

# Project Report: Neon Zombie Survival

Development Team

December 4, 2025

## Contents

<b>1 Project Statement</b>	<b>2</b>
1.1 Overview . . . . .	2
1.2 Objective . . . . .	2
<b>2 Tools and Technologies</b>	<b>2</b>
2.1 Core Technologies . . . . .	2
2.2 Development Environment . . . . .	2
<b>3 Project Details and Architecture</b>	<b>2</b>
3.1 Game Mechanics . . . . .	2
3.2 Controls and Input Handling . . . . .	3
3.3 Visual Style . . . . .	3
<b>4 Conclusion</b>	<b>3</b>

# 1 Project Statement

## 1.1 Overview

**Neon Zombie Survival** is a browser-based, 2D top-down shooter game designed to test player reflexes and strategy. The objective is simple yet challenging: survive as long as possible against infinite, progressively difficult waves of zombies. The game embraces a retro "cyberpunk" aesthetic, utilizing high-contrast neon colors against a dark background to create an immersive atmosphere without relying on heavy external assets.

## 1.2 Objective

The primary goal of this project was to demonstrate the capabilities of modern web technologies (HTML5 Canvas) by building a performant, cross-platform game engine from scratch. The project aims to provide a seamless gaming experience on both desktop computers and mobile devices without requiring installation or plugins.

# 2 Tools and Technologies

The project is built using a lightweight, "vanilla" technology stack to ensure maximum compatibility and performance.

## 2.1 Core Technologies

- **HTML5:** Provides the structural backbone of the application. The `<canvas>` element is utilized as the primary rendering target for all game graphics.
- **CSS3:** Handles the styling of the User Interface (UI) layers, including the Heads-Up Display (HUD), start screens, and menus. It ensures the game is responsive and visually consistent across different screen sizes.
- **JavaScript (ES6+):** The core logic of the game is written in pure JavaScript. It manages the game loop, entity updates, collision detection, and input handling.

## 2.2 Development Environment

- **Rendering Engine:** HTML5 Canvas 2D API (Native).
- **No External Frameworks:** To maintain a small footprint and ensure deep understanding of game mechanics, no third-party game engines (like Unity or Phaser) were used.
- **Version Control:** Git (implied standard practice).

# 3 Project Details and Architecture

## 3.1 Game Mechanics

- **Infinite Wave System:** Zombies spawn at random intervals around the screen edges. As the player's score increases, the spawn rate and zombie speed increase, creating a progressive difficulty curve.
- **Entity Management:** The game utilizes Object-Oriented Programming (OOP) principles. distinct classes exist for the Player, Zombie, Bullet, and Particle entities.

- **Collision Detection:** A custom physics implementation calculates distance-based collisions between circular entities (Circle-Circle collision) to handle interactions between bullets, zombies, and the player.
- **Particle System:** A dynamic particle system renders visual feedback for events such as explosions, zombie deaths, and gunfire, enhancing the "game feel."

### 3.2 Controls and Input Handling

The game features a dual-control scheme to support multiple platforms:

- **Desktop:**
  - **Movement:** W, A, S, D keys or Arrow keys.
  - **Aiming:** Mouse cursor position.
  - **Shooting:** Left Mouse Button.
- **Mobile (Touch):**
  - **Virtual Joystick (Left):** Controls player movement.
  - **Virtual Joystick (Right):** Controls aiming and auto-fires when engaged.

### 3.3 Visual Style

The visual style is procedurally generated using code. Instead of loading static image files (PNG/JPG), the game draws shapes (arcs, rectangles) directly onto the canvas. This allows for:

- **Scalability:** Graphics remain crisp at any resolution.
- **Performance:** Zero network latency for asset loading.
- **Aesthetic:** A consistent "Neon" look with glowing effects achieved via shadow blurring.

## 4 Conclusion

The **Neon Zombie Survival** project successfully demonstrates that engaging, high-performance games can be built using standard web technologies. By eschewing heavy game engines, the project remains lightweight (contained in a single file) and highly accessible.

The implementation of a custom game loop, physics engine, and adaptive input system highlights the versatility of JavaScript and the HTML5 Canvas API. The final product is a fun, responsive arcade shooter that runs smoothly on a wide range of devices.