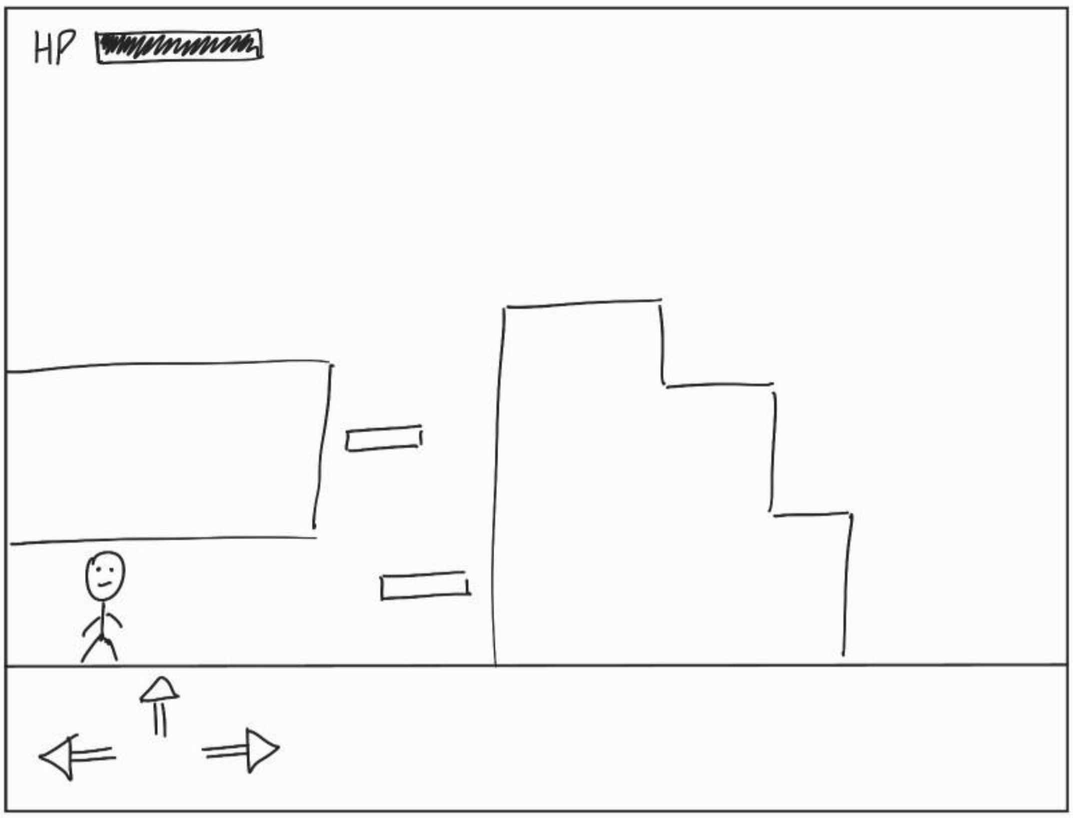
Appendix

Interview 1 – 14/4/2022

(14:49) Me: Hey, I heard your younger brother has developed a bad habit of eating lots of junk food?
(14:49) Client: Yes, I try to keep away from it, but pizzas are too tasty. Plus, it is hard to remind myself.
(14:50) Me: Huh, I wonder if there is any way, we can approach the problem? What are his hobbies or interests?
(14:50) Client: Well, he does like to spend a lot of time playing video games and watching television.
(14:51) Me: Hey! What if I make a game which encourages healthy eating habits, would he play that?
(14:51) Client: Yes, that could work. Video games certainly interest me a lot.
(14:52) Me: Would a platformer game be suitable to his interests?
(14:52) Client: Maybe, it is important to make it easy to understand otherwise he may lose interest.
(14:53) Me: Sure, I will consult you later after sketching up ideas for the design and layout of the game.

Interview 2 – 01/05/2022

(17:37) Me: Hello, I have drawn up some diagrams regarding the layout of the platformer design and the UI of the game. Would you like any changes to the following?



(17:37) Client: This seems fine. However, I think it would be better to just implement keyboard input rather than on-screen buttons for simplicity. Most computers are not touchscreen. Also, I think hearts would work better to show the health and make it more simplified
(17:38) Me: Noted. Furthermore, how should I approach the 'food item' objects in the game. I could have them as static obstacles or moving hazards on the platforms.
(17:38) Client: I think having them as obstacles makes sense since they are food items. You could make the food items differentiable through images.

(17:38) Me: That's a good idea, I will also make them easy to collect.
(17:39) Client: Yep, I would then say you can then finalize the design and workings of the software.
(17:39) Me: Ok, I shall then soon start development of the software.

Interview 3 – 18/05/2022

(21:17) Me: Now that we have agreed upon the basic working of the software, we should decide upon the specific criteria you would need for the software to be useful to you.
(21:17) Client: Ok, basically I just want the game to be immersive and entertaining as it would be the most impactful, I feel. I think that the collectible system is good where the user picks up different food items.
(21:17) Me: That's a good idea.
(21:18) Client: Except for that, everything else is pretty standard. Make sure the healthy food can be collected by the player, and the unhealthy food damages or harms the player in some way.
(21:18) Me: Yea, I plan to do that using a health point system. Moreover, collecting the food can then restore health points.
(21:19) Client: Great! Can you also make sure all the directional buttons, jump and the controls and interface work properly with keyboard input and have no major bugs or any other problems?
(21:19) Me: Yes, I will also implement colliders to make sure the player does not go out of bounds and is able to move as intended on the platforms.
(21:20) Client: I think that's all then. Oh, remember to design an ending phase for the level, maybe once the player passes a checkpoint of some sort, a flag or a tower would be nice.

Interview 4 – 22/01/2023

(17:45)Me: Hi, you should have received the .zip file with all the program files as well as the program itself. You can unzip the file and then you will be able to run the program. Tell me when you have it running
(17:48)Client: I have received the program and have run it. I have it opened.

We then went through and ticked all the success criteria that were met. The following are snippets of the conversation regarding feedback from the client.

Success Criteria 1:
(17:49)Client: The character model, food items and the level design is all visible.

Success Criteria 2:
(17:50)Client: Yes, the controls are responsive and I can move the character in all directions.

Success Criteria 3:
(17:52)Client: I can pickup an unhealthy food item by walking into it and I lose a heart, or a health point as you call it.

Success Criteria 4:
(17:54)Client: Similarly, I can pickup a healthy food item by walking into it but this time I gain a heart/health point.

Success Criteria 5:

(17:56)Client: I was able to complete the whole level easily. It was convenient when I was not allowed to fall off the edges. I think it makes the game easier to play.

Success Criteria 6:
(17:58)Client: Yes, it was clear in the display how many hearts or health I had. It is a good motivator to stay away from unhealthy food.

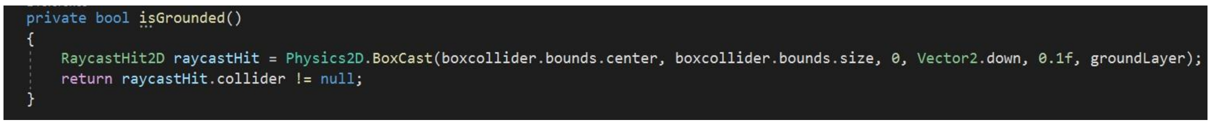
Success Criteria 7:
(18:00)Client: After I lost all my hearts, the character did move back to the starting point at the flag checkpoint. Good way to add replay ability and will teach my brother to stay away from unhealthy food

Success Criteria 8:
(18:02)Client: After I crossed the flag at the end of the level, the “You Won!” screen was displayed. That successfully indicated that the level was completed. I also have the option the exit the application.

Success Criteria 9:
(18:03)Client: I can exit the application by pressing the ‘E’ button on the keyboard.

(18:05)Me: So now that we have gone through all the success criteria, do you have any other feedback for me or any difficulties you have faced separate from the success criteria? (18:06)Client: I know I said it should be simple to understand, but I think you could have added something which would increase the engagement and would be harder to complete, such as moving platforms or changes to the character’s attributes such as the speed or jump height. You can also add more benefits to picking healthy food so that it would be an additional goal along with avoiding unhealthy food. Lots of popular games use a point or achievement system. Lastly, designing more levels and make it a layered game would only be beneficial to have a longer experience.

isGrounded() method:



```
private bool isGrounded()
{
    RaycastHit2D raycastHit = Physics2D.BoxCast(boxcollider.bounds.center, boxcollider.bounds.size, 0, Vector2.down, 0.1f, groundLayer);
    return raycastHit.collider != null;
}
```

Program Code:

Health Class

```
using UnityEngine;

public class Health : MonoBehaviour
{
    [Header ("Health")]
    [SerializeField] private float startingHealth; // Variable value for initial health
    public float currentHealth; // Public variable for current health
public Transform Checkpoint;
     private void Awake() // Sets inital health equal to current
health
```

```
    {
        currentHealth = startingHealth;
    }

    public void AddHealth(float _value)
    {
        currentHealth = Mathf.Clamp(currentHealth + _value, 0, startingHealth);
}
    public void RemoveHealth(float _value)
    {
        currentHealth = Mathf.Clamp(currentHealth - _value, 0, startingHealth);

        if (currentHealth == 0)
        {
            transform.position = Checkpoint.position;
            AddHealth(startingHealth);
            Camera.main.transform.position = Checkpoint.position;
        }
    }
    public void Respawn()
    {
        AddHealth(startingHealth);
    }
}
```

Walk Class

```
using UnityEngine;

public class Walk : MonoBehaviour
{
    [SerializeField] private float speed;
    [SerializeField] private float jump;
    [SerializeField] private LayerMask groundLayer;
private Rigidbody2D body;     private Animator
anim;
    private BoxCollider2D boxcollider;

    public Transform Endpoint;
public GameObject gameOverScreen;

    private void Awake()
    {
        body = GetComponent<Rigidbody2D>();
anim = GetComponent<Animator>();
boxcollider = GetComponent<BoxCollider2D>();
gameOverScreen.SetActive(false);
    }      private void
Update()
    {
        float horizontalInput = Input.GetAxis("Horizontal");
body.velocity = new Vector2(Input.GetAxis("Horizontal") * speed,
body.velocity.y);

if (horizontalInput > 0.01f)
             transform.localScale = Vector3.one;
else if (horizontalInput < 0f)
            transform.localScale = new Vector3(-1, 1, 1);
if (Input.GetKeyDown(KeyCode.Space))
```

```
        {
            if (isGrounded())
            {
                body.velocity = new Vector2(body.velocity.x, jump);
            }
        }
        anim.SetBool("Run", horizontalInput != 0);

        // Control jump height
        if(Input.GetKeyUp(KeyCode.Space) && body.velocity.y > 0)
        {
            body.velocity = new Vector2(body.velocity.x, body.velocity.y / 2);
        }
        if (Input.GetKey(KeyCode.Escape))
            Application.Quit();
    }
private bool isGrounded()
    {
        RaycastHit2D raycastHit = Physics2D.BoxCast(boxcollider.bounds.center,
boxcollider.bounds.size, 0, Vector2.down, 0.1f, groundLayer);
        return raycastHit.collider != null;
    }
private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.transform.tag == "Endpoint")
        {
            Endpoint = collision.transform;
gameOverScreen.SetActive(true);
            collision.GetComponent<Collider2D>().enabled = false;
        }
    }
}
```

Unhealthy Class

```
using UnityEngine;

public class Unhealthy : MonoBehaviour
{
    [SerializeField] private float damage;

    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.tag == "Player")
        {
            collision.GetComponent<Health>().RemoveHealth(damage);
gameObject.SetActive(false);
        }
    }
}
```

Healthy Class

```
using UnityEngine;

public class Healthy : MonoBehaviour
{
    [SerializeField] private float health;
     private void OnTriggerEnter2D(Collider2D
collision)
```

```
    {
        if (collision.tag == "Player")
        {
            collision.GetComponent<Health>().AddHealth(health);
gameObject.SetActive(false);
        }
    }
}
```

Healthbar Class

```
using UnityEngine;
using UnityEngine.UI;

public class Healthbar : MonoBehaviour
{
    [SerializeField] private Health playerHealth;    // Variable value for the amount
of player 'health' points
    [SerializeField] private Image totalhealthBar;   // Variable value for the total
health the healthbar can display
    [SerializeField] private Image currenthealthBar; // Variable value for current
health the healthbar displays

    private void Start() // Runs before update method to quantify how much of the
healthbar is displayed
    {
        totalhealthBar.fillAmount = playerHealth.currentHealth / 10;
    }
    private void Update() // Runs every frame to change value of healthbar amount
according to current health
    {
        currenthealthBar.fillAmount  =  playerHealth.currentHealth  /  10;
}
}
```

Camera follow Class

```
using UnityEngine;

public class Camera_follow : MonoBehaviour
{
    private const float V = 4f;
    public float FollowSpeed = V; // Variable for speed at which camera moves while
following
    public Transform target; // Variable for target to be followed. Transform class is
used to determine position of target

    void Update() // Method which is called once per frame
    {
        Vector3 newPos = new Vector3(target.position.x, target.position.y + 3, -10f);
transform.position = Vector3.Slerp(transform.position, newPos, FollowSpeed *
Time.deltaTime);
    }
}
```