The Purpose of the Project . . . . .	1
1.2	The Stakeholders . . . . .	2
1.2.1	The Client . . . . .	2
1.2.2	The Customers . . . . .	2
1.2.3	Other Stakeholders . . . . .	2
1.3	Mandated Constraints . . . . .	3
1.4	Naming Conventions and Terminology . . . . .	3
1.5	Relevant Facts and Assumptions . . . . .	4
<b>2</b>	<b>Functional Requirements</b>	<b>5</b>
2.1	The Scope of the Work and the Product . . . . .	5
2.1.1	The Context of the Work . . . . .	5
2.1.2	Work Partitioning . . . . .	5
2.1.3	Individual Product Use Cases . . . . .	6
2.2	Functional Requirements . . . . .	6
<b>3</b>	<b>Non-functional Requirements</b>	<b>8</b>
3.1	Look and Feel Requirements . . . . .	8
3.1.1	Appearance Requirements :NFR1 . . . . .	8
3.1.2	Style Requirements:NFR2 . . . . .	8
3.2	Usability and Humanity Requirements:NFR3 . . . . .	8
3.3	Performance Requirements . . . . .	9
3.3.1	Speed Requirements:NFR4 . . . . .	9
3.3.2	Safety Critical requirements :NFR5 . . . . .	9
3.3.3	Precision Requirements :NFR6 . . . . .	9
3.3.4	Reliability and Availability Requirements :NFR7 . . . . .	9
3.3.5	Capacity Requirements :NFR8 . . . . .	9
3.4	Operational and Environmental Requirements . . . . .	9
3.4.1	Expected Physcial environment :NFR9 . . . . .	9
3.4.2	Expected Technological environment :NFR10 . . . . .	9
3.5	Maintainability and Support Requirements . . . . .	10
3.5.1	Maintainability :NFR11 . . . . .	10
3.5.2	Portability :NFR12 . . . . .	10
3.6	Security Requirements :NFR13 . . . . .	10
3.7	Cultural Requirements :NFR14 . . . . .	10

3.8	Legal Requirements :NFR15 . . . . .	10
3.9	Health and Safety Requirements :NFR16 . . . . .	10
<b>4</b>	<b>Project Issues</b>	<b>11</b>
4.1	Open Issues . . . . .	11
4.2	Off-the-Shelf Solutions . . . . .	11
4.3	New Problems . . . . .	11
	4.3.1 Effects on the Current Environment . . . . .	11
	4.3.2 Effects on the Installed Systems . . . . .	12
4.4	Tasks . . . . .	12
4.5	Migration to the New Product . . . . .	12
4.6	Risks . . . . .	13
4.7	Costs . . . . .	13
4.8	User Documentation and Training . . . . .	14
4.9	Waiting Room . . . . .	14
4.10	Ideas for Solutions . . . . .	14
<b>5</b>	<b>Appendix</b>	<b>16</b>
5.1	Symbolic Parameters . . . . .	16

## List of Tables

1	Revision History . . . . .	17
---	----------------------------	----

## List of Figures

This document describes the requirements for Snake 2.0 The template for the Software Requirements Specification (SRS) is a subset of the Volere template (?). If you make further modifications to the template, you should explicitly state what modifications were made.

# 1 Project Drivers

## 1.1 The Purpose of the Project

Almost everyone nowadays relies on a computer as a multipurpose tool for research, video streaming, gaming and many other tasks. With the emergence of fast computing, gaming has become a popular pastime activity and a source of entertainment for many. However, not everyone has a device powerful enough to support extensive game applications. A simple, memory-efficient application of the Snake game allows it to be accessible for gamers without the need for extensive hardware or a high-performance computer. Our team, VUA30, will be creating a desktop application for the well-known “Snake” game with new enhancements and features. This competitive and addictive game can allow the user to play at their own pace and challenge their own high score.

Buying a computing device with high storage and faster performance can be out of budget. Complicated software covers up all the storage and the user is bound to use these applications as opposed to downloading other software. The importance of the redevelopment of “The Snake” is to save computing device’s personal storage and allow the user to play a game 24/7 with strong performance, even offline. Creating a desktop version of the snake game can fit into the category of downloadable calssical games such as the solitaire suite. The recreation of this game will allow the user to enjoy the classical game anytime and anywhere as long as they have installed the application. Improving aspects such as graphics and custom speed will also make the game more interesting. We would like to add more features to the game to make it more customizable and help people enjoy the classical game in an exciting and new way.

## **1.2 The Stakeholders**

Stakeholders involved will be contained within the gaming community, more specifically the desktop gaming community and casual PC owners who are looking for a fun reliever for boredom or quick game to play. This also includes members invested in the project which are mentioned in the subsections below.

### **1.2.1 The Client**

Since this game is a separate entity, the clients are the designers in this project team. In further developments and upon increase in game popularity, the clients could be a desktop gaming distribution service such as steam, google play or apple store. Otherwise, the main client would be Dr. Bokhari who has assigned the project.

### **1.2.2 The Customers**

The main users or customers are desktop gamers, older generation of game enthusiasts, youth and teens. However, the client can be anyone with a PC and an interest in classical gaming or a sudden craving for playing the classical Snake game. Often times, these games are a quick fix to boredom for those who are casually browsing their PC's, so the game will be designed to provide enough stimulus and excitement for regular computer users, similar to the solitaire suite.

### **1.2.3 Other Stakeholders**

Aside from the clients and customers, other stakeholder include 3rd party Desktop game distribution stores and open source project banks which may make use of this project for development purposes:

- 3rd party desktop game distribution stores.
- Game Testers.
- Technology Experts [Part of Project Team].
- Usability experts.
- Dr. Bokhari.

- Project Development Experts: This can include teaching assistants, the professor, experienced peers and so on.

### 1.3 Mandated Constraints

Some constraints that apply to the project include the following:

- No project budget provided; Project cannot use costly API memberships or resources.
- Application should take less than 400MB of storage space to meet requirements.
- The project must be completed within a 4-month period.
- Limited resources in terms of domain experts, specifically in graphic design.
- Application will be developed for one OS due to time constraint.
- open source project must be translated to Python due to development language and scope.

### 1.4 Naming Conventions and Terminology

The naming conventions listed below will be used to clearly define words and terminology that will come up in the project development process. Below is a list of naming conventions, terms, and special vocabulary and their meaning. Since the desktop application is straightforward, there is not much terminology being used as of now:

- DDS: Digital Distribution Service such as play store, microsoft play, etc.
- OS: Operating System.
- Python: The programming language used for application development.
- Pygame: Computer graphics Python library.
- Snake 2.0: The desktop application being developed in Python.

- The interface: The graphics developed using Pygame.
- The source game: The open source original Python snake game being used for this project.

## 1.5 Relevant Facts and Assumptions

Some factors that might affect the outcome of the product are listed as follows:

- DDS contribution will be necessary for the public release of the game.
- Contribution of the development team will affect the outcome of the product.
- Feedback from game testers.
- Availability of resources from pygame library to replicate front-end design in HTML,CSS and JS.
- Time remaining once initial objectives and goals are met. This could affect which additional functionality is added.

There are also assumptions that pertain to the intended operational environment and anything affecting the product:

- Pygame library offers enough functionality to recreate the web app graphics in Python.
- The user is using Windows for game execution otherwise they must compile the source code to run the application.
- The application will not be an exact replica of the source game. Added functionality and a change of graphics is expected.
- The game application will prioritize the completion of the snake game as the central attraction.

Some user characteristics will affect the final design and written requirements:

- Users expect the game to be responsive and timely due to the nature of wanting quick stimulus .

- The game should have an attractive user interface due to the nature of the users expectations. It is mainly used for entertainment and should have a smooth user-interface.

## 2 Functional Requirements

### 2.1 The Scope of the Work and the Product

#### 2.1.1 The Context of the Work

The scope of the project is deliver a Product that has the requirement documentation, and a desktop application that can be installed on a user's system.

To achieve the goals of the Product, the following are decided to be the deadlines of the goal to be on the track:

- Development Plan 28/09/18
- Requirements Document Revision 05/10/18
- Proof of Concept Demonstration 16/10/18
- Test Plan Revision 26/10/19
- Design & Document Revision 09/11/18
- Revise all the Documentation 13/11/18
- Final Documentation 06/12/18

#### 2.1.2 Work Partitioning

The desktop application involves different processes to successfully run: making a user-interface so the user can interact with the application, **Back-End Development** that can handle all the inputs given by the user and **updates the interface** according to the requirements. These tasks can be divided into sub-task. For example, **the system uses a mouse as an input, so whenever the user presses a button on the interface from its computer's mouse, there is**

an update in the display making the user engaged to the application. Tasks such as displaying the food, making the snake appear at random locations, in the beginning, should all be divided and can be accomplished individually by the developers, this would be more efficient to complete the project, and the respective developer would know how to test the functions they created.

### 2.1.3 Individual Product Use Cases

The user can use the system-to-be (the desktop application) to entertain themselves when they are bored. They can use the system to improve their response time, with playing the game in difficulty modes it can be more challenging and the user has to be fast. In addition, the desktop application would be a fun means of entertainment between friends as they can play turn-by-turn and challenge each other.

## 2.2 Functional Requirements

- Requirement number: FR(Functional Requirement)1

When the user sees the interface, it can select the buttons on the screen by using the computer's mouse.

- Requirement number: FR2

The user can press the UP key to move the snake's direction in the upwards direction.

- Requirement number: FR3

The user can press the DOWN key to move the snake's direction in the downwards direction.

- Requirement number: FR4

The user can press the LEFT key to move the snake's direction in the left direction.

- Requirement number: FR5

The user can press the RIGHT key to move the snake's direction in the right direction.

- Requirement number: FR6

The game should display the user's highest score.

- Requirement number: FR7

The initial location of the snake should be random whenever the user starts the game or when it restarts.

- Requirement number: FR8

The user has the option to play in three different modes: **easy, intermediate and advanced**.

- Requirement number: FR9

The desktop application shall provide a facility to play the game in different themes, e.g., Dark or Light, or choose the random option to select either theme.

- Requirement number: FR10

When the snake eats its food, its length should be increased by 3 units. For instance, when the game is started the snake's length is only 1 unit, but after eating a block of food, its new length should be 4 units.

- Requirement number: FR11

Once the snake eats its food, the food should reappear on the screen.

- Requirement number: FR12

When the user is playing the game in easy mode, the snake is allowed to move within the screen size and can pass the boundaries (the window screen edges), if the snake traverses the boundary it should reappear from the opposite direction.

- Requirement number: FR13

When the user is playing the game in either intermediate or advanced mode, the snake is allowed to move within the screen size and cannot pass the boundaries (the window screen edges), if the snake traverses the boundary it should die, and a message should prompt on the screen to restart the game.

- Requirement number: FR14

When the user is playing the game in advanced mode, the game shall offer a maze within the Gameplay to restrict the snake's location and make it more

challenging.

- Requirement number: FR15

If the snake bites itself, the game should be over and a message should prompt on the screen to restart the game.

- Requirement number: FR16

The highest score of the user shall be saved in a text file.

- Requirement number: FR17

The user can view their highest score from the main menu by pressing the Highscore button.

- Requirement number: FR18

The color of the snake changes when a different theme is selected.

## 3 Non-functional Requirements

### 3.1 Look and Feel Requirements

#### 3.1.1 Appearance Requirements :NFR1

The final product shall be a desktop application which will contain a playground with a snake in it. It shall have different theme options and buttons to select different modes to play with. In addition, there should be a help page for a new player in order to get familiar with the commands to play this game. Also, it shall show the highest score made on the specific device.

#### 3.1.2 Style Requirements:NFR2

The product should be given a modern style by adding a nice background to it with a user-friendly interface.

### 3.2 Usability and Humanity Requirements:NFR3

The application must be simple for a person aged 10 or above. It should be understandable by any person within the age group who is familiar to

the technology. No feature should restrict the player to a non-knowledgeable outcome.

### **3.3 Performance Requirements**

#### **3.3.1 Speed Requirements:NFR4**

- All valid interaction should have a maximum response time of half a second.
- The speed for snake should be customizable by the user and should not increase or decrease by itself.

#### **3.3.2 Safety Critical requirements :NFR5**

The game shall not consume any private data from the user's device.

#### **3.3.3 Precision Requirements :NFR6**

The turn for snake should match with the users key and should be done as precisely as possible.

#### **3.3.4 Reliability and Availability Requirements :NFR7**

The game should be available 24 hours, 365 days per year to the user.

#### **3.3.5 Capacity Requirements :NFR8**

The game shall not overload the clients device's memory.

### **3.4 Operational and Environmental Requirements**

#### **3.4.1 Expected Physical environment :NFR9**

The application is intended to be used anywhere, at any desktop device. It can be used in any climatic condition from harsh summers to chilly winter( given that the device is working as well).

#### **3.4.2 Expected Technological environment :NFR10**

This application should work on any desktop device as long as the device is working.

## **3.5 Maintainability and Support Requirements**

### **3.5.1 Maintainability :NFR11**

The application shall require minimum maintenance. Also, the application shall be revised every year. Also, the code should be heavily commented in order to provide ease to the developer/maintainer.

### **3.5.2 Portability :NFR12**

The application is expected to run on Windows, Mac OS and Linux environment.

## **3.6 Security Requirements :NFR13**

This is an open-ended application. However, the application must not break the privacy policy by interfering with files stored on the desktop.

## **3.7 Cultural Requirements :NFR14**

The application will not use any kind of communicating data that will offend any religion, country or user in any way. The product will give a detailed explanation in case of use of any cultural or political symbol.

## **3.8 Legal Requirements :NFR15**

The application shall comply with all national and federal laws. In addition, the application must agree to the MIT Open License.

## **3.9 Health and Safety Requirements :NFR16**

This software should not affect the health of the user by any means. Color contrast ratio between colors used in the game is at a minimum of 4.5:1 according to G18 of the W3C Web Content Accessibility Guidelines 2.0

## 4 Project Issues

### 4.1 Open Issues

Below is a list of open issues pertaining to the project scope:

- Investigating and understanding the capabilities of the Pygame library is yet to be completed.
- Integrating additional features is not decided on as of yet. It is dependant on time constraints.
- snake-game multiplayer mode is an open issue on the open source project which we may or may not choose to implement as time permits.

### 4.2 Off-the-Shelf Solutions

Although there are available solutions on developing such a game, the project team is aiming to enhance the game by producing a desktop version with added functionality.

Ready-made simple implementations of the projects are available and can be used as reference but otherwise, enhanced features will have to be created from scratch (light/ dark theme, custom player settings, high scores and so on).

### 4.3 New Problems

#### 4.3.1 Effects on the Current Environment

The Microsoft Store contains the "250k snake" app for windows, an implementation of the old-school snake game. Aside from this application, other applications that appear when searching "snake" or "snake game" do not reflect the classical snake game. By developing the snake game as a desktop app, we will be able to provide game shoppers with more options to pick from.

### 4.3.2 Effects on the Installed Systems

The existence of the 250k snake will make it difficult to push the project team's implementation of the game, Snake 2.0, into the microsoft store market successfully. However, the new snake game will fill a niche for customizability by allowing users to pick from many different settings.

## 4.4 Tasks

An article on linkedIn by Sumit Jain summarizes the steps involved in the game development process [ ? ]. In his article, he outlines 6 main steps to the game development cycle: Idea & Story, Conceptualize & Design, Technical Analysis, Development, Testing, Deployment. Considering the project scope and the redevelopment of the snake game, the main three steps involved in the development cycle are the following:

- Technical Analysis: Use reverse engineering to understand how the game was originally built and analyze the main modules/ framework used to develop the game.
- Development: Using Python and Pygame to develop the source code for the game; Analysis from the previous step will be necessary to break down the development process.
- Testing: Test using unittest in python and principles of white box and black box testing. In further developments, this would also include integration testing with the user interface and the collection of modules created for the application.

Project members should expect the development cycle to resemble the previously mentioned framework. Once the cycle has been iterated until completion of Snake 2.0, the team will move on to the deployment stage, considering options for making the game available on a DDS such as the Microsoft Store.

## 4.5 Migration to the New Product

Snake 2.0 will require the following conversion tasks:

- Converting JS,HTML and CSS graphics and animations to Pygame graphics.

- Converting the source project into modularized step-based tasks.
- Converting from JS,HTML and CSS source code to Python source code.

The source project will be run with Snake 2.0 for performance comparison and visual feedback on the accuracy of the redevelopment as well as the enhance features that were added to snake 2.0.

## 4.6 Risks

Snake 2.0 will be a classical desktop application and therefore does not present many risks to the user or any stakeholders involved. In terms of taking risk to advance the project, there is risk in striving for the completion of a multiplayer mode for the game since it may take substantial time and effort. However, this risk is low since the project requirements have already been met and other features of the game have been enhanced, aside from the addition of a multiplayer mode.

In the case that more risks are perceived in the future, the project team will take the following course of action to come up with early warnings:

- If the development is taking place 1 week prior to the project deadline, an early warning will be issued and the group must decide to continue or dismiss the development.
- If the development is currently taking place with 2 weeks left until the project deadline and less than 50% of the development is in place, it will be dismissed.
- If the main project is missing any component (testing, code modularization, documentation, commenting, etc.) no development will proceed until the main requirements (minimum requirements) are met.
- If any of the main project components are deemed to have lower quality, a warning is issued and the team members must discuss whether to continue with further development or improving the existing product.

## 4.7 Costs

As mentioned in the development plan document, team members will be dedicating 2 hours outside of lab time for team meetings and discussions

along with 5 hours of individual work on the project itself. Since the project is open source and uses open libraries such as Pygame, the monetary cost is \$0. However, there may be additional costs to publishing Snake 2.0 with a DDS.

## 4.8 User Documentation and Training

The user will be provided with the following documentation and training:

- Snake 2.0 User Manual: The document will explain the basic permisses of the game, user settings, graphic themes, menu headings, and any other information necessary for the user to understand the features of the game.
- Snake 2.0 Installation Manual: Provided that the user will not be using Windows or the native OS that is decided on, this document will provide simple installation instructions for compiling the code on different OS's.

## 4.9 Waiting Room

In future releases of the project, the following requirements might be included in the revised requirements document:

- Snake 2.0 User Manual - Multiplayer Mode: A section explaining how to connect and play the snake game with friends
- additional 'multiplayer mode' module: A separate module to encapsulate the multiplayer mode
- additional 'themes' module - a module encapsulating the different graphic themes available for the game

## 4.10 Ideas for Solutions

Some rudimentary ideas for project modules and solutions have been mentioned down below:

- Classes/modules for individual objects like the snake, food block, the frame, the menu bar, the settings bar and so on.

- import graphics developed in adobe illustrator into the game as characters, props and so on.
- the snake class can have method that correspond to the snakes functionality such as moveLeft, moveRight, moveUp, moveDown, and Lengthen.
- the food item can have a randomPlacement method for when being placed at random around the window.
- UI: a custom header section can contain the entry fields for custom speed and other important parameters.

## 5 Appendix

NA

### 5.1 Symbolic Parameters

NA

Table 1: **Revision History**

<b>Date</b>	<b>Version</b>	<b>Notes</b>
Oct 5, 2018	1.0	Andy worked on Project Drivers and Project Issues. Usman worked on Functional requirements. Vaibhav worked on Non-Functional Requirements
Dec 2, 2018	1.1	Vaibhav is performing revision 1 in order to improve the quality of the document
Dec 2, 2018	1.1	Andy edited the formatting issues stated in feedback for Revision 1
Dec 2, 2018	1.1	Usman is performing revision 1 in order to improve the quality of the document