```
/// <summary>
/// Regenerate HP, Stamina, MP, and Food each frame.
/// </summary>
private void regenPlayerStats()
{
    if (all_curentStats[3] > 0 && all_curentStats[0] < all_sliderStats[0].maxValue)
all_curentStats[0] += addHP * Time.deltaTime;
    if (staminaRegen && all_curentStats[1] < all_sliderStats[1].maxValue)
    {
        if (cooldown_to_regen_the_stamina <= 0 || numbers_of_enemy_is_in_battle_with_player <= 0)
            all_curentStats[1] += (!playerAttack_script.swordAnimator.GetBool("shield") ? 1f :
0.35f) * addStamina * Time.deltaTime;
        else cooldown_to_regen_the_stamina -= Time.deltaTime;
    }
    if (all_curentStats[2] < all_sliderStats[2].maxValue) all_curentStats[2] += addMP *
Time.deltaTime;
    if (all_curentStats[3] > all_sliderStats[3].minValue) all_curentStats[3] += addFood *
Time.deltaTime;
    if (!staminaRegen) { cooldwonStamina -= Time.deltaTime; if (cooldwonStamina < 0) staminaRegen
= true; }
    if (all_curentStats[3] <= all_sliderStats[3].minValue) all_curentStats[0] -= addHP *
Time.deltaTime;
    if (cooldown_for_take_dmg > 0) cooldown_for_take_dmg -= Time.deltaTime;
}

/// <summary>
/// Apply damage to player, trigger effects and death.
/// </summary>
public void takeDMG(float dmg)
{
    if (cooldown_for_take_dmg > 0) return;
    if (!mainPlayerController_script.block_with_dodge)
    {
        float arrmor = player_can_block_the_Enemy ? CalculateArmor() : all_curentStats[9];
        all_curentStats[0] -= dmg * (1 - arrmor / (arrmor + 100));
        mainCamera_animator.SetTrigger("takeDmg");
        HandleDamageEffects();
    }
    cooldown_for_take_dmg = 0.5f;
    if (all_curentStats[0] <= 0) death_now();
}

/// <summary>
/// Calculate armor with shield and equipment.
/// </summary>
private float CalculateArmor()
{
    float multiplier = (playerAttack_script.cooldown_shield > 0.8 &&
playerAttack_script.cooldown_shield < 9) ? 10 : 1;
    float extra = playerAttack_script.swordAnimator.GetBool("shield") ?
playerInventory_script.sword_echipe_now.protection_for_block : 0;
    return all_curentStats[9] + extra * multiplier;
}

/// <summary>
/// Handle visual/audio effects when taking damage.
/// </summary>
private void HandleDamageEffects()
{
    if ((playerAttack_script.cooldown_shield > 0.8f && playerAttack_script.cooldown_shield < 9)
&& player_can_block_the_Enemy)
    { expertBlock_effect(); generate_voidSound(transform, masterBlock_clip, 0.25f); }
    else if (playerAttack_script.swordAnimator.GetBool("shield") && player_can_block_the_Enemy)
        generate_voidSound(transform, simpleBlock_clip, 0.35f);
    else { generate_voidSound(transform, take_dmg_clip, 0.3f); generate_voidSound(transform,
blood_dmg_effect_clip, 0.2f); }
}

/// <summary>
/// Generate sound effect prefab at a transform.
/// </summary>
public void generate_voidSound(Transform parent, AudioClip clip, float vol)
{
    GameObject go = Instantiate(pref_of_soundEffect_generate);
    AudioSource src = go.GetComponent<AudioSource>();
    src.clip = clip; src.volume = vol; src.Stop();
    go.transform.SetParent(parent); go.transform.localPosition = Vector3.zero;
    src.Play(); Destroy(go, 3);
}

/// <summary>
/// Instantiate visual effect for expert block.
/// </summary>
private void expertBlock_effect()
{
    GameObject e = Instantiate(prefab_of_expertBlock);
    e.transform.SetParent(playerInventory_script.sword_echipe_now.position_for_expertBlock);
    e.transform.localPosition = Vector3.zero; e.transform.rotation = Quaternion.identity;
    e.transform.localScale *= UnityEngine.Random.Range(scale_of_expertBlock_Effect * 0.6f,
scale_of_expertBlock_Effect);
    Destroy(e, 2);
}

/// <summary>
/// Handle player death and start reload coroutine.
/// </summary>
private void death_now()
{
    mainPlayerController_script.you_are_death = true;
    death_camera.SetTrigger("death"); death_visual_effect.SetActive(true);
    StartCoroutine(reset_save());
}

/// <summary>
/// Reload scene after death delay.
/// </summary>
IEnumerator reset_save() { yield return new WaitForSeconds(5); SceneManager.LoadScene(3); }

/// <summary>
/// Add XP, level up if needed.
/// </summary>
public void update_XP(float xp)
{
    all_curentStats[4] += xp;
    if (all_curentStats[4] > all_curentStats[5])
    {
        do { all_curentStats[7] += 5; all_curentStats[6]++; all_curentStats[4] -=
all_curentStats[5]; all_curentStats[5] *= 1.3f; }
        while (all_curentStats[4] > all_curentStats[5]);
    }
}

/// <summary>
/// Update a specific stat category and spend points.
/// </summary>
public void update_stats(int cat)
{
    all_curentStats[7]--;
    if (cat != 8 && cat != 3) all_sliderStats[cat].maxValue++;
    else if (cat != 3) all_curentStats[cat]++;
    else playerAttack_script.update_Original_DMG(0.25f);
}

/// <summary>
/// Drink potion, apply buffs, handle cooldown.
/// </summary>
public void drink_Potion(potions_Manager p)
{
    playerAttack_script.original_dmg += p.get_all_bonousPotion(1);
    all_curentStats[0] = Mathf.Min(all_curentStats[0] + p.get_all_bonousPotion(2),
all_sliderStats[0].maxValue);
    all_curentStats[1] = Mathf.Min(all_curentStats[1] + p.get_all_bonousPotion(3),
all_sliderStats[1].maxValue);
    mainPlayerController_script.updateSpeed_drinkPotion_for_Speed(p.get_all_bonousPotion(4));
    StartCoroutine(cooldownEnumerator_to_drink_Potion(p.get_all_bonousPotion(0), p));
}

/// <summary>
/// Potion cooldown coroutine to revert buffs.
/// </summary>
IEnumerator cooldownEnumerator_to_drink_Potion(float cd, potions_Manager p)
{
    whait_to_drink_nexPotion = true; yield return new WaitForSeconds(cd);
    playerAttack_script.original_dmg -= p.get_all_bonousPotion(1);
    mainPlayerController_script.updateSpeed_drinkPotion_for_Speed(1);
    yield return new WaitForSeconds(cd * 0.1f); whait_to_drink_nexPotion = false;
}
```