NTNU

Kunnskap for en bedre verden

Department of Computer Science

TDT4240 - Project

# Requirements

*Authors:*


| NTNU email | Last name | First name |
|---|---|---|
| yohans@ntnu.no | Saba | Yohan |
| @stud.ntnu.no | etternavn | fornavn |
| @stud.ntnu.no | etternavn | fornavn |
| xxxx@stud.ntnu.no | etternavn | fornavn |
| eliaswhe@stud.ntnu.no | Heimdal | Elias Ward |
| @stud.ntnu.no | etternavn | Snorre Kværnø |
| @stud.ntnu.no | Åsetternavnheim | fornavn |


*Game name:* xyz

*Group number:* 17

*Chosen COTS:* xyz

*Primary quality attributes chosen:* xyz

*Secondary quality attributes(s)* xyz

Spring 2025

# Table of Contents

# 1 Introduction

## 1.1 Description of the project and this phase

This project will carry out the development of a multiplayer online game on Android devices. The game is inspired by Slither.io [**<empty citation>**] and will contain many of the same qualities. Before developing the game, it is necessary to go through a requirement phase. In this phase, the foundation of the project will be cemented. This document will explain how this will be done, by going through functional and quality requirements of the game, as well as potential challenges with the commercial off-the-shelf frameworks that will be used.

## 1.2 The game

The objective of the game is to control a noodle figure around a wide area and eat wheat or rice which makes you larger. You play against others trying to defeat and consume them in order to grow the largest and longest in the game. The game will borrow mechanics from Slither.io as seen in Figure 1.

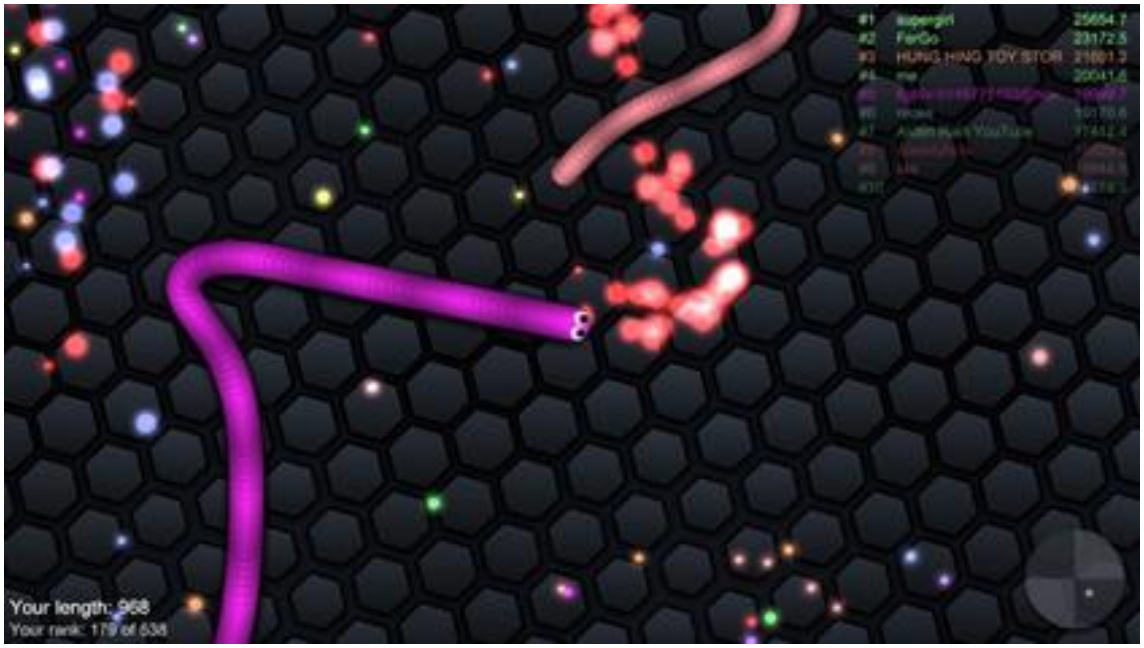


Figure 1: Screenshot from Slither.Io

The game has a map that should represent a bowl of noodles, and each player is a noodle. If another noodle crashes with its head into the body of your noodle, they are eliminated and their noodle is evaporated into wheat and rice for other players to eat. The game will be touch based and the noodle will move in the direction of the touch input it is given. Players are also able to customize their noodles before a game and will have the option to play offline against AI as well as online against their friends.

# 2 Functional requirements

| ID | Description | Priority |
|---|---|---|
| FR1 | The user should be able to control game with touch gestures | High |
| FR2 | The noodle must grow longer when consuming wheat or rice scattered across the map or when eating the remains of defeated players | High |
| FR3 | The user should be able to play online multiplayer with known players | High |
| FR4 | User should be able to change skins of their noodle between a set variability | Medium |
| FR5 | Players should be able to set and display a custom name | Medium |
| FR6 | Players spawn in random locations on the map upon game start | Medium |
| FR7 | Players can accelerate temporarily at the cost of losing some size | Low |
| FR8 | The game should have a main menu which includes options to start a game, customize the noodle, check settings, and view leaderboards | High |
| FR9 | The in-game HUD should display all players' current size/score, in order | Medium |
| FR10 | The in-game HUD should display an elimination message when a player is defeated | Low |
| FR11 | At the end of each game, the scores of the game should be displayed to all users | Low |
| FR12 | The server should keep all player movements synchronized with minimal latency | High |
| FR13 | The game map must resemble a bowl of noodles with scattered wheat and rice | Low |
| FR13.1 | Wheat and rice should spawn dynamically to maintain a balanced game | Low |
| FR13.2 | Players must not be able to exit the map | High |

Table 1: Functional Requirements for Noodle game

# 3 Quality Requirements

## 3.1 Modifiability

| | |
|---|---|
| **ID:** | M1 |
| **Source:** | Developer |
| **Stimulus:** | Wishes to add a new power-up |
| **Artifacts:** | PowerUpManager class, PowerUpFactory class, GameState class |
| **Environment:** | Design Time |
| **Response:** | The new power-up should be implemented with minimal code changes, and integrated seamlessly into the existing system without affecting other components. |
| **Response Measure:** | The power-up should be added and tested in less than 1 hour, with no more than one module impacted by the change. |

Table 2: Modifiability: Adding a New Power-up

| | |
|---|---|
| **ID:** | M2 |
| **Source:** | Developer |
| **Stimulus:** | Wishes to modify game rules |
| **Artifacts:** | GameRules class, GameLogic class, Configurations file |
| **Environment:** | Design Time |
| **Response:** | Changes made should not require modifying unrelated modules, preserving system modularity. |
| **Response Measure:** | Game rules modified and tested in under 1 hour, affecting no more than one module. |

Table 3: Modifiability: Modifying Game Rules

## 3.2 Usability

| | |
|---|---|
| **ID:** | U1 |
| **Source:** | Player |
| **Stimulus:** | Wishes to customize the appearance of their noodle |
| **Artifacts:** | NoodleCustomization class, PlayerProfile class, UI Configuration file |
| **Environment:** | Runtime |
| **Response:** | Players should be able to easily change the appearance of their noodle through the game settings. |
| **Response Measure:** | Customization changes reflected immediately, with no more than 1 second delay. |

Table 4: Usability: Noodle Customization

## 3.3 Performance

| | |
|---|---|
| **ID:** | P1 |
| **Source:** | System |
| **Stimulus:** | Increase in number of players in a session |
| **Artifacts:** | NetworkManager class, GameSession class, Server Configuration file |
| **Environment:** | Runtime |
| **Response:** | The system must maintain smooth gameplay and low latency even as player count increases. |
| **Response Measure:** | Latency must remain below 100ms for up to 50 players per session. |

Table 5: Performance: Player Count Impact on Latency

### 3.4 Testability

| | |
|---|---|
| **ID:** | T1 |
| **Source:** | Developer |
| **Stimulus:** | Wishes to test the collision mechanics |
| **Artifacts:** | CollisionDetection class, PhysicsEngine class, Test-Suite file |
| **Environment:** | Testing Phase |
| **Response:** | The collision mechanics should be isolated and tested independently of other modules. |
| **Response Measure:** | Tests for collision mechanics completed and results verified in less than 30 minutes. |

Table 6: Testability: Collision Mechanics Testing

For a video-game, performance is a crucial point and can vary depending on the hardware on which the game is being run. If the performance of a game is not good, the user experience be impacted and it can lead to real disaster. Therefore, we need to be self-conscious of the important of it and spend enough time to ensure that our performance stay stable even in the worst scenario. We already thought about what in our game may cause performance issue. The first one could be on the collision detection when some noodles enter in contact together. At this moment, some calculation will be done to resolve this collision and it can impact the performance of the game if the part of code doing that is not enough optimized. In addition, the collision resolution should be send to the other player(s) which add another layer of calculation on top of it. When starting developing this game mechanic, we will try to be very cautious on the code responsible for that.

## 4 COTS Components and Technical Constraints

### 4.1 LibGdx

Among all the COTS we will use for this project, Libgdx is the most important one. Libgdx is a cross-platform java game development framework based on OpenGL which works on all the main OS (Windows, Linux and macOS), Android, iOS and your browser. Many quite successful games have been produced and released using Libgdx and some of them are really impressive, which highlights the quality of this framework. The graphical part of the engine allow to work with some advances features such as shaders, meshes, framebuffer and more... It also offers a pre-made 2D UI library, which can be interesting to use because creating UI from scratch is not an easy task. Nevertheless, these are the many reasons why we decided to go with LibGDX, it is a really good framework that is more robust than maybe pygame but less complicated that like Monogame (XNA) or others, so it's a very good compromise to learn game developpement and more generally software developpement.

### 4.2 Android SDK

For our game being able to run on smartphone, we decided to use the Android SDK (Smartphone Developement Kit), which give tools to developp smartphone application quite easily. While using it, we should therefore be carefull that our game can run on all android devices regardless of their screen size, its hardware or even it's own configuration. This sould be one of our main point of concern during this project and thus we should do many tests on different android phones during the developement to assure that our game will be able to run properly with the same quality on the big majority of android smartphone.

### 4.3 SupaBase

To store our data like the leaderboard of a party, we need to use backend sercive that offers database. One of the obvious choice will be to go for firebase, but there is out an alternative to it called Supabase that works quite the same but with some differences. We also decided to use supabase over firebase because some people in this group already has experience using which is an advantage.

## 5 Issues

## 6 Changes

# Appendix