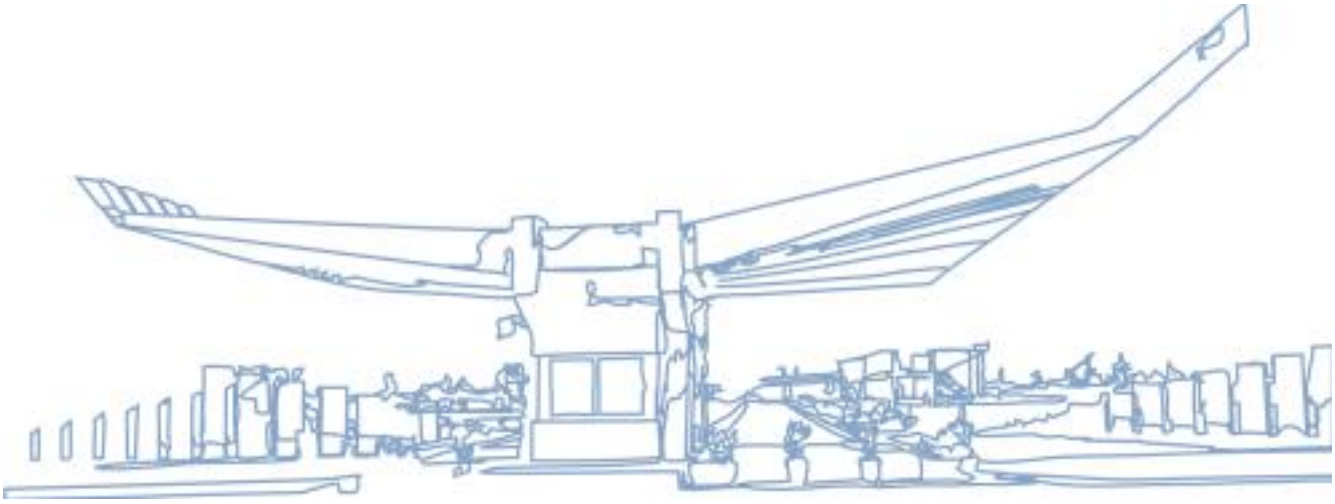
INTRODUCTION TO SOFTWARE ENGINEERING

Snake Game



Team Members:

Kristi Samara

Engjëll Abazaj

Indrit Ferati

Snake Game Requirements Specification

TABLE OF CONTENTS

1. Executive Summary

1.1 Project Overview

A well-known and traditional game is the snake game. It involves a snake that consumes a specific item that happens to emerge randomly. As it consumes more food, the snake grows bigger and moves faster. The snake's own body or walls should not be touched. Due to the lack of specific skill requirements, the game is appropriate for almost any age group. Everyone may play it on a computer or a mobile device, and it is highly entertaining and interesting.

1.2 Purpose and Scope of this Specification

In scope

The following specifications describe the purpose of this product and how it intends to be used:

- The purpose of the game is to provide an engaging and fun gameplay experience for players.
- Objective: Moving the snake and consuming as much food as you can while avoiding collisions with the walls or the snake's own body is the goal of the game.
- Gameplay: Grid-based gameplay will be used, and the snake will travel in a predetermined route across the grid. If the snake strikes a wall or its own body, the game is over.
- Controls: The snake can be moved in the game by using the arrow keys or other designated keys.
- Scoring system: The snake will receive points for each food item it consumes.
- Power-ups or bonuses: The snake may be able to collect special goods that grant perks like faster speed or more points and this could be implemented on a future release.
- Technical requirements: The game must function on desktops and mobile platforms and might need particular programming languages or libraries.
- User interface: The interface of the game will be user-friendly, and the player will receive feedback that is concise and simple to understand.

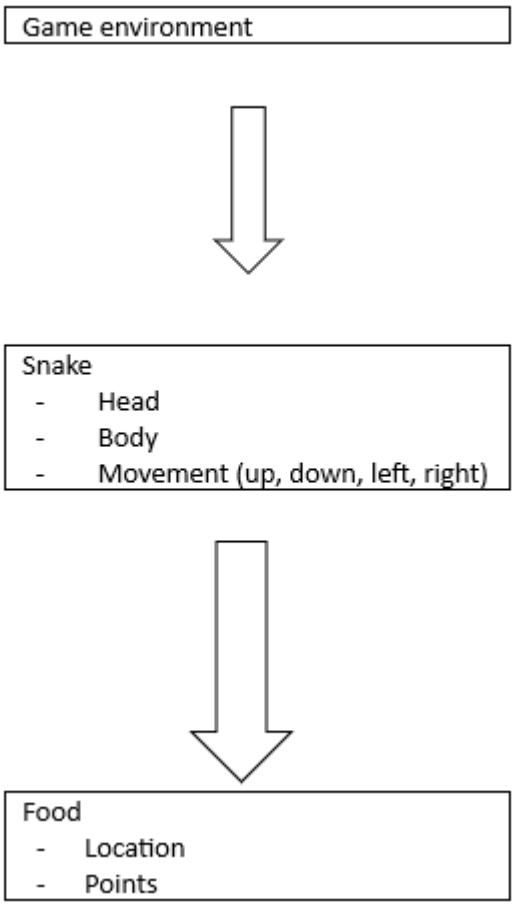
Out of Scope

- Multiplayer functionality: It might be beyond the scope of a basic game to include a multiplayer option where players can compete against one another.
- Advanced graphics: For a simple game, creating advanced graphics or animations might not be necessary because it would take more time and resources.
- Customizable game settings: Customizable game options like altering the snake's size or pace might not be appropriate for a simple version of the game.
- Localization: If the game is designed for a particular audience or region, adapting it to multiple languages can be outside the scope of the project.
- In-game purchases or ads: Implementing in-game purchases or ads may be out of scope if the game is intended to be a simple and free experience.

2. Product/Service Description

2.1 Product Context

The snake game is a standalone, self-contained game that ordinarily has no interaction with other games or associated platforms. The game is intended to be played independently on a variety of platforms and gadgets without the aid of external interfaces or connections.
While the snake game may resemble other games, including retro arcade favorites like Pac-Man or cutting-edge mobile games like Candy Crush, it differs in terms of gameplay mechanics and goals.
Here is a diagram that shows the major components of the snake game:



The area of the game where the snake and food items are located is called the playing field. The snake item may move in many directions and is made up of head and body segments. The food item is generated at random on the playing surface and awards points to the snake when it consumes it.
Since the snake game is meant to be a straightforward and self-contained gaming experience, it typically lacks significant external interfaces or connections.

2.2 User Characteristics

The snake game will have three types of users that will be differentiated by their roles:

- Student
- Faculty/Staff
- Other

Student

Age: 10-18 years old

Experience: Casual gamer, may have played similar games before

Technical Expertise: Basic familiarity with gaming and gaming equipment

Characteristics: May be looking for a simple and enjoyable game to play in their free time or to take a break from their academic work. Possibly enjoys playing against friends.

Faculty/Staff

Age: 30-60 years old

Experience: Casual gamer, may not have played games before

Technical Expertise: Basic familiarity with gaming and gaming equipment

Characteristics: May be looking for a fun and engaging game to play during breaks or downtime. May appreciate games that don't take a lot of practice and don't demand a lot of time.

Other

Age: Any age

Experience: Casual gamer, may have played similar games before

Technical Expertise: Basic familiarity with gaming and gaming equipment

Characteristics: May be looking for a simple and fun game to play during your free time as a way to relax. May enjoy games that don't have a steep learning curve or time investment.

2.3 Assumptions

- Availability of compatible devices: It is anticipated that the game will be played on equipment that meets all of the system requirements, including those screen size, processor speed, and graphics.
- User expertise: Users with a basic understanding of video games and gaming systems are regarded as being able to play the game. Users with more sophisticated technological skills could have different needs or expectations from others for the game.
- Availability of internet connectivity: It is anticipated that the game can be enjoyed without a reliable internet connection. However, some features, like multiplayer modes or online leaderboards, might need an internet connection.
- Operating system compatibility: It is expected that the game works with particular operating systems, such as Windows, macOS, iOS, or Android. The specifications might need to be adjusted if the game is incompatible with a certain operating system.
- Availability of necessary software: It is believed that the game will be played on hardware that has the required software, such as game engines or web browsers. The specifications might need to be adjusted if the required software is not available.

2.4 Constraints

- Parallel operation with an old system: The design possibilities may be limited by compatibility difficulties or the requirement to integrate with outdated technology if the snake game needs to run in parallel with an older system.
- Audit functions: The design possibilities may be limited if the snake game has audit features, such as an audit trail or log files, due to the requirement to capture and retain pertinent data regarding user activities, system events, or data changes.
- Access, management, and security: The design possibilities may be limited by the requirement to integrate safe authentication and authorization systems, data encryption, or other security measures if the snake game requires access control, user management, or security features.
- Criticality of the application: The design options may be limited if the snake game is essential for a specific business or user function due to the requirement for high availability, fault tolerance, or disaster recovery capabilities.
- System resource constraints: The design choices may be limited by the requirement to maximize performance and reduce resource utilization if the snake game has disk space or other resource restrictions.
- Other design constraints: The snake game can have to adhere to design guidelines or other restrictions, such as architecture, programming language, or framework selections. The design possibilities accessible to developers may be impacted by these limitations.

2.5 Dependencies

- Platform dependencies: Depending on the platform or operating system—such as Windows, macOS, iOS, or Android—that the snake game will be running on, the system requirements may change.
- Framework dependencies: The specifications for the snake game may vary depending on the framework or coding language chosen for development, such as Unity, Unreal Engine, or HTML5.
- Resource dependencies: Depending on the availability of resources like disk space, memory, or network bandwidth, the snake game's system requirements may change.
- Third-party dependencies: The specifications for the snake game might be reliant on third-party applications or services, including a game engine, a cloud storage service, or a payment processing system.
- Time dependencies: Depending on the amount of time available to create and deploy the game, such as a deadline or launch date, the requirements for the snake game may change.
- Design dependencies: The specifications for the snake game may be influenced by how other parts or systems, including user interfaces, game mechanics, or graphics, are created.

3. Requirements

1. **User Interface Requirements**
    - The user interfaces should be appealing visually and simple to navigate. (Priority 1)
    - The user interface should include options for starting, pausing and resetting the game. (Priority 1)
    - The current and final scores should be displayed on the screen. (Priority 2)
    - The game should provide visual feedback for eating the food or hitting the walls. (Priority 2)
    - Sound effects should be included for various actions. (Priority 3)
2. **Gameplay Requirements**
    - The game should follow the rules of the classic snake game. (Priority 1)
    - The movement of the snake should be controlled by the arrow keys. (Priority 1)
    - The snake should increase in size when it eats food. (Priority 1)
    - The game should end when the snake hits the wall of touches itself. (Priority 1)
    - The game should become increasingly more difficult as the snake eats more and the player goes through the levels. (Priority 2)
    - Levels should become faster as the player progresses. (Priority 2)
    - The game could provide and support a multiplayer mode. (Priority 3)
3. **Score Tracking Requirements**
    - The game should track the current score of the player during gameplay. (Priority 1)
    - The game should display the score on the screen. (Priority 1)
    - The game should also display the final score of the player when the game is over. (Priority 2)
4. **Error Handling Requirements**
    - The game should handle invalid input errors and provide the appropriate feedback. (Priority 1)
    - The game should handle memory errors to avoid bugs and potential game crashes. (Priority 2)
5. **Testing Requirements**
    - The game should undergo a testing process to ensure that it functions properly. (Priority 1)
    - The game should be tested for different scenarios, including boundary and edge cases. (Priority 2)
6. **Documentation Requirements**
    - The source code should be associated with comments and explanations about the logic of the game. (Priority 2)
    - A project report should be prepared, which contains information about the development process and the issues faced during this process. (Priority 2)

Snake Game Requirements Specification

3.1 Functional Requirements

FR_## - Functional Requirements number

Req#	Requirement	Comments	Priority	Date Rvwd	SME Reviewed/ Approved
FR_01	User-friendly Interface	The game should be visually appealing and simple to navigate.	1	07/05/2023	Engjëll Abazaj
FR_02	Buttons	The game should display visible buttons for starting, pausing and resetting the game.	1	07/05/2023	Engjëll Abazaj
FR_03	Control Keys	The player should be able to control the movements of the snake using the arrow keys.	1	07/05/2023	Engjëll Abazaj
FR_04	Gameplay	The snake should grow longer when it eats food and move faster as the game progresses.	1	07/05/2023	Engjëll Abazaj
FR_05	Game Over	The game should end when the snake touches a wall or itself.	1	07/05/2023	Engjëll Abazaj
FR_06	Score tracking and display	The game should keep track of the players points and display them on the screen.	1	07/05/2023	Engjëll Abazaj
FR_07	Testing	The game should run through multiple tests and cases to ensure that it works properly.	1	07/05/2023	Engjëll Abazaj
FR_08	Levels	The game should have multiple levels which increase in difficulty as the game progresses.	2	07/05/2023	Engjëll Abazaj
FR_09	Sound effects	The game should have sound effects for eating food, touching walls and other actions in the game.	2	07/05/2023	Engjëll Abazaj

FR_10	Final Score	The game should display the final score on the screen and update it whenever a new value is obtained.	3	07/05/2023	Engjëll Abazaj
FR_11	Multiplayer Mode	The game could have a multiplayer mode for different players to play together in the same setting (this could be a feature in an updated version of the game).	3	07/05/2023	Engjëll Abazaj

3.2 Non-Functional Requirements

3.2.1 User Interface Requirements

Required Screen Formats/Organization

The game should have a main menu screen containing the new game button and the exit option. During the gameplay the screen should display the environment surrounded by walls, the snake, the food and the player's current score. When the game ends a Game Over message should be displayed together with the final score.

Menu Structures

The main menu should contain understandable and intuitive options, such as "New Game", "Exit" and a button that explains the rules of the game. The settings menu can contain some additional features, such as adjusting the sound level or the brightness of the game or even customizing the snake and the environment, but these are a bit advanced so they can be implemented on a future release of the game.

Error and Other Messages

The game should be able to display error messages in case some problem occurs and suggest ways to fix those problems. For example, the player might be trying to move the snake using AWSD keys, but these inputs aren't supported. Other messages can include an indicator that tells the user when the game is over and displays the final score on the screen.

Function Keys

The function keys can be used as short cuts in the game. For example, F1 can be used to pause and resume, F2 can be used to mute the music, F3 can be used to go back to the main menu and so on.

Input

The player should be able to control the snake's movements using the arrow keys.

Progress Indicators

The score should be displayed on screen and the sound effects should be used to inform the player that the snake has successfully eaten a fruit.

3.2.2 Usability

Learnability

- The user documentation and help resources should be comprehensive and accessible.
- The help system should be adaptive to the situation of the user, being able to display the proper message at the right time.
- The system should be easy to learn and consistent in its features so that the user can adapt really quickly.
- There should be a clear set of instructions teaching the user how to play the game and what keys to use to perform certain actions.
- The game should be visually appealing as well and intuitive to enhance the experience of the user.

3.2.3 Performance

Static Numerical Requirements

The system should support at least 10 terminals concurrently, it should support a minimum of 2 users if multiplayer mode will be a feature and it should be able to handle up to 30 GB of game data. The system should support the processing of various types of information, including text, images and sound files.

Dynamic Numerical Requirements

The system should be able to handle a minimum of 100 transactions per minute under normal workload conditions. It should be able to handle a minimum of 50 tasks per minute and it should be able to process 10 gigabytes of data within a 5-minute time period under normal workload conditions. The system should be able to handle peak workload conditions, where it may need to process 200 transactions per minute, 100 tasks per minute, and 20 gigabytes of data within a 5-minute time period.

3.2.4 Manageability/Maintainability

3.2.1.1 Monitoring

The system should have a logging mechanism to track and record important events, errors, and exceptions and it should include error detection mechanisms to identify and report any failures or issues. The system should also have a monitoring system in place to regularly check the health and performance of the system. This monitoring system should be able to generate alerts or notifications in case of critical failures or abnormal conditions. The system should provide real-time metrics and statistics to monitor the system's performance and resource usage as well.

3.2.1.2 Maintenance

The system should be designed with modularity in mind, allowing easy maintenance and updates, it should have clear and well-documented code to facilitate future maintenance tasks and it should follow best practices and coding standards to ensure maintainability. The system should include error handling and exception

management to make troubleshooting and maintenance easier. It should also be designed to minimize dependencies and coupling between components to simplify maintenance.

3.2.1.3 Operations

The system should be easy to deploy and operate, with clear instructions and documentation for installation and configuration. The system should also provide a user-friendly interface or command-line interface for interaction and control. It should handle common operational tasks, such as backups, data management, and system configuration. The system should have appropriate security measures in place to protect against unauthorized access and data breaches and it should also be scalable and able to handle increasing loads or changes in requirements without significant disruptions.

3.2.5 Security

3.2.5.1 Protection:

To protect the Snake game system from malicious or accidental access, modification, disclosure, destruction, or misuse, the following factors could be considered:

●Access controls: Implement access controls to restrict access to sensitive data and system resources to authorized personnel only.

●Encryption: Encrypt sensitive data such as passwords, user data, and game settings to prevent unauthorized access or disclosure.

●Error handling and exception handling: Proper error handling and exception handling should be implemented to avoid system crashes or data corruption due to invalid input or unexpected errors.

●Data backups: Regular backups of game data should be taken to avoid data loss due to system failure or other unexpected events.

●Antivirus and malware protection: Install an antivirus and malware protection software to protect the system from viruses and other malicious software.

3.2.5.2 Authorization and Authentication:

To ensure proper Authorization and Authentication, the following factors could be considered:

●User authentication: Implement a secure user authentication mechanism such as a username and password combination or multi-factor authentication to prevent unauthorized access.

●Role-based access control: Implement a role-based access control mechanism to restrict access to sensitive data and system resources based on user roles and privileges.

●Session management: Implement proper session management to prevent session hijacking and ensure that user sessions expire after a certain amount of time.

3.2.6 Standards Compliance:

To ensure compliance with existing standards, policies, regulations, or laws, the following requirements could be considered:

● Data privacy: Ensure compliance with data privacy regulations such as the General Data Protection Regulation (GDPR) and ensure that user data is collected and processed in accordance with these regulations.

●Accessibility: Ensure that the game is accessible to users with disabilities and complies with accessibility standards such as the Web Content Accessibility Guidelines (WCAG).

● Intellectual property rights: Ensure that the game does not infringe on any intellectual property rights such as copyrights or trademarks. Licensing: Ensure that the game complies with any licensing requirements for third-party software or libraries used in the development of the game.

3.2.7 Other Non-Functional Requirements:

Other non-functional requirements that could be considered include:

●Performance: The game should perform well and not have any significant lag or delays, even when the player's score is high.

●Usability: The game should be easy to use and understand, with clear instructions and intuitive controls.

● Scalability: The game should be able to handle an increasing number of users without any significant decrease in performance.

● Reliability: The game should be reliable and not crash or freeze during gameplay. Maintainability: The game should be designed and developed in a way that makes it easy to maintain and update in the future.

3.3 Domain Requirements

Game mechanics: The game should follow the basic rules of the classic snake game, including the movement of the snake, the appearance of food, and the game over conditions.

Graphics and user interface: The game should have simple graphics that are easy to understand and use, and a user interface that allows the player to start and restart the game, control the snake, and see their score.

Input and output: The game should accept input from the player's keyboard to control the snake's movement and display output to show the game board, score, and game over messages. Performance: The game should run smoothly and without lag, even as the snake grows longer and the game board becomes more crowded.

Compatibility: The game should be compatible with different operating systems, screen sizes, and Python versions.

Code structure and organization: The game code should be well-organized and easy to read and modify, with clear and concise comments and documentation.

Testing and debugging: The game should be thoroughly tested to ensure that it works as expected, and any bugs or errors should be identified and fixed in a timely manner

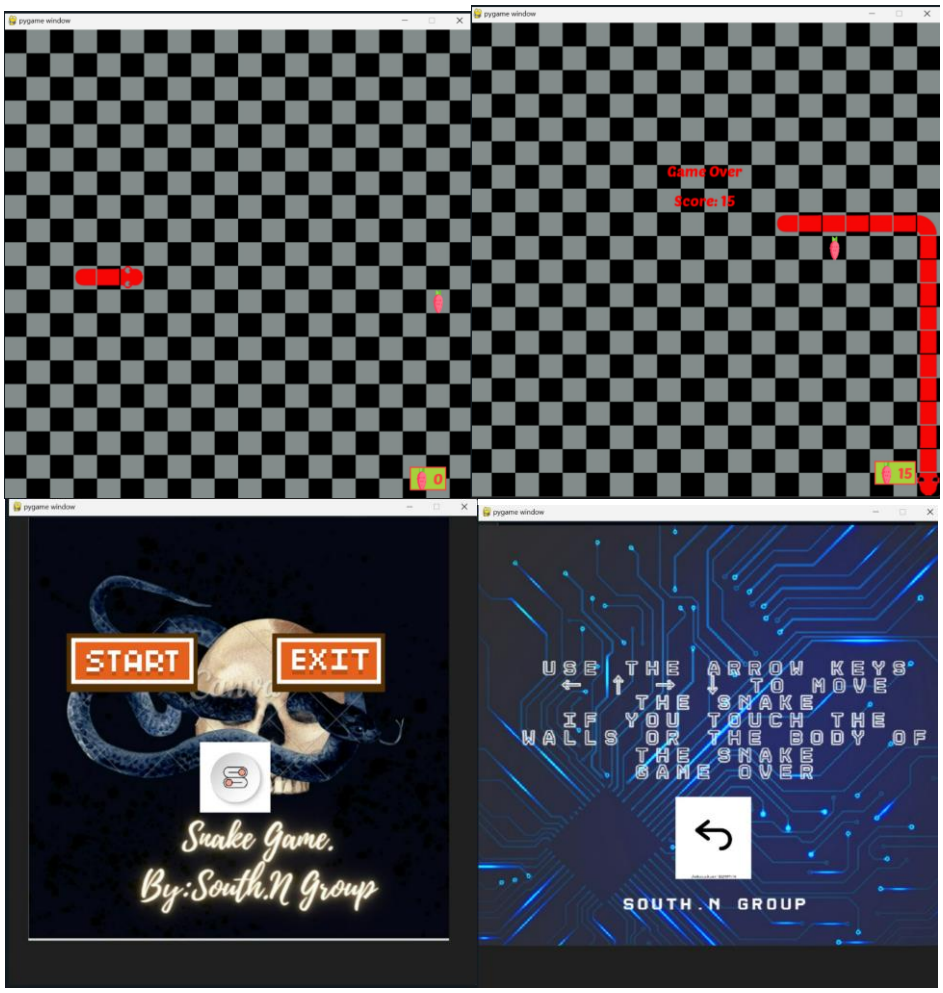
4. Design thinking methodologies

4.1 Negotiation involves working collaboratively with stakeholders to balance competing interests and achieve a common goal. In the context of a snake game, negotiation can involve working with developers, testers, and other team members to ensure that the game is both functional and enjoyable to play. This may involve making trade-offs between different features or design elements to achieve the best possible user experience

4.2 Empathy involves putting oneself in the user's shoes to understand their needs, goals, and challenges. In the context of a snake game, empathy can involve understanding the user's preferences in terms of the game's difficulty level, controls, and visual design. By empathizing with the user, the designer can develop a game that is both engaging and satisfying to play.

4.3 Noticing involves paying close attention to details and patterns in the user's behavior, environment, and interactions with the game. In a snake game, noticing can involve observing the user's gameplay to identify areas where the game can be improved. For example, if the user is struggling with a particular level or feature, the designer can identify the problem and make changes to improve the user experience.

4.4 GUI (Screenshots)



5. Software Design

5.1 Use Case

The Snake Game's Major Functions, in Brief:

1. Game Setup: Create a grid or game board to set up the game's atmosphere.

- Establish the snake's and the food's beginning positions.
- Assign the snake a starting direction.

2. Keyboard inputs should be available for the player to use to move the snake.
- Adjust the snake's location in accordance with the selected direction.
- Scan for impacts with the game's limits, the snake's body, or objects like food.

3. Random Food Generation: Produce food at random on the game board.
- Make sure the food doesn't touch the snake's body or any already-existing food.

4. Progress and Scoring:
Keep score of how many food items the player has acquired, and display that score.
When the snake consumes food, the score rises.
As the player advances, the game's level or difficulty may change.

5. Game Over Requirements:
Determine when the snake slams into itself or the game's borders.
The game will end, and the results will be shown.
Give the player the option to restart the game.
interface for users:

6. Make the game UI pleasant to the eye.
Show the snake, food, and score on the game board.
Implement interactive game-start, pause, and restart buttons or controls.

Business scenario and use case:

Business Need: To draw customers and raise their user engagement numbers, a gaming business needs to create a straightforward and entertaining Snake game.

Problem Proposition:

Lack of Interactive Entertainment: Snake, a timeless game with straightforward but addicting gameplay, is absent from the company's current gaming catalog.
User Engagement Metrics: The business needs to improve its user engagement metrics, which include session length and retention rate.
Expanding Game Portfolio: The corporation wants to expand its game selection to appeal to more people and draw in more players.
Technical and Business Environment:

The company's user base is well-established and it operates in the gaming sector.
Technical Environment: A suitable programming language, game development framework, and graphic assets will be used to create the game.
The desired goals are:

Create a fun Snake game with a straightforward but addictive gaming experience.
the retention rate and session duration parameters for user engagement should be raised.

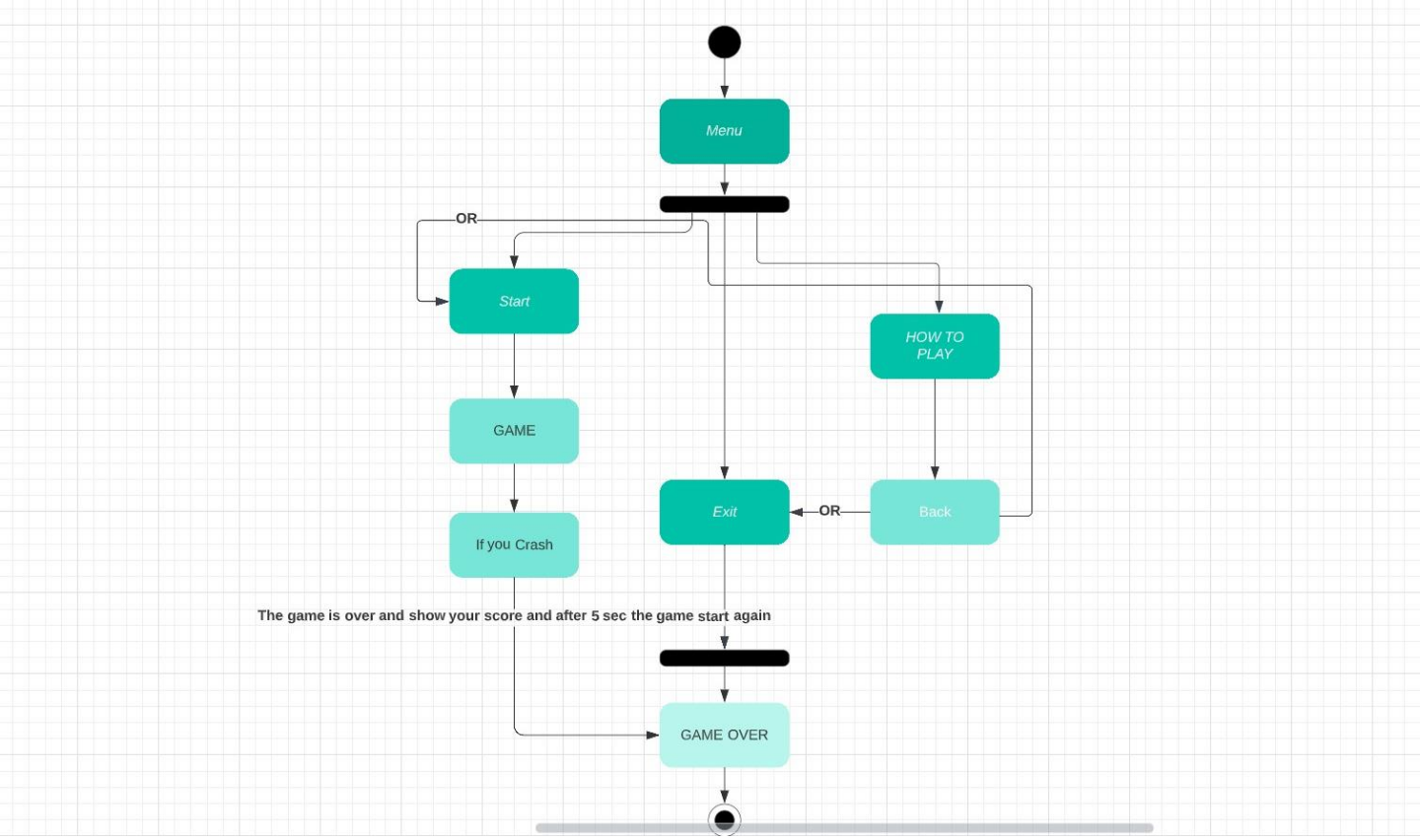
The Business Model and Actors:

Players: Users who will use the game's interface to play the Snake game.
The group in charge of planning, creating, and testing the Snake video game.

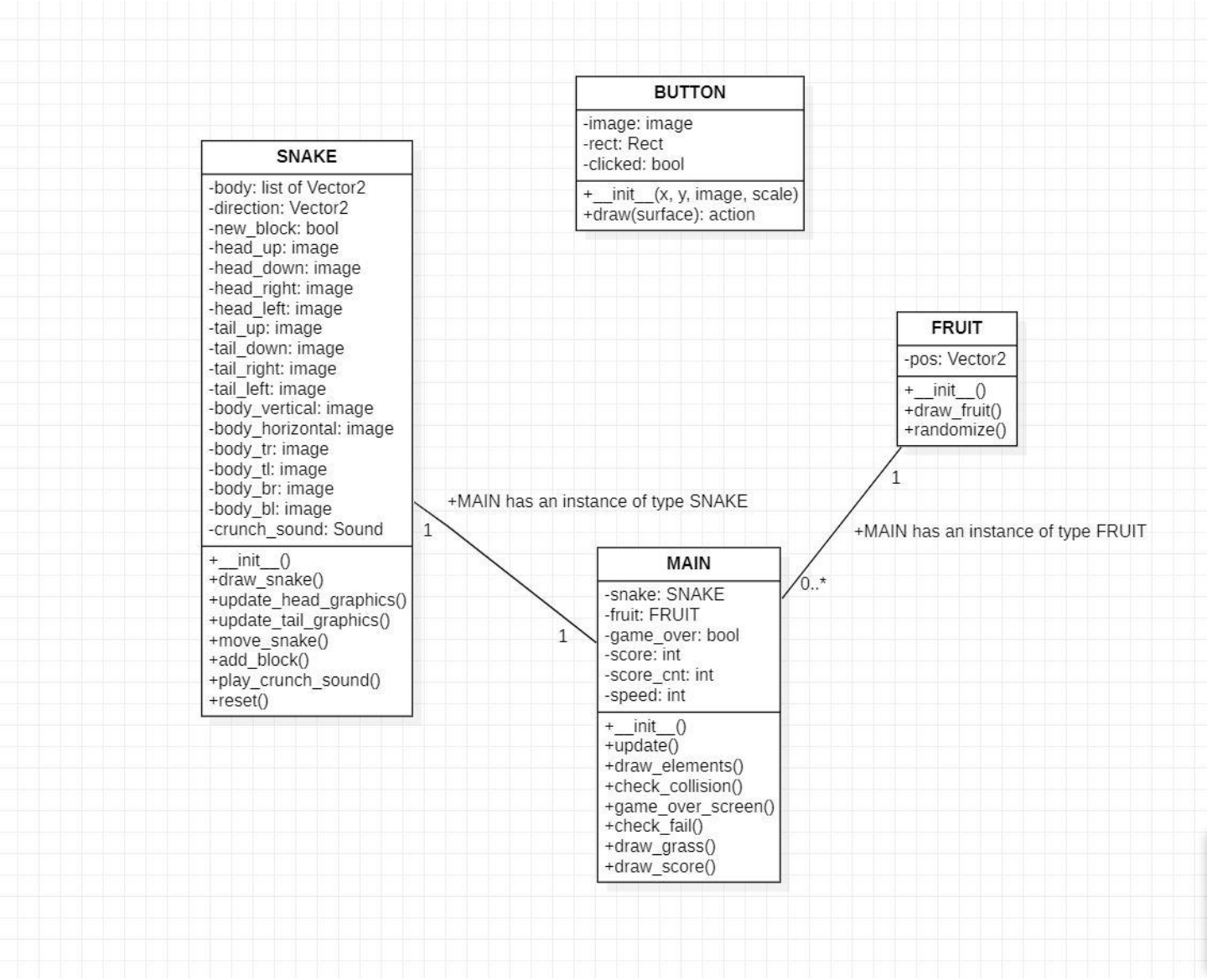
The marketing team is in charge of publicizing the game and drawing in new players.
User engagement indicators are measured and analyzed by the analytics team.
Measures of Success

Increased Session Length: Aim for at least 10-minute sessions on average.
Better Retention Rate: Increase by 20% the proportion of participants who play for at least three sessions.
Positive customer feedback should be gathered via reviews or surveys, with an aim for an average rating of at least 4 out of 5.

5.2 State Diagram



5.3 Class Diagram



Appendix A. Definitions, Acronyms and Abbreviations

- FR: Functional Requirement
  Definition: A functional requirement specifies a particular behavior or functionality that the snake game software must possess.
  Usage: Functional requirements are essential for describing the specific features and actions that the game should exhibit.

- GUI: Graphical User Interface
  Definition: The graphical user interface (GUI) refers to the visual elements and controls through which users interact with the snake game.
  Usage: The GUI provides the user with a means to control the game, view the game environment, and receive feedback on their actions.

Appendix B. References

This document references the lecture notes and PowerPoint presentations of the *Introduction to Software Engineering* Lab hours, as they were used as the source material to help the members of this group complete their respective sections of the document.

Appendix C. Organizing the Requirements

1.*Start-Up Mode Requirements*
1.1. System Initialization
1.2. Display Game Logo
1.3. Load Game Resources
1.4. Show Main Menu
1.5. Accept User Input for Game Options

2.*Gameplay Mode Requirements*
2.1. Display Game Grid and Snake
2.2. Generate Food Items
2.3. Control Snake Movement
2.4. Handle Snake Collisions with Walls
2.5. Handle Snake Collisions with Itself
2.6. Detect Snake's Consumption of Food Items
2.7. Update Score
2.8. Display Game Over Screen on Snake's Death
2.9. Allow Restart or Quit Options

3.*Pause Mode Requirements*
3.1. Pause Gameplay
3.2. Display Pause Menu
3.3. Accept User Input for Pause Options
3.4. Resume Gameplay

4.*High Score Mode Requirements*
4.1. Store and Retrieve High Scores
4.2. Display High Score Table
4.3. Allow Resetting of High Scores

5.*Settings Mode Requirements*
5.1. Display Settings Menu
5.2. Allow Modification of Game Options
5.3. Save Settings

6.*Multiplayer Mode Requirements*
6.1. Establish Network Connection
6.2. Enable Multiplayer Functionality
6.3. Handle Multiplayer Interactions
6.4. Display Multiplayer Scoreboard

7.*Graphics Mode Requirements*
7.1. Implement Advanced Graphics
7.2. Use Animation Effects
7.3. Provide Visual Feedback and Indicators

8.*Customization Mode Requirements*
8.1. Allow Customization of Game Elements
8.2. Provide Customizable Themes
8.3. Enable User-Defined Settings

9.*Technical Requirements*
9.1. Support for Multiple Platforms
9.2. Compatibility with Operating Systems
9.3. Programming Language and Frameworks
9.4. Memory and Performance Optimization

***Note:** Please note that some of these requirements have not been implemented into the first version of the game yet, but they could be implemented in future releases.*