1. Introduction

1.1 Purpose

The purpose of this document is to give a detailed description of the requirements for the Asteroid. It will illustrate the purpose and complete declaration for the development of the system. It will also explain system constraints, interface and interactions with other external applications. This document is primarily intended to be proposed to a customer for its approval and a reference for developing the first version of the system for the development team.

1.2 Document Conventions

Main Sections Titles:
Font: Calibri
Face: Bold
Size: 18

Subsection Titles:
Font: Calibri
Face: Bold
Size: 14

Other Text Explanations:
Font: Calibri
Face: Normal
Size: 11

1.3 Intended Audience and Reading Suggestions

This Software Requirements document is intended for:

- **Developers** who can review project's capabilities and more easily understand where their efforts should be targeted to improve or add more features to it (design and code the application – it sets the guidelines for future development).
- **Project testers** can use this document as a base for their testing strategy as some bugs are easier to find using a requirements document. This way testing becomes more methodically organized.
- **End users** of this application who wish to read about what this project can do.

1.4 Product Scope

Asteroid is a game application which provides end users an interface to play a space based game.

It will be a low bandwidth game that will be fun and intuitive, with a lot of visual cues.

We hope to foster a competitive gaming environment.

2. Overall Description

2.1 Product Perspective

Everybody likes to play games. The game we design aims to be very light-weight and intuitive. There will be no memory leaks, or CPU intensive tasks.

It is a space environment with freely moving meteors. The user controls a rocket and shoots the meteors to break them up. On destroying all the meteors, the user proceeds to the next level. The gamer is provided with 3 lives at the beginning of the game.

At the end of the game the user enters his name and has the option to share his score with friends.

2.2 Product Functions

- **Meteor movement**
  - Meteor.update : updates meteor coordinates
  - Meteor.draw : draws meteor on canvas
- **Rocket movement**
  - Rocket.update : updates rocket coordinates
  - Rocket.draw : draws rocket on canvas
- **Bullet movement**
  - Bullet.update : updates bullet coordinates
  - Bullet.draw : draws bullet on canvas
- **Wrap around :** Wraps around meteors and rocket

```
if (this.x > global.width) {
   this.x = 0;
} else if (this.x < 0) {
   this.x = global.width;
} else if (this.y > global.height) {
   this.y = 0;
} else if (this.y < 0) {
   this.y = global.height;
};
```

- **Collision detection :**
  - Rocket & Meteors
  - Bullet & Meteors
  - Meteors
- **Game audio:** Audio for the game (crash, explosions)
- **Lives**
- **Game Levels :** Various difficulty levels as the game progresses
- **User Scoreboard :** List of all scores

2.3 System interfaces

Asteroid integrates two internal systems to provide functionality:

- **Client** The game has an interface to the user's client to receive user input and moves selections for the game.

- **Network** The game has an interface to the network in order to transmit information and connect players.

2.4 Operating Environment

The game needs an Internet browser that supports HTML5 and JavaScript to run, preferably Google Chrome.

3. External Interface Requirements

3.1 User Characteristics

The user of Asteroid needs experience and must be able to play the game at least a basic level. Furthermore, users need to be very familiar with the rules of the game.

3.2 Hardware Interfaces

Asteroid runs on any computer hardware meeting the following criteria:
- Capable to use an Internet connection
- Includes memory storage
- Includes a mouse
- Includes a keyboard

3.3 Software Interfaces

Requires an HTML5 and JavaScript enabled web browser.

4. Other Nonfunctional Requirements

4.1 Performance Requirements

The game is expected not to hang nor lag. The collision detection should be immediate.

4.2 Software Quality Attributes

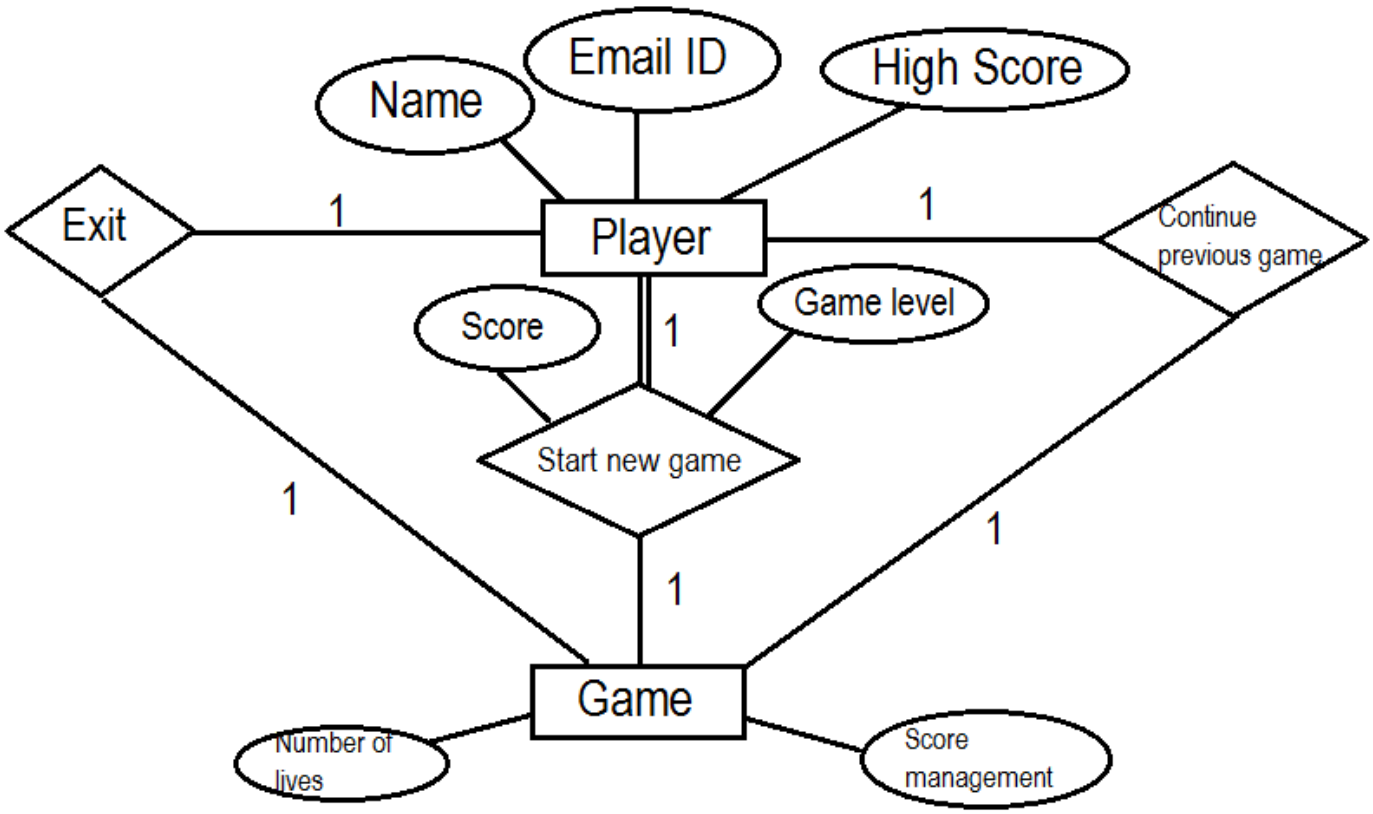
The game should be

- **Correct:** It should 'just work'.
- **Flexible:** The user should be able to move, shoot, and carry out operations however he wants to. There will be no fixed movements.
- **Maintainable:** The code should be clean, modular, and adhere to standards.
- **Portable:** The game engine should be portable to any framework.
- **Reliable:** The physics behind the game must be reliable and work predictably. There should be no undefined results.
- **Testable:** The game engine should be easy to test; the code should provide for easy testing, deleting, etc.

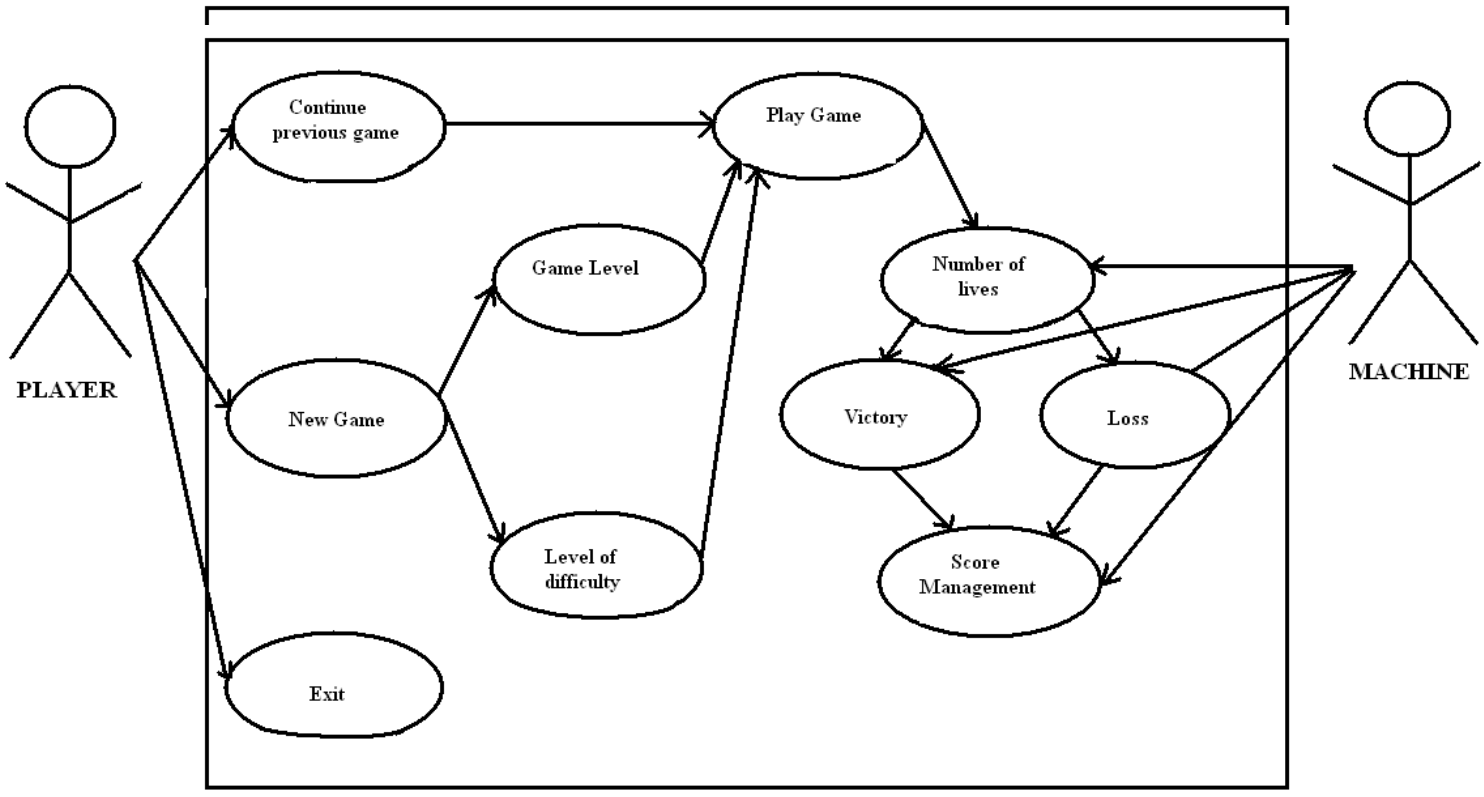
5. Technologies Used:

- HTML5
- CSS3
- Javascript (ES6)
- Sinatra
- PostgreSQL
- Heroku
- Gimp/Photoshop

6. Entity-Relationship Diagram:



7. Use Case Diagram:



8. Sequence Diagram: