```cpp


#include "ShooterCharacter.h"
#include "Gun.h"
#include "Components/CapsuleComponent.h"
#include "SimpleShooterGameModeBase.h"

AShooterCharacter::AShooterCharacter()
{
    PrimaryActorTick.bCanEverTick = true;

}

void AShooterCharacter::BeginPlay()
{
    Super::BeginPlay();

    Health = MaxHealth;

    Gun = GetWorld()->SpawnActor<AGun>(GunClass);
    GetMesh()->HideBoneByName(TEXT("weapon_r"), EPhysBodyOp::PBO_None);
    Gun->AttachToComponent(GetMesh(),
      FAttachmentTransformRules::KeepRelativeTransform, TEXT
      ("WeaponSocket"));
    Gun->SetOwner(this);
}

bool AShooterCharacter::IsDead() const
{
    return Health <= 0;
}

float AShooterCharacter::GetHealthPercent() const
{
    return Health / MaxHealth;
}

void AShooterCharacter::Tick(float DeltaTime)
{
    Super::Tick(DeltaTime);

}

void AShooterCharacter::SetupPlayerInputComponent(UInputComponent*
  PlayerInputComponent)
{
    Super::SetupPlayerInputComponent(PlayerInputComponent);

    PlayerInputComponent->BindAxis(TEXT("MoveForward"), this,
        &AShooterCharacter::MoveForward);
    PlayerInputComponent->BindAxis(TEXT("LookUp"), this, &APawn::AddControllerPitchInput);
    PlayerInputComponent->BindAxis(TEXT("MoveRight"), this, &AShooterCharacter::MoveRight);
    PlayerInputComponent->BindAxis(TEXT("LookRight"), this, &APawn::AddControllerYawInput);
    PlayerInputComponent->BindAxis(TEXT("LookUpRate"), this, &AShooterCharacter::LookUpRate);
    PlayerInputComponent->BindAxis(TEXT("LookRightRate"), this, &AShooterCharacter::LookRightRate);
    PlayerInputComponent->BindAction(TEXT("Jump"), EInputEvent::IE_Pressed, this, &ACharacter::Jump);
    PlayerInputComponent->BindAction(TEXT("Shoot"), EInputEvent::IE_Pressed, this, &AShooterCharacter::Shoot);
}

float AShooterCharacter::TakeDamage(float DamageAmount, struct FDamageEvent const &DamageEvent, class AController *EventInstigator, AActor *DamageCauser)
{
    DamageToApply = Super::TakeDamage(DamageAmount, DamageEvent, EventInstigator, DamageCauser);
    DamageToApply = FMath::Min(Health, DamageToApply);
    Health -= DamageToApply;
    UE_LOG(LogTemp, Warning, TEXT("Health left %f"), Health);

    if (IsDead())
    {
        ASimpleShooterGameModeBase* GameMode = GetWorld()->GetAuthGameMode<ASimpleShooterGameModeBase>();
        if (GameMode != nullptr)
        {
            GameMode->PawnKilled(this);
        }

        DetachFromControllerPendingDestroy();
        GetCapsuleComponent()->SetCollisionEnabled(ECollisionEnabled::NoCollision);
    }

    return DamageToApply;
}

void AShooterCharacter::MoveForward(float AxisValue)
{
    AddMovementInput(GetActorForwardVector() * AxisValue);
}

```

```cpp
void AShooterCharacter::MoveRight(float AxisValue)
{
    AddMovementInput(GetActorRightVector() * AxisValue);
}

void AShooterCharacter::LookUpRate(float AxisValue)
{
    AddControllerPitchInput(AxisValue * RotationRate * GetWorld()-
      >GetDeltaSeconds());
}

void AShooterCharacter::LookRightRate(float AxisValue)
{
    AddControllerYawInput(AxisValue * RotationRate * GetWorld()-
      >GetDeltaSeconds());
}

void AShooterCharacter::Shoot()
{
    Gun->PullTrigger();
}

// void AShooterCharacter::LookUp(float AxisValue)
// {
//  AddControllerPitchInput(AxisValue);
// }


```