# Fruit Ninja 2D – Documentation

## 1. Project Overview

Project Title: Fruit Ninja 2D  
Technology: Python, Pygame Library  
Type: 2D Action / Arcade Game  
Objective:  
The game aims to simulate the original Fruit Ninja gameplay, where the player slices fruits by swiping the mouse cursor while avoiding bombs.

## 2. Project Description

Fruit Ninja 2D is a fast-paced arcade-style game.  
Fruits are thrown randomly onto the screen with varying speeds and trajectories.  
The player must “slice” the fruits by swiping over them using the mouse.  
Each sliced fruit adds to the player’s score, but hitting a bomb ends the game.  
The goal is to achieve the highest score possible before missing too many fruits or hitting a bomb.

## 3. System Requirements

Software:  
- Python 3.8 or higher  
- Pygame 2.0 or higher  
  
Hardware:  
- Minimum 2 GB RAM  
- Any dual-core processor  
- Screen resolution: 1280×720 recommended

## 4. Modules and Components

- main.py: Entry point of the game.  
- fruit.py: Contains the Fruit class responsible for fruit movement, spawning, and slicing detection.  
- bomb.py: Contains the Bomb class with explosion animation.  
- effects.py: Manages slicing effects, sound, and particle animations.  
- assets/: Folder containing images (fruits, bomb, background) and sound effects.

## 5. Game Logic

1. Game initializes and shows the start screen.  
2. Fruits and bombs are launched at random intervals and angles.  
3. Player moves the mouse — a white line (blade) follows the cursor.  
4. If the blade crosses a fruit → it’s sliced.  
5. If the blade hits a bomb → game over.  
6. Score increases per fruit; lives decrease if a fruit falls uncut.

## 6. Scoring System

- Slice Fruit: +10 points  
- Slice Multiple Fruits (Combo): +20 points  
- Miss Fruit: -1 Life  
- Hit Bomb: Game Over

## 7. Controls

- Slice: Move the mouse (swipe)  
- Quit: ESC or close window  
- Restart: Press Space on Game Over screen

## 8. Libraries Used

- pygame: Rendering graphics, event handling, sound  
- random: Random fruit spawn and speed generation  
- math: Calculating angles and blade direction  
- sys: Exiting the game loop safely  
- os: Asset loading

## 9. Future Enhancements

- Add combo detection (multiple fruits sliced in one swipe).  
- Add difficulty levels (spawn rate, bomb frequency).  
- Add power-ups (slow motion, double points).  
- Add leaderboard to save top scores.  
- Add menu & pause screen with GUI buttons.

## 10. Learning Outcomes

- Understand 2D physics and gravity simulation.  
- Work with real-time mouse input and collision detection.  
- Manage sprites, images, and sound in Python.  
- Handle game loops, frame rate, and updates.