To begin the project, I created a brief design outline based on a game kit I selected from the Unity Asset Store. After identifying the core mechanics and structure, I started building the level in Unity. I used Tilemap to construct most of the environment, ensuring consistency and efficient level iteration. For traps and platforms, I implemented separate GameObjects to allow precise control over their behavior and physics interactions.

Most visual assets came from the selected game kit. In addition, I generated the heart icon used for the health system and the instructional control screen using AI tools. Music and sound effects were sourced from itch.io to complete the audiovisual design.

From a technical perspective, I focused on maintaining a decoupled architecture using C# events. Systems communicate through events rather than direct references, improving maintainability and scalability. UI classes are responsible solely for rendering, gameplay classes trigger events, and utility systems manage cross-system interactions. As an example of applied gameplay logic, I implemented a trampoline mechanic where bounce height scales dynamically based on the player's fall velocity.

The inventory system was the most technically demanding feature. It is built using ScriptableObjects for item data, logic classes for inventory management, and UI components for visualization. I chose to use a List instead of an array to support dynamic resizing and automatic reordering when items are consumed.

For persistence, I implemented a JSON-based save system that stores player position, health, and inventory data. I adapted a previously developed save framework – created in the game that I launched for steam - to meet this project's specific requirements, including checkpoint-based saving and structured data serialization.