SNIPE AND STRIKE

A PROJECT REPORT
for
MINI PROJECT - II (ID201B)
Session (2024-25)

Submitted by

Ankit Kumar Shahi
(202410116100028)
Ansh Mishra
(202410116100030)
Ankit Sisodia
(202410116100029)

Submitted in partial fulfilment of the
Requirements for the Degree of

MASTER OF COMPUTER APPLICATION

Under the Supervision of
Dr. Vipin Kumar
Associate Professor

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206
(MAY- 2025)

CERTIFICATE

Certified that **Ankit Kumar Shahi (202410116100028), Ansh Mishra(202410116100030)**, **Ankit Sisodia (202410116100029)** has carried out the project work having "**Snipe & Strike"** (MINI PROJECT - 2 (FULL STACK DEVELOPMENT) ID201B) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Dr. Vipin Kumar
Associate Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Dr. Akash Rajak
Dean
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

ABSTRACT

The **Snipe and Strike** project is a web-based arcade-style shooting game designed to provide an engaging, interactive experience where users control a UFO and must destroy enemy UFOs approaching from the front. With smooth controls, real-time gameplay mechanics, and a dynamic interface, the game challenges players to test their reflexes and precision while competing for high scores. The motivation behind building this project stems from the desire to blend entertainment with core programming concepts, especially in frontend development and interactive design. Many existing browser-based games are either overly simplistic or require extensive resources to run. **Snipe and Strike** addresses this by delivering a lightweight, fully functional game that runs seamlessly in the browser without additional installations. The system is developed using **HTML, CSS, and JavaScript** for the frontend, while the backend functionalities such as leaderboard storage and score submission are powered by **Python Flask APIs**. The game is divided into modular components including the UI engine, game loop, collision detection system, and backend integration. Key features include real-time enemy spawning, shooting mechanics, score tracking, and dynamic difficulty adjustment. This project not only demonstrates the practical implementation of real-time game logic and web technologies but also opens up possibilities for further improvements such as multiplayer support, power-ups, and advanced scoring algorithms. By merging fun with functionality, **Snipe and Strike** highlights the potential of web technologies in developing interactive entertainment experiences.

Keywords: Snipe and Strike, Web Game Development, Real-Time Gameplay, JavaScript Game, Frontend Project, Flask Integration, Collision Detection, Score Tracking.

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Dr. Vipin Kumar,** for his guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to Dr. Akash Rajak, Professor and Dean, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Ankit Kumar Shahi

Ansh Mishra

Ankit Sisodia

TABLE OF CONTENTS

TABLE OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Overview

The project titled "Snipe and Strike" is an engaging and interactive 2D web-based asteroid shooting game where the player controls a UFO and shoots down enemy UFOs approaching from the top of the screen. The main objective is to destroy these enemies before they cross the screen. If even one enemy manages to pass, the game ends, emphasizing quick reflexes and accuracy. The game design combines both fun and challenge to keep the player engaged throughout the experience.

This game is built using a combination of HTML, CSS, JavaScript, and Python, and aims to offer an enjoyable yet challenging experience. The visual appeal is enhanced by dynamic animations and real-time interactions that make it feel like a modern version of classic arcade games. The game's lightweight nature ensures that it runs efficiently across different devices, including PCs, tablets, and smartphones. Furthermore, it is designed to be fast and compatible across various browsers, allowing users to enjoy seamless gameplay regardless of their preferred platform.

The user interface is intuitive, and no prior experience is needed to play the game, making it accessible to both new players and seasoned gamers. The game's simplicity allows anyone to jump in and enjoy, but its addictive nature ensures that players will keep returning to improve their scores and challenge their reflexes.

1.2 Problem Statement

The rapid growth of casual gaming demands interactive and engaging games that are simple yet addictive. As gaming becomes more mainstream, players increasingly seek experiences that are easy to pick up and play, especially when they're short on time. Many traditional web-based games lack dynamic interaction, real-time action, and engaging gameplay mechanics. This project addresses that gap by delivering a game that is not only visually appealing but also tests the player's reflexes, accuracy, and decision-making under pressure.

The gaming community's demand for instant accessibility and fast-paced action has led to the popularity of games that offer quick play sessions, and this is where "Snipe and Strike" excels. Unlike traditional games that may require long learning curves or installation of software, "Snipe and Strike" can be played directly in any modern web browser, making it an ideal solution for users seeking a quick gaming fix. The focus on reaction speed and precision adds a layer of challenge that will appeal to casual gamers, while also offering a sense of progression and accomplishment through the leaderboard and scoring system.

1.3 Objective

- **To develop a 2D shooter game using web technologies**: This objective focuses on utilizing HTML, CSS, JavaScript, and Python to bring the game to life, ensuring it operates efficiently in web browsers.

- **To create a simple yet addictive gameplay experience**: The game is designed to be easy to understand but challenging enough to keep players engaged. The quick-paced action and the need for precise shooting ensure that players will keep coming back to improve their performance.

- **To implement real-time collision detection and responsive controls**: One of the key elements of the game is the real-time shooting mechanism, which allows players to fire bullets and detect collisions with enemies instantly. This feature adds to the dynamic nature of the game and provides instant feedback to the player.

- **To build a scalable, browser-compatible game that runs efficiently**: The game is designed to be lightweight and optimized, ensuring it can run on various devices without any performance issues. The game should work seamlessly on any modern browser, ensuring wide accessibility.

- **To apply core frontend and backend concepts in a practical project**: Through the development of this game, the developer applied key web development principles, including frontend design with HTML, CSS, and JavaScript, along with optional backend integration through Python, to enhance the game's functionality.

1.4 Scope

This project demonstrates real-time game mechanics through web development tools. The game is scalable and can be extended with additional features such as multiple levels, power-ups, leaderboards, and multiplayer modes. The core game loop and mechanics are designed to be flexible enough to allow for future enhancements and optimizations.

The game's primary scope includes:

- **UFO-controlled player with shooting ability**: The player controls a UFO that can move horizontally across the screen, shooting bullets to destroy incoming enemies.
- **Enemy generation from the top**: Enemy UFOs appear from the top of the screen and move downwards, challenging the player to destroy them before they pass.
- **Game over on enemy pass**: If any enemy crosses the bottom of the screen without being destroyed, the game ends, adding a sense of urgency and importance to the player's actions.

- **Scoreboard and high score tracking**: The game tracks the player's score and high score, motivating players to beat their previous achievements and compete against others.

- **Difficulty modes**: The game can include different difficulty levels, such as Easy, Medium, and Hard, which affect enemy speed, frequency, and other gameplay elements.

1.5 Features

1. **Simple UI**: The game interface is designed to be user-friendly and easy to navigate. With its responsive design, the game adapts to different screen sizes and devices, ensuring a consistent experience whether played on a desktop, tablet, or mobile phone.

2. **Live Enemy Movement**: Enemies are dynamically generated and move from top to bottom at varying speeds. As the player progresses, the enemies may appear in larger numbers or at faster speeds, ramping up the difficulty level.

3. **Bullet Collision Detection**: The game includes a real-time shooting system, where bullets fired by the player instantly collide with enemies, causing them to disappear and updating the player's score.

4. **Game Over Logic**: A game over occurs when an enemy UFO crosses the bottom of the screen. This creates a sense of urgency, pushing the player to act quickly and think strategically.

5. **Score Tracking**: The player's score is continuously updated based on the number of enemies destroyed. A high score system adds competition, encouraging players to improve their performance.

6. **Level Selection**: (If applicable) The game can include multiple difficulty levels, allowing players to choose from Easy, Medium, or Hard, which affects the speed of enemies, the number of enemies, and the complexity of the gameplay.

1.6 Hardware/Software Used

Hardware Requirement

S.No	Hardware Requirement
1	PC/Laptop with 4 GB RAM
2	Core i3 processor or above
3	Minimum 1 GB storage space

Software Requirement

S.No	Software Requirement
1	OS: Windows 8 / 10 / 11
2	Frontend: HTML, CSS, JS
3	Backend: Python (optional)

S.No	Software Requirement
4	Code Editor: VS Code
5	Browser: Chrome/Edge/Firefox

1.7 Background

"Snipe and Strike" is built on the idea of bringing classic arcade shooting games to the web with modern styling. The inspiration behind the game comes from traditional arcade games like *Space Invaders* and *Galaga*, which were famous for their simplicity and addictiveness.

This project takes those familiar concepts and enhances them with modern web technologies and design principles. By using HTML, CSS, and JavaScript, the developer has built a game that runs efficiently in browsers while delivering an engaging experience.

The game also helped the developer explore key game development concepts such as collision detection, animation handling, and game loops. These concepts were crucial in making the game feel responsive and real-time. The integration of Python for optional backend functionalities, such as score tracking or player data management, added an additional layer of functionality and scalability to the game.

Throughout the development process, the game has evolved into a project that not only challenges the player's reflexes but also tests the developer’s skills in building interactive web-based applications. By creating this game, the developer has gained valuable insights into the world of browser-based game development, animation handling, and user interaction within the web environment.

CHAPTER 2

FEASIBLITY STUDY

2.1 Economical Feasibility

The development of "Snipe and Strike" proves to be highly economical, especially for an academic-level project undertaken by students. One of the biggest advantages is the use of **open-source** and **freely available technologies** such as HTML, CSS, JavaScript, and Python. These technologies are not only free to use but also extensively documented, making development easier and more accessible for beginners.

For the backend, the game utilizes **Flask**, a lightweight Python framework, which is also open-source and perfect for building small web applications and APIs. Tools used in development include **Visual Studio Code (VS Code)**, which is a free, powerful code editor that supports a variety of extensions for web development, and **Google Chrome or Firefox** for testing and running the game.

Hosting costs are negligible or even zero. The frontend of the game can be deployed using free platforms like **GitHub Pages**, which supports static websites. For backend services like storing high scores, platforms such as **Render**, **Replit**, or **Glitch** can be used without any subscription fees at the basic level.

Thus, the total investment made into the project remains minimal, and **all resources used fall well within the reach of an average college student**. No premium tools, licenses, or hardware upgrades were required. This makes "Snipe and Strike" an excellent example of a cost-effective software project that delivers solid functionality and a polished user experience.

2.2 Technical Feasibility

From a technical standpoint, "Snipe and Strike" is entirely feasible and well-aligned with the capabilities of modern web technologies. It is designed using **HTML**, **CSS**, and **JavaScript** for the frontend—technologies that are not only universally supported across browsers but also allow for real-time, interactive gameplay experiences. These tools offer excellent control over layout, animation, and user interaction, all of which are essential for creating an engaging shooter game.

The **backend** component, built using **Python with Flask**, is lightweight and perfect for the scope of this project. Flask provides an efficient way to create RESTful APIs which are used to store and retrieve high scores from a file or a small database. Since the game doesn't require user registration, authentication, or large-scale data handling, the backend remains simple, fast, and easy to maintain.

The **data storage** is also minimal. A simple **text file** or a **SQLite** database can be used to persist scores. This eliminates the need for complex database setups like MySQL or MongoDB, reducing both development and operational complexity.

Performance-wise, all **core game logic runs client-side**, within the browser, ensuring smooth gameplay even on lower-end systems. There's no reliance on heavy computation or complex libraries, keeping the resource consumption low. Furthermore, the game UI is **responsive** and adjusts to different screen sizes, making it accessible on desktops, tablets, and mobile devices.

In short, the technologies used are reliable, efficient, and appropriate for the project's scope, making the entire system technically sound and sustainable.

2.3 Operational Feasibility

"Snipe and Strike" is designed with a focus on **ease of use and seamless user experience**. The operational feasibility of this project is excellent, particularly because the game requires **no installation or setup**. The player simply opens the game in a web browser and starts playing instantly.

The game's controls are extremely intuitive—players only need to move the UFO left or right using arrow keys (or A/D keys) and shoot using a spacebar or a mouse click. This minimal learning curve ensures that **even first-time users can understand and enjoy the game within seconds**.

Another factor contributing to operational feasibility is the **automated score tracking system**. When a player finishes a game, their score can be automatically stored and retrieved without any manual input. This process is handled in the background using AJAX calls to a Flask-based API, ensuring a smooth flow of gameplay without interruptions.

The game interface has been designed with **simplicity and clarity in mind**, featuring a clean layout, readable fonts, and easily identifiable buttons. The game's responsiveness allows it to function effectively across multiple devices, increasing accessibility.

Operationally, the system is self-contained, requires no technical knowledge to run, and is robust enough to be used in demonstrations, exhibitions, or casual settings without the need for supervision or troubleshooting. This makes it ideal for classroom use or public display.

2.4 Behavioural Feasibility

The behavioural feasibility of “Snipe and Strike” is also high, considering it aligns well with the **patterns and expectations of modern casual gamers**. Most players today prefer games that are quick to load, easy to understand, but challenging enough to keep them engaged—and this project meets all of those criteria.

The game’s controls are minimalistic: move and shoot. This matches user behaviour in terms of simplicity and direct engagement. There is **no overwhelming UI**, no tutorials required, and no clutter. The player is drawn directly into the action, which caters to the short attention spans and fast-paced lifestyle of modern users.

The gameplay is **addictive and encourages replayability** by introducing a scoring system and increasing difficulty over time. The integration of a high score leaderboard taps into players' competitive nature, motivating them to play repeatedly to beat their own or others’ scores.

Another behavioral strength is the game’s **zero friction** model. There’s no need to sign in, register, or download any software. Players can simply click a link, play the game, and leave—all within a few minutes. This ease of access significantly increases the likelihood of users engaging with the game.

Since browser-based games are already popular among students and casual users, the chances of players enjoying and sharing this game with others are quite high. Its lightweight design and short gameplay sessions make it perfect for quick breaks or casual fun.

CHAPTER 3

SOFTWARE REQUIREMENT SPECIFICATION

3.1 Functionalities

"Snipe and Strike" is packed with several essential and advanced game functionalities that ensure smooth gameplay and an engaging user experience. The core features are:

- **Start Game Interface**
  The game begins with a start screen where players can choose between different difficulty levels—**Easy**, **Medium**, and **Hard**. This adds replayability and custom challenge levels for players of varying skill.

- **UFO Control System**
  The player controls a UFO spaceship that can move **left and right** using the arrow keys or A/D keys. The movement is smooth and responsive, giving the player full control over positioning during enemy attacks.

- **Shooting Mechanism**
  Pressing the **spacebar** triggers a bullet to fire upward from the player's UFO. This shooting mechanism allows the player to destroy enemy UFOs. Bullets travel in real-time and disappear upon impact or if they miss.

- **Enemy Spawning and Movement**
  Enemy UFOs continuously spawn from the top of the screen at regular intervals. Their movement speed depends on the chosen difficulty level. As the game progresses, enemies can spawn faster, increasing challenge.

- **Real-Time Collision Detection**
  The game uses **JavaScript-based collision detection** to determine when a bullet hits an enemy UFO. Upon detection, the enemy is destroyed, and an explosion animation or visual feedback is shown.

- **Score System**
  Players earn **points** for each enemy they destroy. The score is displayed live on the screen and encourages players to keep playing and improve their performance.

- **Health System**
  Each time an enemy successfully reaches the bottom of the screen, the player's **health decreases**. Once the health bar is fully depleted, the game ends. This creates tension and motivates quick reactions.

- **Game Over Logic**
  The game ends when **health reaches zero** or the player fails to stop too many enemies. After the game ends, the final score is displayed and the player is given an option to **submit their score** to the backend.

- **High Score Submission**
  Upon game over, the score is sent to the Flask backend through an API call. This data is stored in a backend file or database for leaderboard purposes.

- **Leaderboard Display**
  The Flask backend can retrieve the **top scores** via a GET request and display them in a leaderboard format. This adds a competitive element to the game.

3.2 Users and Their Characteristics

There are two primary user roles in the project:

1. **Player (End-User)**
- Opens the game in any modern browser.
- Uses keyboard keys to **control movement and shoot** enemies.
- Can choose the difficulty level.
- Plays the game, tries to maintain health, and aims to beat high scores.
- Views score and submits it to the leaderboard upon game over.
- Doesn't require any technical knowledge to play.

2. **Administrator (Developer)**
- Has access to the backend files, APIs, and server configuration.
- Can monitor game logs, debug errors, and improve game mechanics.
- Has the ability to manage score data (add/edit/delete if needed).
- Responsible for **hosting and maintaining** the Flask backend and restarting it in case of downtime or crashes.

3.3 Features of the Project

"Snipe and Strike" offers a wide range of features that contribute to its popularity and ease of access:

- **Browser-Based Gameplay**
  No need to install software. The game can be played directly in Chrome, Firefox, or Edge with just one click.

- **Real-Time Interaction**
  The gameplay is dynamic and uses JavaScript for real-time animations, enemy movement, bullet handling, and collision detection.

- **Difficulty Modes**
  Players can choose from Easy, Medium, and Hard levels. These modes adjust the speed and frequency of enemies, catering to casual and hardcore gamers.

- **Responsive Controls**
  Instant feedback on keyboard input ensures that shooting and movement feel natural and responsive.

- **High Score Leaderboard**
  Scores are stored and displayed on a central leaderboard, encouraging players to keep playing and improve their rankings.

- **Backend Integration with Flask**
  Secure score handling is done using Python's Flask framework. APIs are used for submitting and retrieving scores efficiently.

- **Lightweight Design**
  The game loads quickly and runs smoothly on machines with low specifications, making it widely accessible.

3.4 Admin Features (Developer Side)

The developer, acting as the administrator, has the following privileges and responsibilities:

- **Monitor Score Submissions**
  Can view or log each score submitted to ensure backend functionality and fairness.

- **Manage Score Data**
  Scores can be edited or deleted from the storage file or database if needed (e.g., for debugging or testing).

- **Server Management**
  The developer must ensure the Flask server is live. If it crashes, the admin can restart it easily through command line or a hosting dashboard (like Render or Replit).

- **Add New Features**
  The developer can enhance the game by introducing new enemy types, levels, or power-ups.

- **Debugging and Testing**
  Handles fixing bugs, performance tuning, and upgrading libraries or frameworks used in the game.

3.5 User Features (Player Side)

From the player's perspective, the game offers the following features:

- **One-Click Start**
  No setup or installation—just click and start playing immediately in the browser.

- **Difficulty Selection**
  Players can choose between three difficulty modes depending on their skill and mood.

- **Smooth Gameplay Controls**
  Move the UFO with arrow keys and shoot enemies using the spacebar. No lag or delay in input response.

- **Real-Time Score & Health Display**
  The game UI continuously updates the current score, remaining health, and shooting feedback.

- **High Score Display**
  At the end of the game, the player's score is compared with the leaderboard, adding a motivational factor.

- **Option to Replay**
  After submitting the score, the player can click **Play Again** to restart the game immediately.

CHAPTER 4

SYSTEM REQUIREMENTS

4.1 Functional Requirements

Functional requirements define the core functionality that the system **must** deliver. For the "Snipe and Strike" game, the following functionalities are essential:

- **Start Screen & Difficulty Selection**
  The game initiates with a welcoming interface where the user can choose from multiple difficulty levels: **Easy**, **Medium**, or **Hard**. Each difficulty adjusts the **enemy spawn rate and speed**, offering varied levels of challenge.

- **Player Movement**
  The player controls a UFO character, which can move **horizontally (left and right)** across the screen using keyboard inputs (typically the **arrow keys** or **A/D** keys). The movement is smooth, lag-free, and restricted within screen boundaries.

- **Shooting Mechanism**
  Pressing the **spacebar** fires bullets vertically upward from the UFO's position. These bullets are the primary attack mechanism used to eliminate approaching enemy UFOs.

- **Enemy UFO Spawning**
  Enemy UFOs continuously spawn at the top of the screen at **random X-positions** and descend toward the bottom. The **spawn rate** and **speed** vary with the selected difficulty level, maintaining progressive challenge.

- **Collision Detection**
  The game detects when a bullet **collides** with an enemy UFO. On collision:
  - The enemy is destroyed with a visual animation or sound.
  - The player is awarded points.
  - The bullet is removed from the screen.

- **Health Management**
  Each time an enemy crosses the screen without being shot, the **player's health is reduced**. The health status is displayed on the UI, and acts as a timer of survival.

- **Game Over Condition**
  The game ends in either of the following scenarios:
  - The **health reaches 0**.
  - A critical number of enemies pass the screen. A "Game Over" screen is displayed along with the final score and an option to restart or submit the score.

- **Score Display**
  During gameplay, the **live score** is prominently shown in the top corner. The score increases based on the number of enemies destroyed.

- **Backend Communication**
  Upon game completion, the score is **submitted to a Flask backend** using REST API calls (typically POST request). Similarly, the leaderboard is fetched using a **GET request** and displayed to the player.

4.2 Non-Functional Requirements

Non-functional requirements deal with how the system performs rather than what it does. For this project, they ensure a smooth, user-friendly, and scalable experience:

- **Performance**
  The game must maintain a **minimum frame rate of 30 FPS** to ensure smooth animations and responsive controls. It uses JavaScript's requestAnimationFrame() to optimize performance.

- **Scalability**
  The backend, though lightweight, must support **multiple concurrent score submissions** without failing. This is important if multiple players are submitting scores simultaneously (especially in a class demo or exhibition).

- **Reliability**
  The game must **not crash unexpectedly** during normal play. Backend services should consistently respond to API calls and properly store score data.

- **Security**
  Although it's a small academic game, the **backend endpoints** should not allow unauthorized write access. Basic validation and security measures (like sanitizing input) should be implemented to protect stored data.

- **Usability**
  The game should be **intuitive and beginner-friendly**. First-time users should be able to understand controls and objectives without reading a manual.

- **Compatibility**
  The game must function properly on modern web browsers like **Google Chrome**, **Microsoft Edge**, and **Mozilla Firefox**, across desktop platforms (Windows, MacOS, Linux).

- **Maintainability**
  The codebase should follow **modular design principles** so that each part (UI, logic, backend) can be easily updated or debugged without affecting the entire system.

4.3 Design Goals

The "Snipe and Strike" game is designed with certain key principles in mind to deliver an enjoyable and technically sound experience:

- **Simplicity**
  The game should have a clean and minimal user interface. Controls must be easy to understand and visuals should not overwhelm the player. It focuses on core mechanics without unnecessary complexity.

- **Focus on Gameplay**
  Special emphasis is given to **real-time shooting**, **accurate collision detection**, and **smooth control** of the player's UFO. The gameplay should be engaging enough to encourage repeat plays.

- **Visual Clarity**
  Enemy UFOs are designed in **bright contrasting colors** to easily distinguish them from the background. Health, score, and instructions are clearly visible. The background remains non-distracting but visually appealing.

- **Performance Optimization**
  The game uses **efficient JavaScript rendering techniques**. By relying on requestAnimationFrame() instead of timers, the game loop stays optimized, even on low-end machines.

- **Separation of Concerns**
  The system is divided logically:
  - **Frontend** handles player interaction, visuals, and real-time mechanics.
  - **Backend (Flask)** only manages high score storage and retrieval. This ensures that gameplay doesn't suffer due to backend operations and keeps both systems loosely coupled.

CHAPTER 5

SYSTEM DESIGN

5.1 Primary Design Phase

The **primary design phase** of the *Snipe and Strike* game emphasizes a **modular approach**, which allows the system to be developed, tested, and maintained more effectively. The game is divided into several logical modules, each responsible for a specific task.

Modular Breakdown:

Module Name	Purpose
UI Module	Manages visual elements like menus, score display, health bar, and buttons.
Game Engine Module	Controls the core game loop, updates frame-by-frame logic, and enemy timing.
Player Control Module	Captures user input (keyboard keys) and handles UFO movement and firing.
Collision Detection Module	Detects bullet-enemy collisions and enemy-player interactions.
Score Management Module	Tracks, calculates, and displays the real-time score and final score.
Flask API Module	Handles backend communication to store and retrieve high scores.

Benefits of Modular Design:

- **Reusability** – Each module can be reused or improved without affecting others.
- **Debugging Simplicity** – Errors are easier to isolate.
- **Scalability** – New features (like power-ups, new enemy types) can be added efficiently.

5.2 Secondary Design Phase

This phase involves a **more detailed explanation of each module** and how they interact with one another.

Frontend Architecture (HTML/CSS/JavaScript):

- **Rendering Approach:**
  - Utilizes either a canvas element for rendering player, enemies, and bullets.
  - The screen refreshes using requestAnimationFrame() to ensure smooth animations.

- **Enemy Movement & Bullet Logic:**
  - Each enemy object is spawned at random X-coordinates from the top.
  - Bullets move upward and check for collisions with enemies.

- **Game Loop:**
  - All gameplay logic such as score updates, health tracking, and enemy movement is controlled in a continuous loop.

Backend Architecture (Python Flask):

- **API Endpoints:**
  - POST /submit-score – Receives JSON-formatted score data and stores it in a file.
  - GET /get-highscores – Returns the top N scores for leaderboard display.

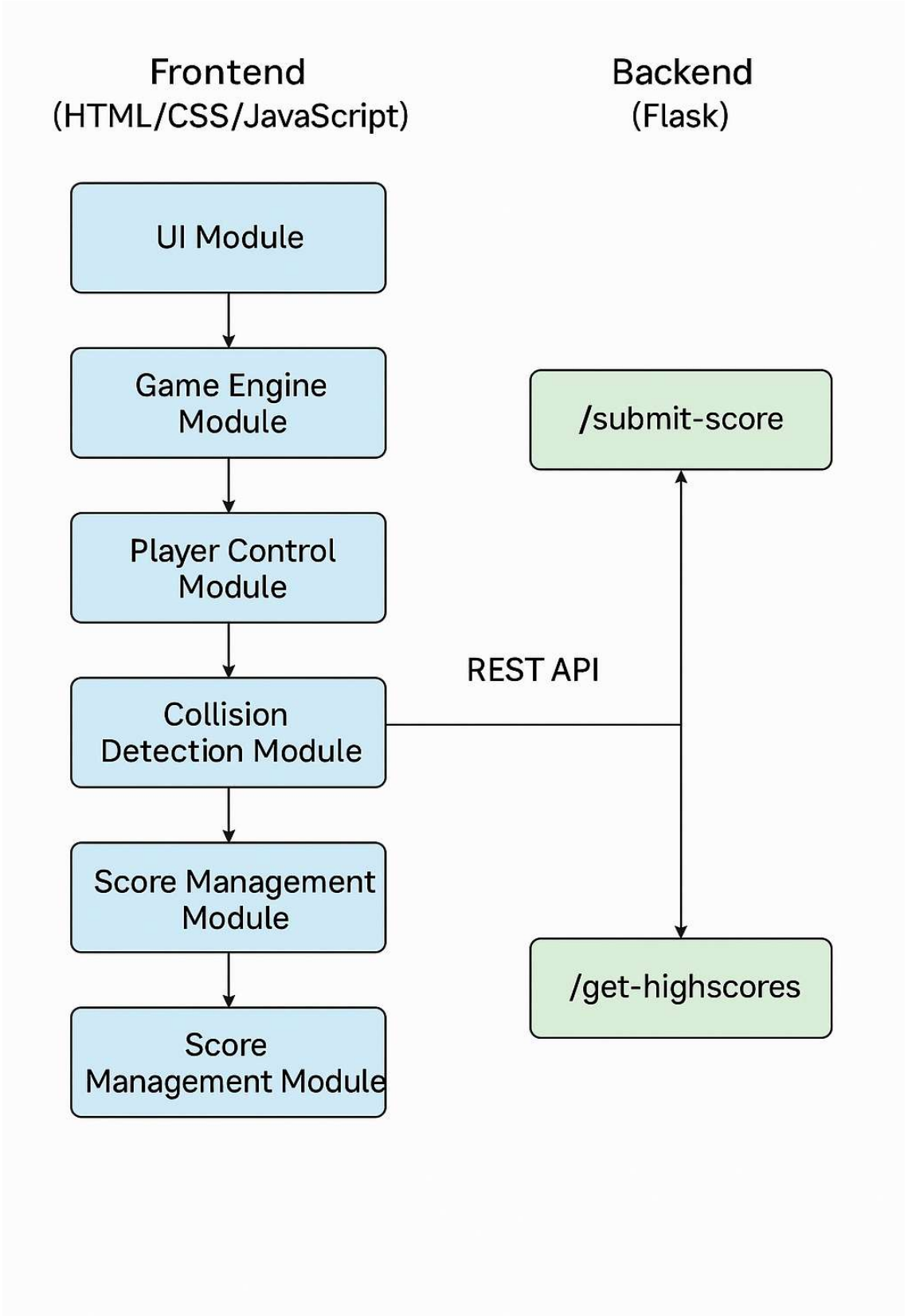
- **Data Handling:**
  - Scores are validated and stored securely.
  - Backend can be hosted locally or on platforms like Render/Replit.

Frontend–Backend Communication:

Component	Technology	Purpose
AJAX/Fetch API	JavaScript	Sends score data on game over
Flask API	Python (Flask)	Stores and retrieves high scores
JSON Format	Data Standard	Ensures smooth data transmission

5.3 System Architecture Design.

The following diagram visually represents the interaction between different components of the system:



Chapter 6

Project Flow

6.1 Game Initialization Stage

1. **Start Screen:**
    - On launching the game, users are welcomed with a start screen where they can select the difficulty level (Easy, Medium, Hard).
    - The interface is intuitive with mouse-click buttons and game instructions.

2. **Difficulty Selection:**
    - The difficulty selected affects enemy spawn rate, speed, and bullet availability.
    - Based on the selection, game parameters are set before entering the game loop.

6.2 Player Interaction Stage

1. **UFO Movement:**
    - Player can control the UFO using the left and right arrow keys.
    - Movement is smooth and responsive using event listeners in JavaScript.

2. **Shooting Mechanism:**
    - Pressing the spacebar shoots a bullet upward from the UFO's current position.
    - There is a short delay between shots depending on the difficulty level.

6.3 Game Engine Stage

1. **Enemy Spawning:**
    - Enemies are spawned from the top of the screen at random horizontal positions.
    - The spawn frequency is controlled by difficulty level.

2. **Bullet and Enemy Collision:**
    - The game constantly checks for collision between bullets and enemy UFOs using bounding box detection.
    - Upon successful hit, the enemy disappears, and score increases.

3. **Health Management:**
    - If any enemy reaches the bottom of the screen, player health decreases.
    - Game ends when health reaches zero.

6.4 Backend Communication Stage

1. **Score Submission:**
    - When the game ends, the final score is sent to the Flask backend using a POST API (/submit-score).
    - The backend validates and stores the score securely.
2. **Leaderboard Retrieval:**
    - The frontend sends a GET request (/get-highscores) to fetch and display the top scores from the backend.
    - Scores are shown in descending order with usernames if applicable.

6.5 Output Stage

1. **Score Display:**
    - During gameplay, the current score and health are shown on-screen.
    - At game over, final score and high score list are displayed.

2. **Restart Option:**
    - Users can restart the game or go back to difficulty selection using buttons.

6.6 Data Flow Diagram (DFD)

Level 0 - Context Diagram:

- Player interacts with the "Snipe and Strike" system.
- Player provides inputs: movement and shooting.
- System outputs: enemy motion, score updates, game over state.
- Backend communicates via API for high score tracking.

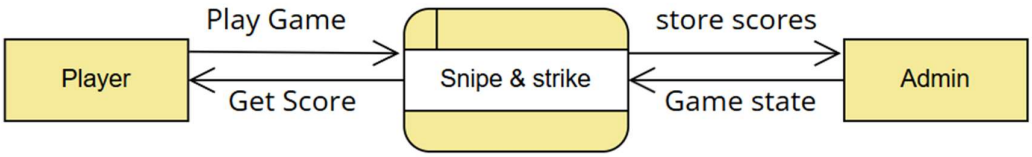


Fig. 1: Level 0 Data Flow Diagram

Level 1 - Detailed DFD:

- Inputs (keyboard/mouse) are captured and passed to the game engine.
- Enemy and bullet movement are handled in a loop.
- Flask APIs handle data storage and retrieval for scores.

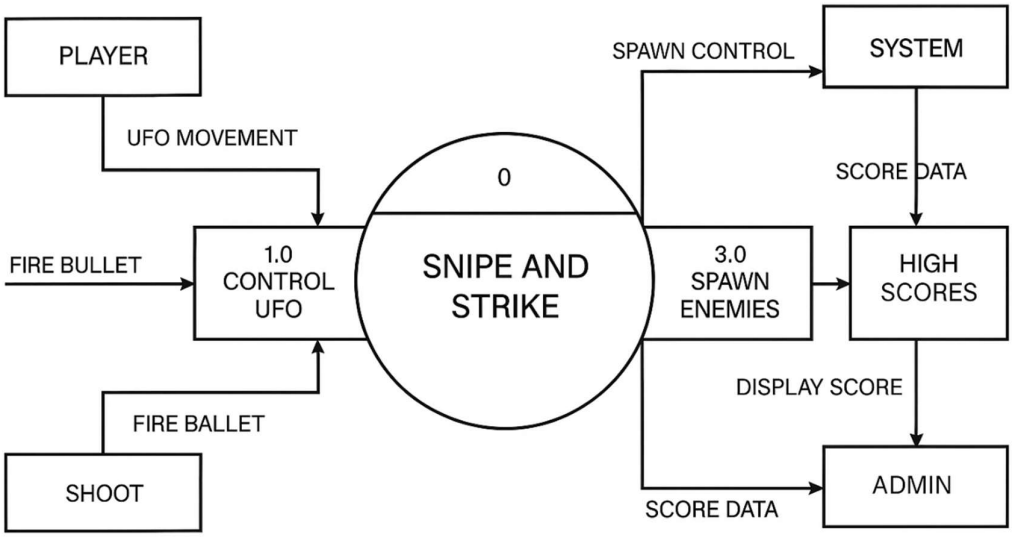


Fig. 2: Level 1 Data Flow Diagram

6.7 Use Case Diagram

- **Player:** Selects difficulty, moves UFO, shoots enemies, views score, restarts game.
- **System:** Handles enemy spawning, collision detection, score tracking, game over logic, and API communication.

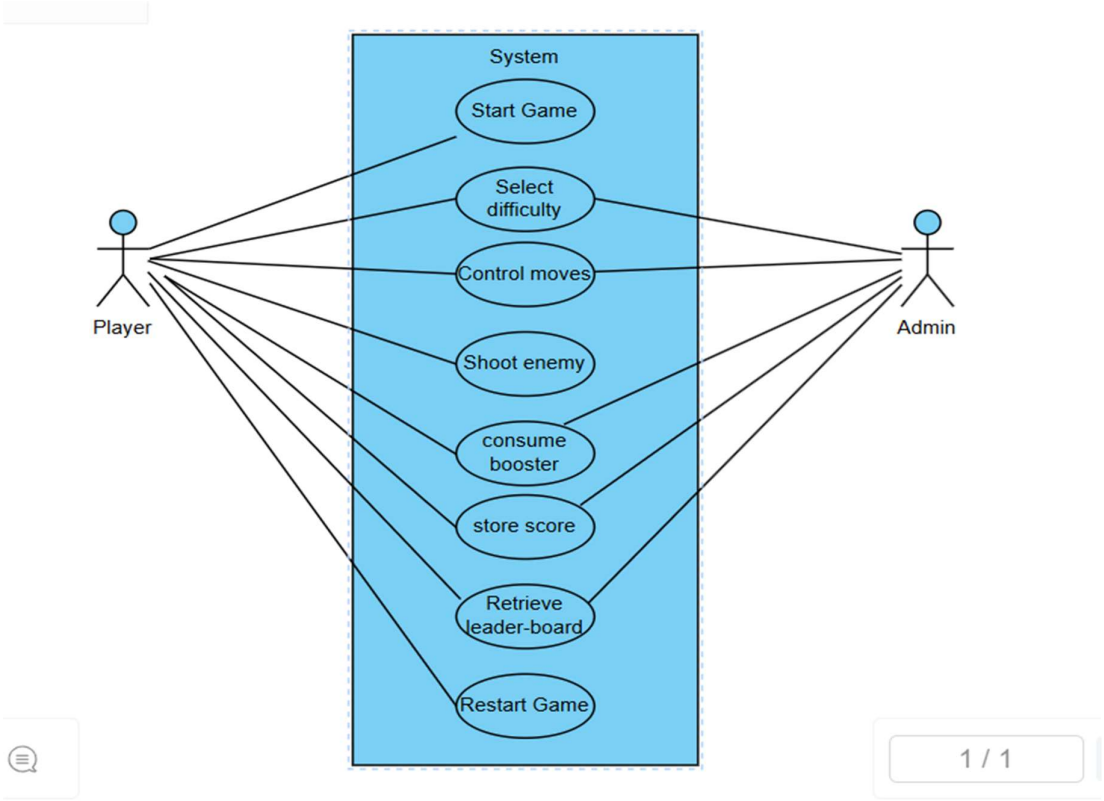


Fig. 3: Use Case Diagram

6.8 Flowchart & Algorithm

- **Flowchart of Game Logic:**

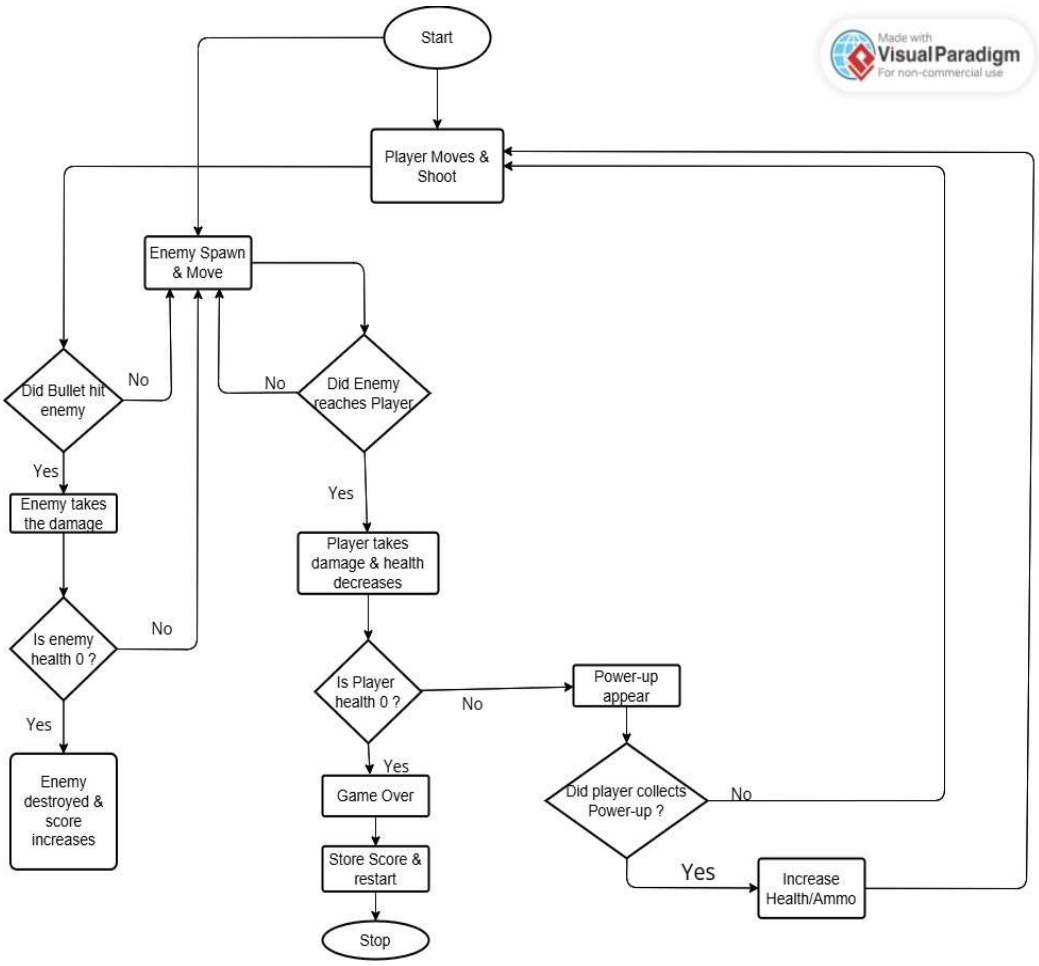


Fig. 4: Flowchart of "Snipe and Strike" Game

- **Algorithm:**

1. Start game and show difficulty selection.
2. Initialize game environment based on selection.
3. Listen for player movement and shooting input.
4. Spawn enemies periodically.
5. Detect bullet-enemy collisions and update score.
6. Check if enemies reach the bottom – reduce health.
7. If health = 0, end game and send score to backend.
8. Fetch and show high scores.
9. Option to restart or quit.
10. End.

Chapter 7

Project Outcome

7.1 Key Outcomes

1. **Engaging Gaming Experience:**
    - The “Snipe and Strike” game delivers an interactive and thrilling user experience, combining fast-paced gameplay with simple controls.
    - Players are immersed in a UFO shooting environment that keeps them alert and entertained.

2. **Real-Time Scoring System:**
    - The score is updated in real-time as players destroy enemy UFOs.
    - This provides immediate feedback and encourages players to improve their performance continuously.

3. **Progressive Difficulty:**
    - Players can choose between Easy, Medium, and Hard modes.
    - Each level dynamically adjusts enemy speed and spawn frequency, maintaining challenge and replayability.

4. **Responsive Controls and Mechanics:**
    - Smooth movement of the UFO and immediate shooting response enhances playability.
    - Collision detection and health reduction are processed in real time for a seamless experience.

5. **Backend Integration:**
    - The game records player scores and communicates with a Flask backend.
    - This helps in storing and retrieving high scores and maintaining a basic leaderboard system.

7.2 Social Impact

1. **Cognitive Skill Development:**
    - Enhances players' hand-eye coordination, reaction time, and decision-making under pressure.
    - Particularly beneficial for students looking to sharpen mental agility in a fun way.

2. **Encouragement of Creativity:**
    - Game development projects like “Snipe and Strike” encourage students to explore animation, event handling, and logic structuring in programming.

3. **Inspiration for Learning Game Development:**
    - Projects like this serve as a gateway for learners to dive into more complex game development with frameworks or engines such as Pygame or Unity.

7.3 Technological Advancement

1. **Web Technologies Utilized:**
    - Built using **HTML**, **CSS**, and **JavaScript** for frontend logic.
    - Backend functionalities are supported through **Python Flask** for API-based communication.

2. **Real-Time Interactivity:**
    - The use of animation loops and event listeners allows real-time game interactions like shooting, collisions, and enemy movement.

3. **REST API Integration:**
    - POST and GET APIs handle score submission and leaderboard retrieval.
    - Demonstrates practical knowledge of frontend-backend interaction in real-world applications.

4. **Data Handling and Scalability:**
    - The backend is capable of handling multiple users’ scores, showing potential for future extensions like user authentication, multiplayer mode, etc.

7.4 User Interface

1. Start Screen

- The entry point of the game where the player selects the difficulty level.
- Clean layout with colorful buttons, brief instructions, and an engaging visual design.

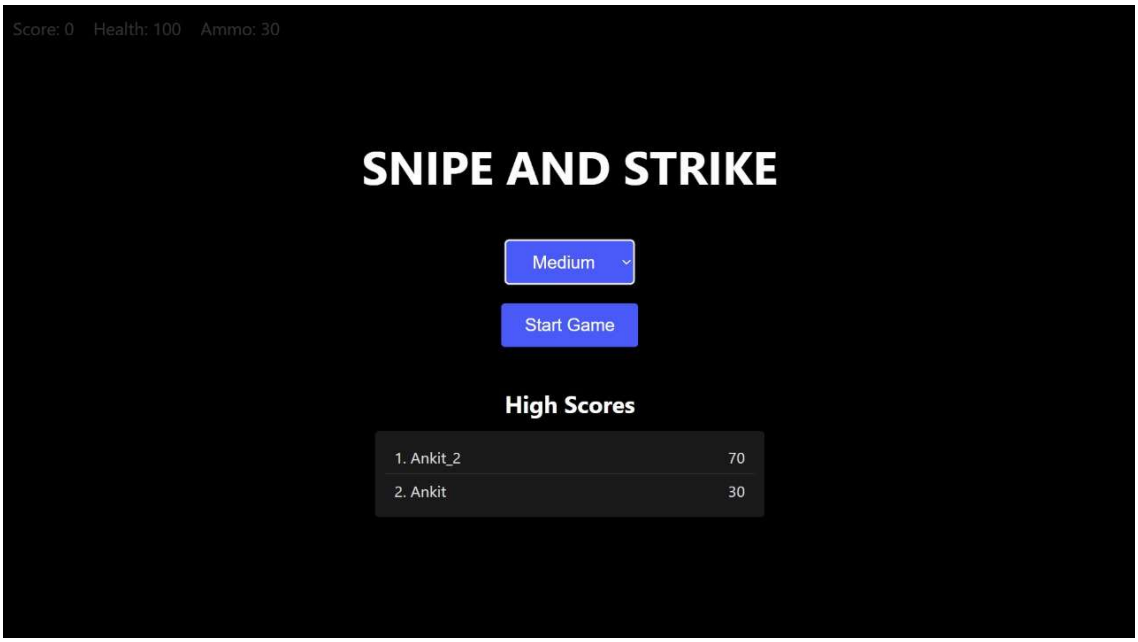


Fig.5 Start Screen of Snipe and Strike

2. Gameplay Screen

- The main game interface where:
  - Player UFO is visible at the bottom.
  - Enemy UFOs descend from the top.
  - Bullets are fired using the spacebar.
- Real-time display of score and remaining health.

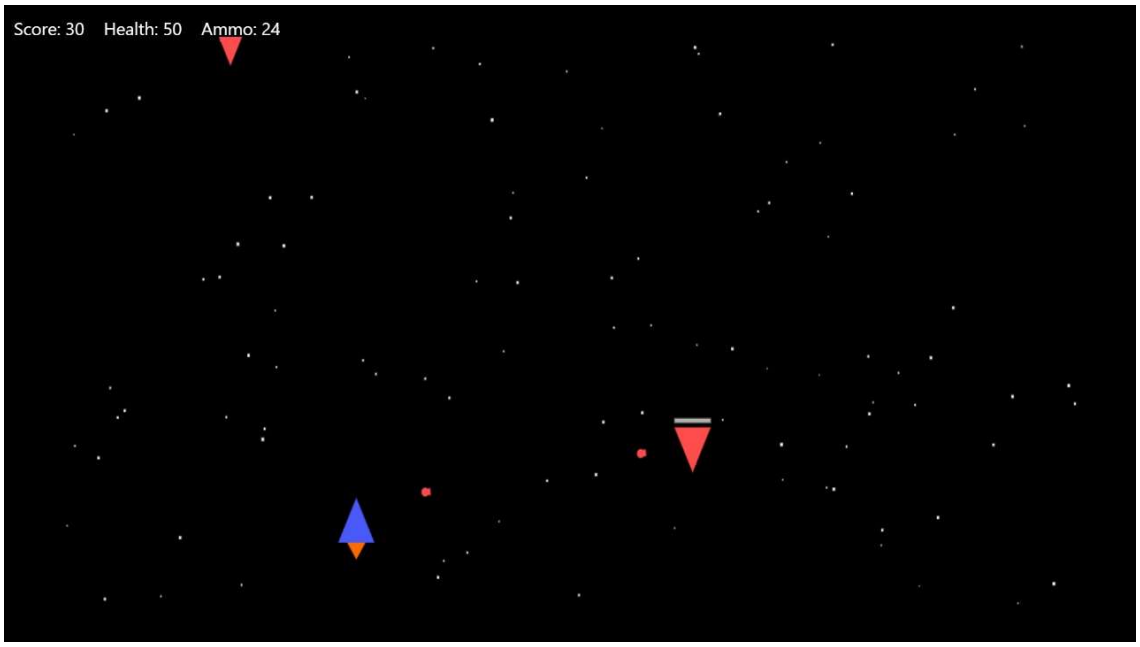


Fig.6 Gameplay Interface of Snipe and Strike

3. Game Over Screen

- Displayed when the player's health reaches zero.
- Shows final score with an option to:
  - Restart the game.
  - View high scores (retrieved from backend).
  - Return to main menu.

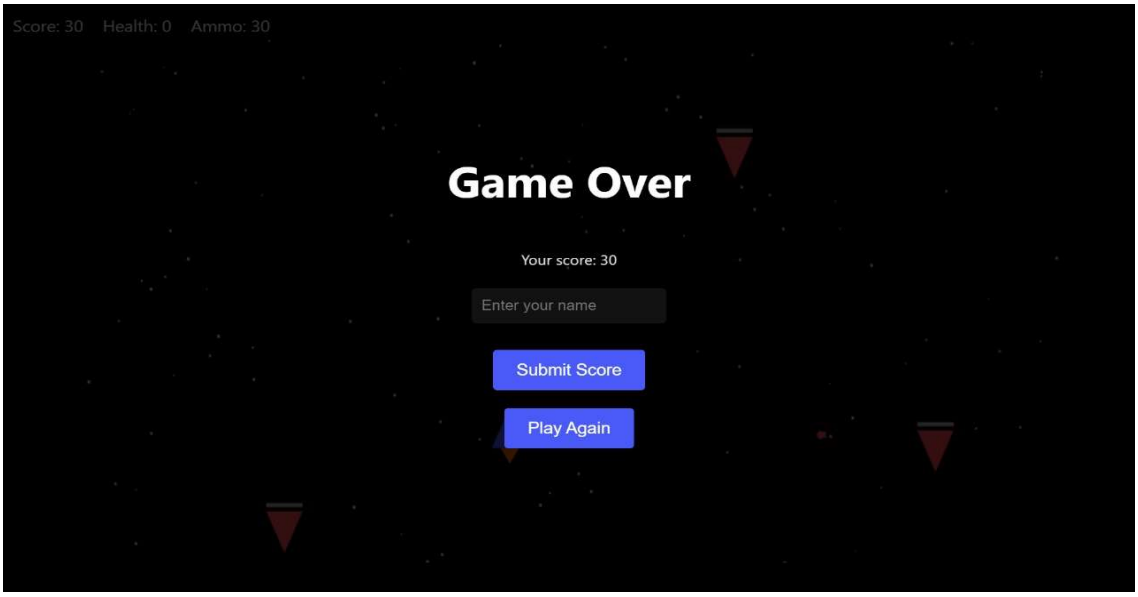


Fig.7 Game Over Screen

4. Leaderboard Interface

- Shows top scores retrieved via Flask API.
- Allows players to track their ranking.
- Option to play again or change difficulty.

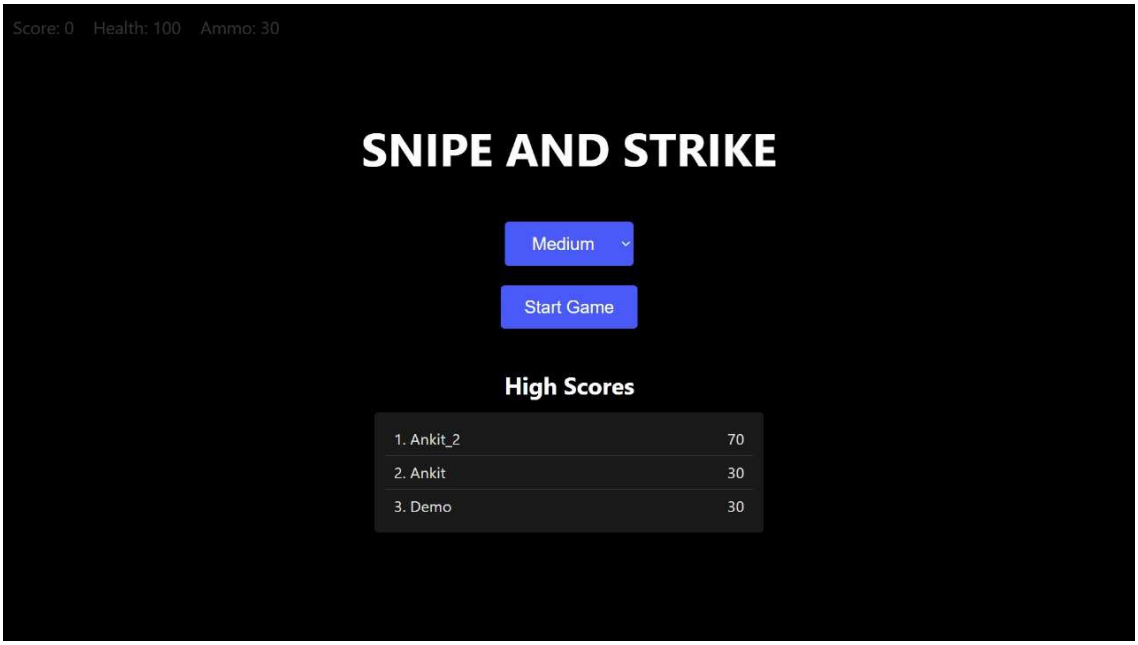


Fig.8 Leaderboard Preview

Chapter 8

References

1. **MDN Web Docs (Mozilla Developer Network)**
   HTML, CSS, and JavaScript documentation
   https://developer.mozilla.org/

2. **W3Schools**
   Web development tutorials and code snippets
   https://www.w3schools.com/

3. **GeeksforGeeks**
   JavaScript game development, Flask tutorials, and collision detection techniques
   https://www.geeksforgeeks.org/

4. **Real Python**
   Flask backend development and REST API tutorials
   https://realpython.com/

5. **Stack Overflow**
   Community-driven Q&A platform for debugging and code optimization
   https://stackoverflow.com/

6. **FreeCodeCamp**
   Beginner to intermediate game development guides and interactive tutorials
   https://www.freecodecamp.org/