Plugins in Game Design

Zhou(Joe) Zhang, Ph. D.
Assistant Professor
SUNY Farmingdale State College
ZhangZ@farmingdale.edu

Importance of Plugins in Game Design

- What is a Plugin?
  - A modular extension that adds functionality to game Engine, such as Unreal, Unity, Valve, etc.
  - Allows developers to integrate third-party tools and APIs
- Why Plugins Matter:
  - Extend core engine capabilities without modifying source code
  - Promote modularity and reusability across different projects
  - Facilitate easy integration of third-party APIs and functionalities

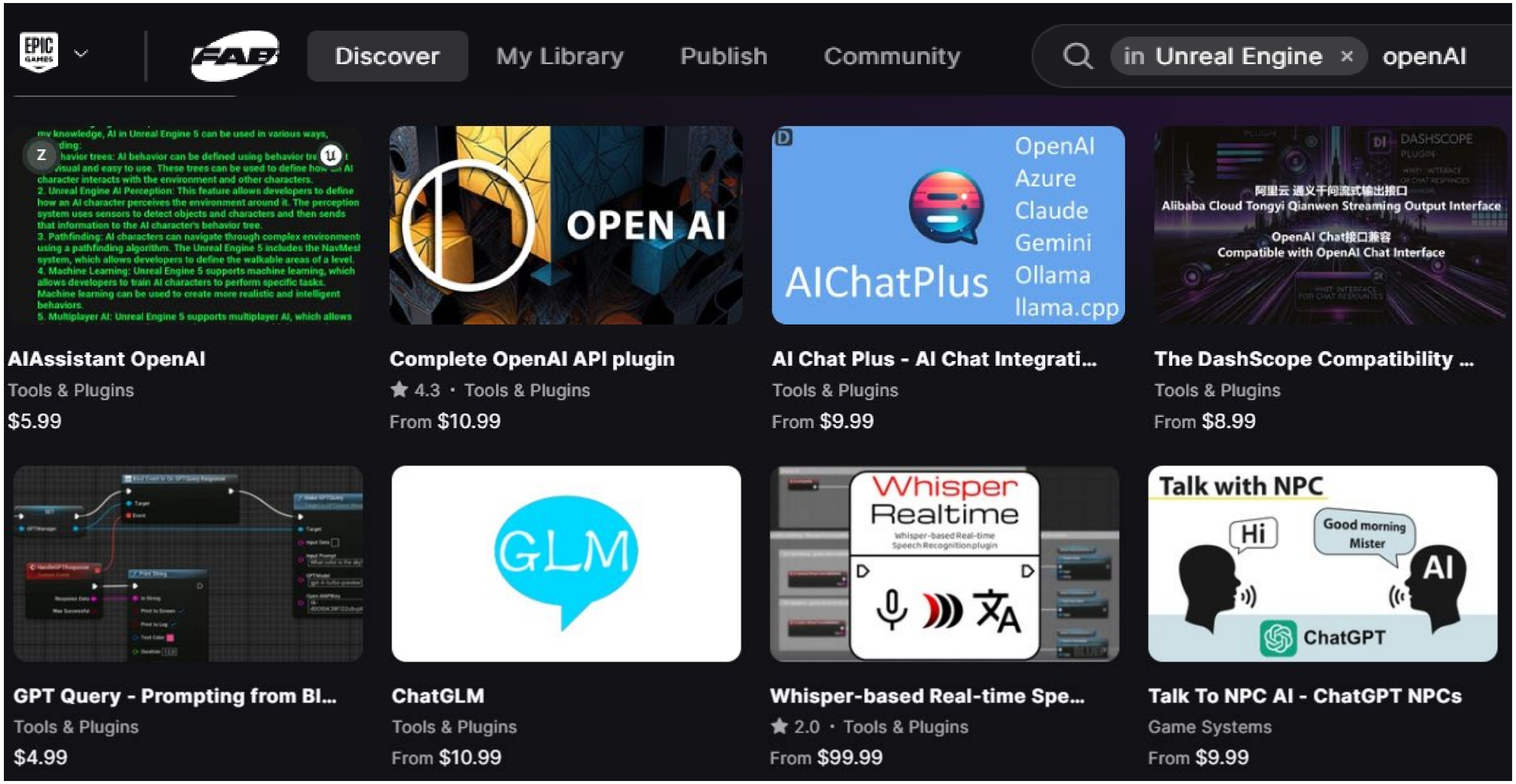
New York is so amazing have you never played a video game before
Grand Theft Auto V

- AI-based NPCs
- Physics and rendering enhancements
- Advanced multiplayer networking

What Do You Think?

Can we get it free?

Can we have a possible passive income via it?



my knowledge, AI in Unreal Engine 5 can be used in various ways, ding:
havior trees: AI behavior can be defined using behavior tre visual and easy to use. These trees can be used to define how AI character interacts with the environment and other characters.
2. Unreal Engine AI Perception: This feature allows developers to define how an AI character perceives the environment around it. The perception system uses sensors to detect objects and characters and then sends that information to the AI character's behavior tree.
3. Pathfinding: AI characters can navigate through complex environment using a pathfinding algorithm. The Unreal Engine 5 includes the NavMesl system, which allows developers to define the walkable areas of a level.
4. Machine Learning: Unreal Engine 5 supports machine learning, which allows developers to train AI characters to perform specific tasks. Machine learning can be used to create more realistic and intelligent behaviors.
5. Multiplayer AI: Unreal Engine 5 supports multiplayer AI, which allows

AIAssistant OpenAI
Tools & Plugins
$5.99

Complete OpenAI API plugin
★ 4.3 · Tools & Plugins
From $10.99

AI Chat Plus - AI Chat Integrati...
Tools & Plugins
From $9.99

The DashScope Compatibility ...
Tools & Plugins
From $8.99

GPT Query - Prompting from Bl...
Tools & Plugins
$4.99

ChatGLM
Tools & Plugins
From $10.99

Whisper-based Real-time Spe...
★ 2.0 · Tools & Plugins
From $99.99

Talk To NPC AI - ChatGPT NPCs
Game Systems
From $9.99

Learning Objectives

- ❑ Understand the role of plugins in game design
- ❑ Learn to set up, develop, and integrate OpenAI API into a plugin
- ❑ Debug, test, and optimize AI-driven interactions in games

Learning Outcomes

- ❑ Can Explain Unreal Engine plugin architecture
- ❑ Can Set up a plugin development environment
- ❑ Can Develop and integrate an OpenAI-powered plugin

Repository in Github

https://github.com/roboticsjoe2020/OpenAIAPI

git clone https://github.com/roboticsjoe2020/OpenAIAPI *yourprojectfolder*

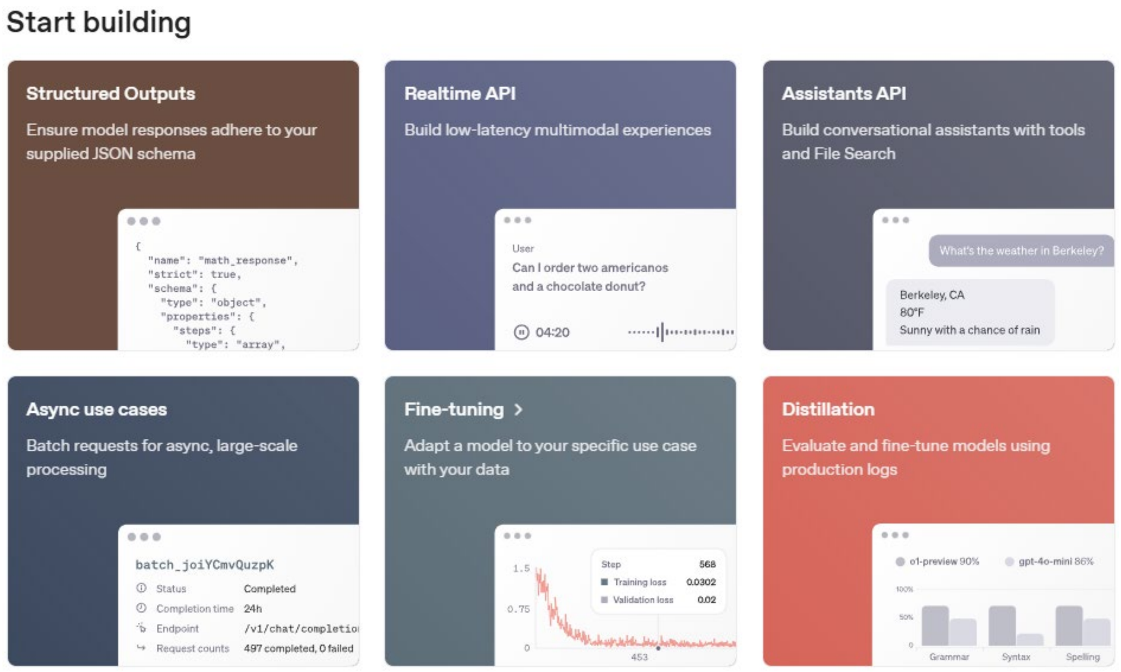
Or directly download

Endpoint: https://api.openai.com/v1/completions

***API Key**:*

OpenAI API

- OpenAI API capabilities:
  - Text generation (ChatGPT, GPT models)
  - Image generation (DALL-E)
  - Code generation (Codex)
  - Speech-to-text (Whisper)
- Potential applications:
  - AI-driven NPCs
  - Dynamic story generation
  - Procedural content creation



Start building

Structured Outputs

Ensure model responses adhere to your supplied JSON schema

```
{
  "name": "math_response",
  "strict": true,
  "schema": {
    "type": "object",
    "properties": {
      "steps": {
        "type": "array",
```

Realtime API

Build low-latency multimodal experiences

User

Can I order two americanos and a chocolate donut?

04:20

Assistants API

Build conversational assistants with tools and File Search

What's the weather in Berkeley?

Berkeley, CA
80°F
Sunny with a chance of rain

Async use cases

Batch requests for async, large-scale processing

```
batch_joiYCmvQuzpK
```

Status	Completed
Completion time	24h
Endpoint	/v1/chat/completio
Request counts	497 completed, 0 failed

Fine-tuning >

Adapt a model to your specific use case with your data

Step	568
Training loss	0.0302
Validation loss	0.02

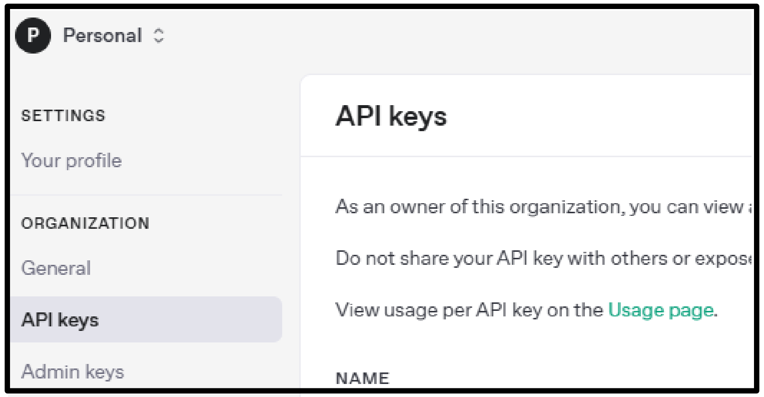
Distillation

Evaluate and fine-tune models using production logs

Setting Up Development Environment

- ❑ Requirements:

Unreal Engine (Latest Version)



P Personal

SETTINGS

Your profile

ORGANIZATION

General

API keys

Admin keys

API keys

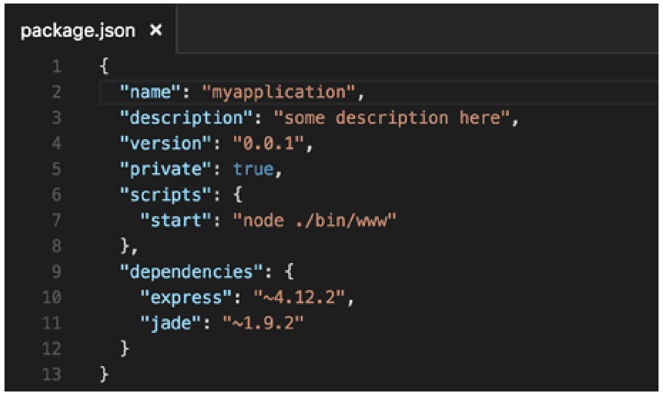
As an owner of this organization, you can view

Do not share your API key with others or expose

View usage per API key on the Usage page.

NAME

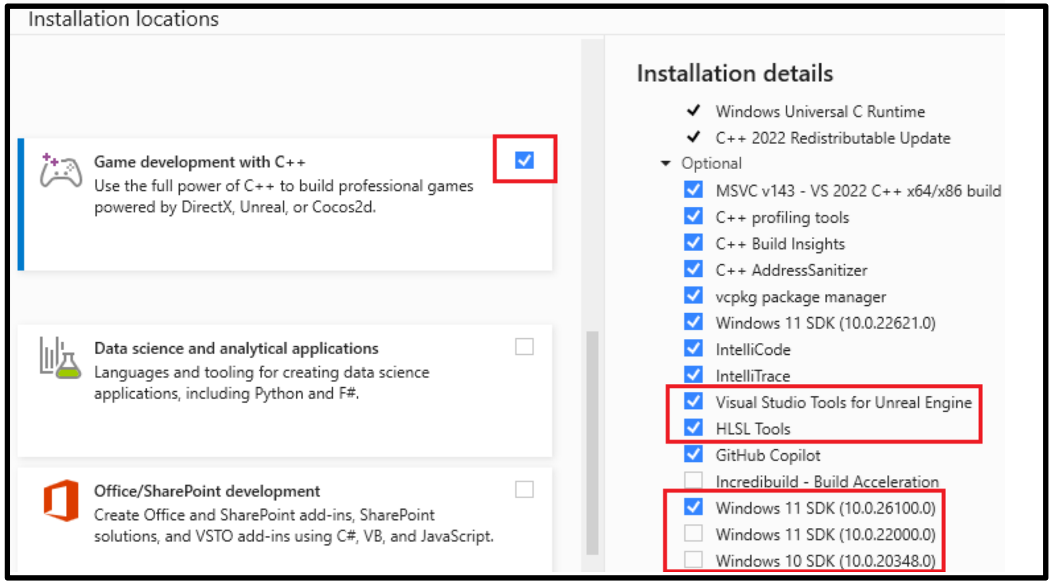
OpenAI API Key



package.json

```
{
  "name": "myapplication",
  "description": "some description here",
  "version": "0.0.1",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "express": "~4.12.2",
    "jade": "~1.9.2"
  }
}
```

JSON Parsing Libraries (RapidJSON, etc.)



Installation locations

Game development with C++ [✓]
Use the full power of C++ to build professional games powered by DirectX, Unreal, or Cocos2d.

Data science and analytical applications
Languages and tooling for creating data science applications, including Python and F#.

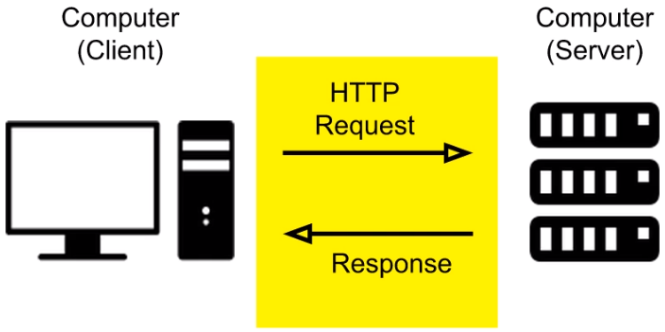
Office/SharePoint development
Create Office and SharePoint add-ins, SharePoint solutions, and VSTO add-ins using C#, VB, and JavaScript.

Installation details

- ✓ Windows Universal C Runtime
- ✓ C++ 2022 Redistributable Update
- Optional
  - [✓] MSVC v143 - VS 2022 C++ x64/x86 build
  - [✓] C++ profiling tools
  - [✓] C++ Build Insights
  - [✓] C++ AddressSanitizer
  - [✓] vcpkg package manager
  - [✓] Windows 11 SDK (10.0.22621.0)
  - [✓] IntelliCode
  - [✓] IntelliTrace
  - [✓] Visual Studio Tools for Unreal Engine
  - [✓] HLSL Tools
  - [✓] GitHub Copilot
  - [ ] Incredibuild - Build Acceleration
  - [✓] Windows 11 SDK (10.0.26100.0)
  - [ ] Windows 11 SDK (10.0.22000.0)
  - [ ] Windows 10 SDK (10.0.20348.0)

C++ Development Setup

https://learn.microsoft.com/en-us/visualstudio/gamedev/unreal/get-started/vs-tools-unreal-uproject



Network Access (HTTP Requests)

Other IDEs Recommendation

- Visual Studio Code (a lighter, more flexible alternative)

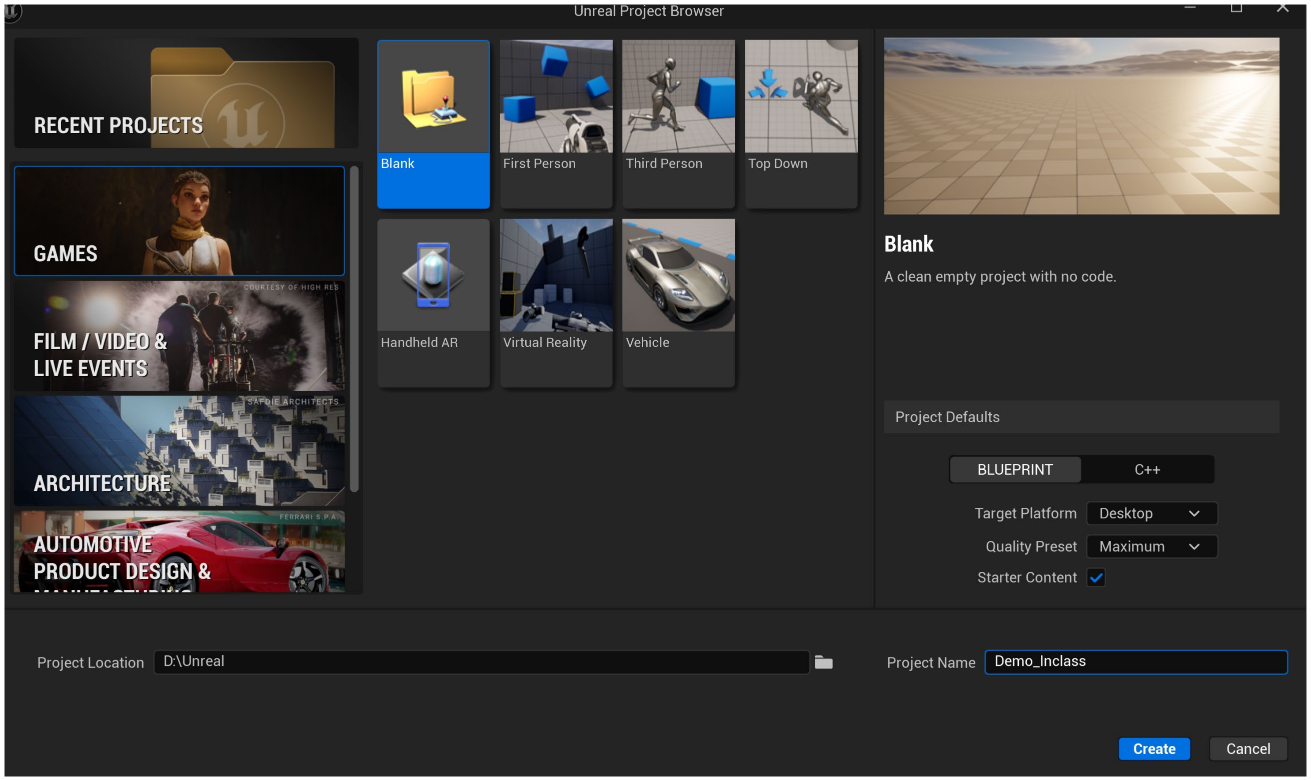
  https://dev.epicgames.com/documentation/en-us/unreal-engine/setting-up-visual-studio-code-for-unreal-engine

- Rider from JetBrain

  https://www.jetbrains.com/help/rider/Working_with_Unreal_Engine.html

Creating a Plugin in Unreal Engine

- Blueprint to create Full-fledged plugin?
  - Must have at least a minimal C++ plugin framework
  - Can expose functionality to Blueprints inside the plugin
  - Can create Blueprint function libraries within a plugin
  - Can create Blueprint assets that are distributed with the plugin.



Blank

A clean empty project with no code.

Project Defaults

[✓ BLUEPRINT] C++

Target Platform: Desktop

Quality Preset: Maximum

Starter Content: ✓

Project Location: D:\Unreal

Project Name: Demo_Inclass

Integrating OpenAI API into the Plugin

- Include HTTP module for API communication.
- Implement API request handling (POST requests with JSON payloads).
- Process OpenAI API responses.
- Convert AI-generated content for in-game use.

Example: Calling OpenAI API to generate NPC dialogue:

```
FHttpRequestPtr Request = FHttpModule::Get().CreateRequest();
Request->SetURL("https://api.openai.com/v1/completions");
Request->SetVerb("POST");
Request->SetHeader("Authorization", "Bearer YOUR_API_KEY");
Request->SetContentAsString(PayloadJsonString);
Request->OnProcessRequestComplete().BindUObject(this,
&YourClass::OnResponseReceived);
Request->ProcessRequest();
```

Handling API Responses in Unreal Engine

- Response Parsing:
  - JSON response handling with FJsonObject and FJsonReader
  - Extracting relevant text/image data

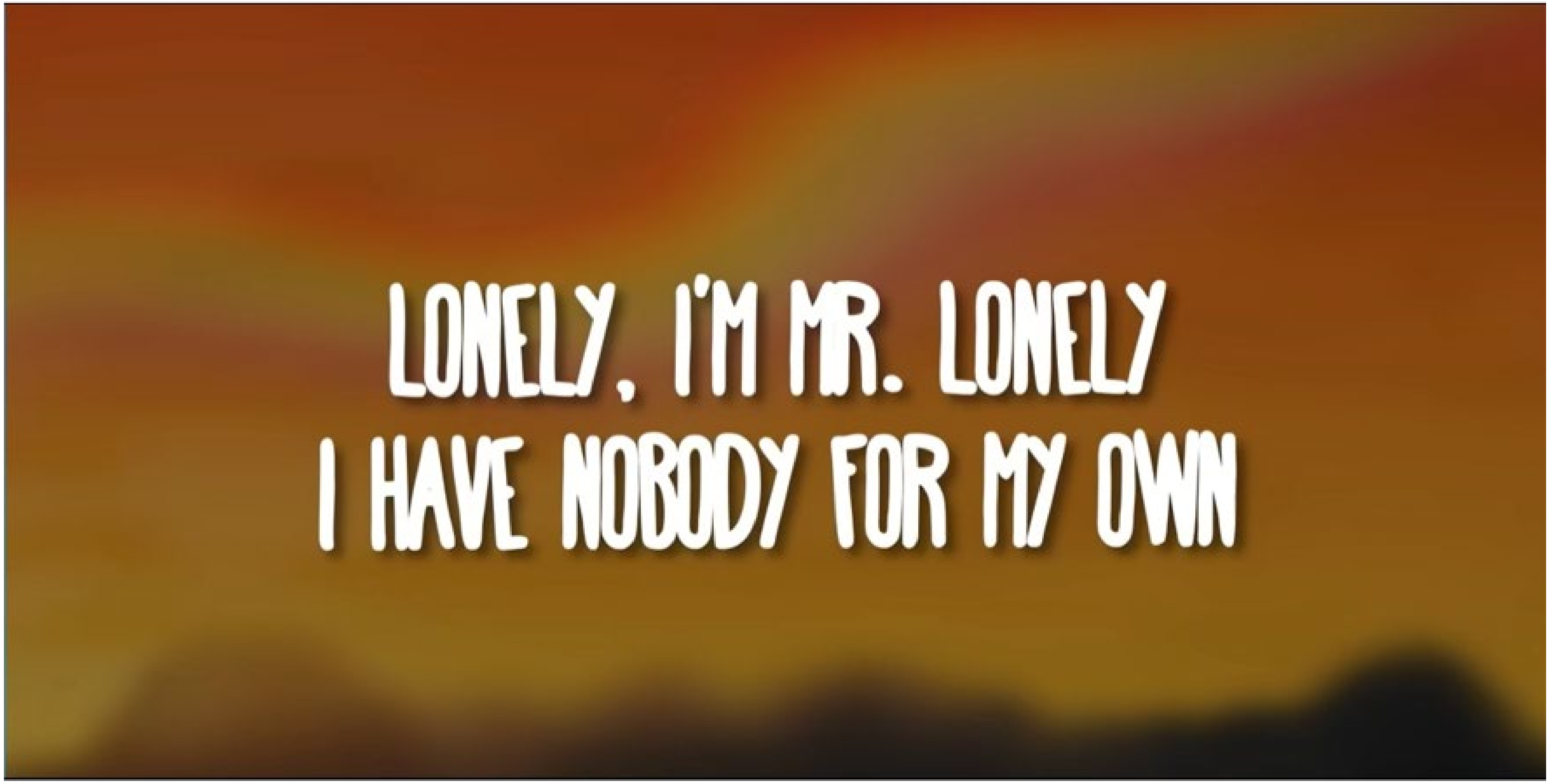
- Integrating into Gameplay:
  - Dynamic NPC dialogue
  - AI-assisted storytelling
  - Procedural environment descriptions

Testing & Debugging the Plugin

- Debugging API Calls:
  - Use UE_LOG to track API requests and responses
  - Validate JSON format before sending requests
  - Handle rate limits and errors from OpenAI API

- Testing in Unreal Engine:
  - Use Play Mode for real-time validation
  - Create test scenarios with controlled AI interactions

What Do You Get?


LONELY, I'M MR. LONELY
I HAVE NOBODY FOR MY OWN

Repository in Github

https://github.com/roboticsjoe2020/OpenAIAPI

git clone https://github.com/roboticsjoe2020/OpenAIAPI   *yourprojectfolder*

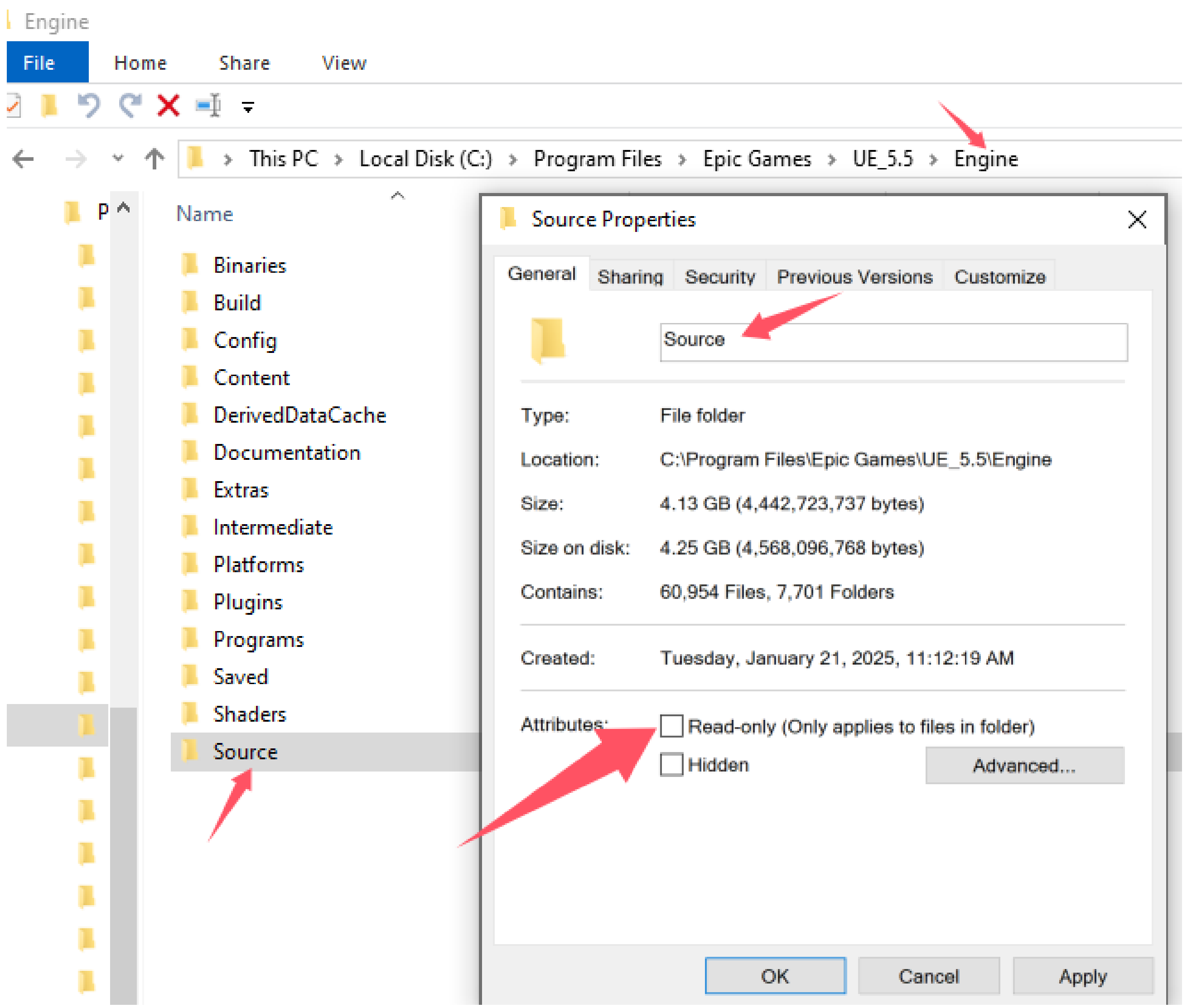
Or directly download

Endpoint: https://api.openai.com/v1/completions

API Key:

Practical Procedures (1)

Make Sure 'Source' Folder R/W



Source

Type: File folder

Location: C:\Program Files\Epic Games\UE_5.5\Engine

Size: 4.13 GB (4,442,723,737 bytes)

Size on disk: 4.25 GB (4,568,096,768 bytes)

Contains: 60,954 Files, 7,701 Folders

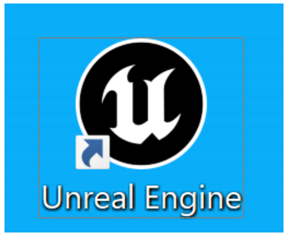
Created: Tuesday, January 21, 2025, 11:12:19 AM

Attributes: ☐ Read-only (Only applies to files in folder)

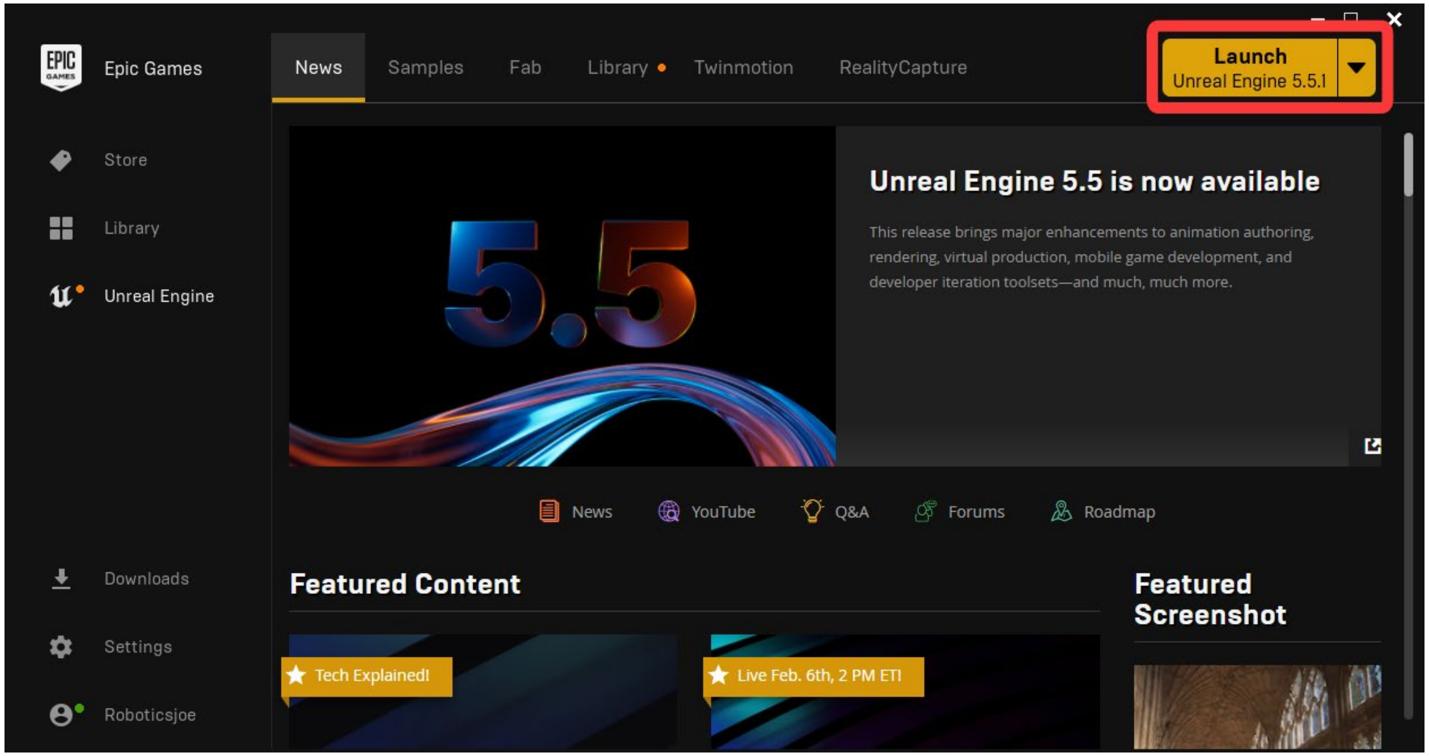
☐ Hidden

Practical Procedures (2)

Directly Launch Unreal Engine



Or Through Epic Game Launcher



Unreal Engine 5.5 is now available

This release brings major enhancements to animation authoring, rendering, virtual production, mobile game development, and developer iteration toolsets—and much, much more.

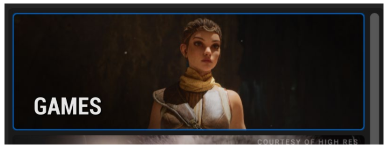
Featured Content

Featured Screenshot

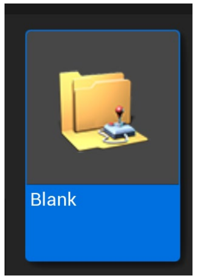
Practical Procedures (3)

Create a New C++ Project:

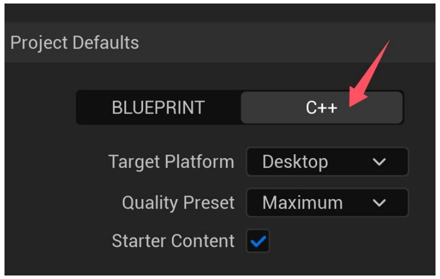
→Click 'GAMES'



→ Choose 'Blank

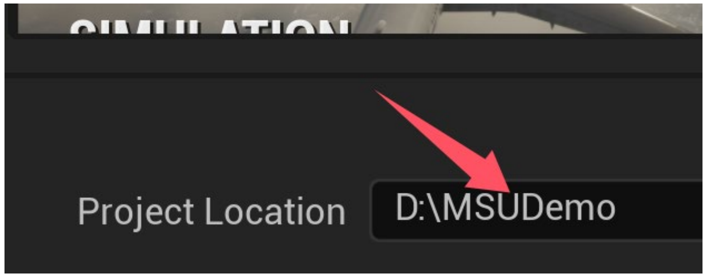


→Choose C++ as the template



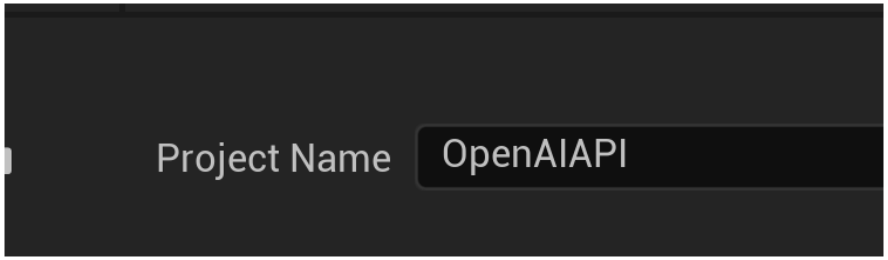
Target Platform: Desktop
Quality Preset: Maximum
Starter Content: ✓

→Project Location (Your Local Drive)



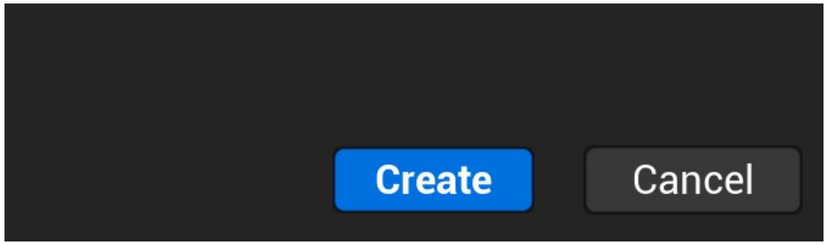
Project Location: D:\MSUDemo

→Name your project (e.g., OpenAIAPI).



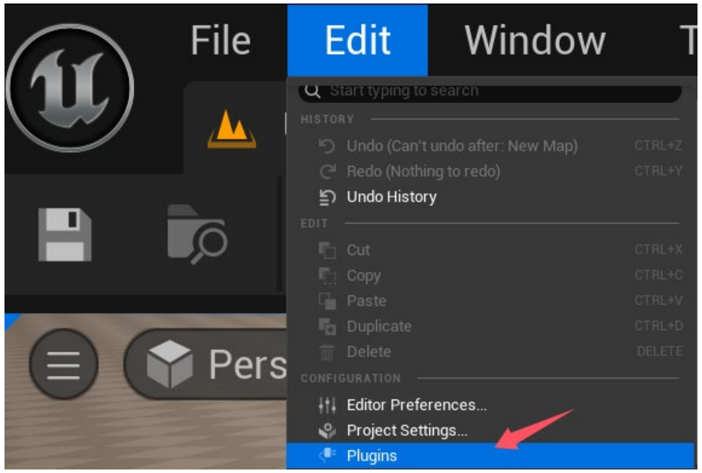
Project Name: OpenAIAPI

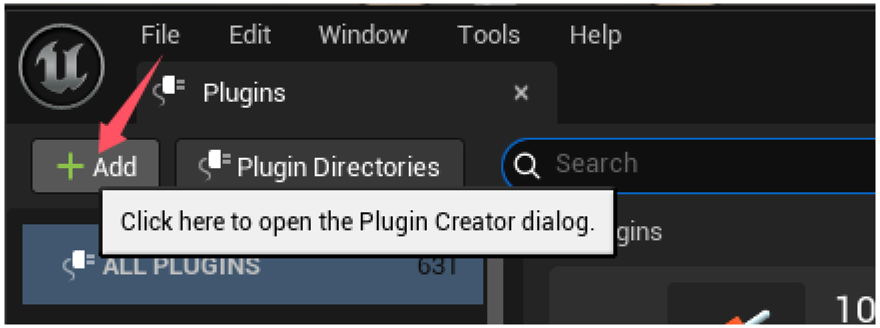
→Click 'Create'.



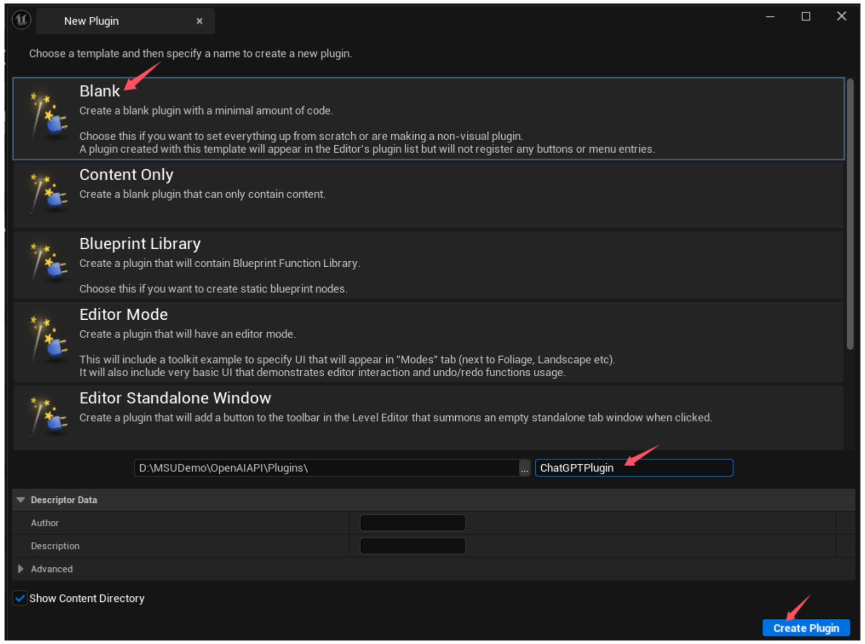
Practical Procedures (4)

Create a New C++ plugin in Unreal editor





Click here to open the Plugin Creator dialog.



Choose a template and then specify a name to create a new plugin.

Blank
Create a blank plugin with a minimal amount of code.
Choose this if you want to set everything up from scratch or are making a non-visual plugin.
A plugin created with this template will appear in the Editor's plugin list but will not register any buttons or menu entries.

Content Only
Create a blank plugin that can only contain content.

Blueprint Library
Create a plugin that will contain Blueprint Function Library.
Choose this if you want to create static blueprint nodes.

Editor Mode
Create a plugin that will have an editor mode.
This will include a toolkit example to specify UI that will appear in "Modes" tab (next to Foliage, Landscape etc).
It will also include very basic UI that demonstrates editor interaction and undo/redo functions usage.

Editor Standalone Window
Create a plugin that will add a button to the toolbar in the Level Editor that summons an empty standalone tab window when clicked.

D:\MSUDemo\OpenAIAPI\Plugins\ ChatGPTPlugin

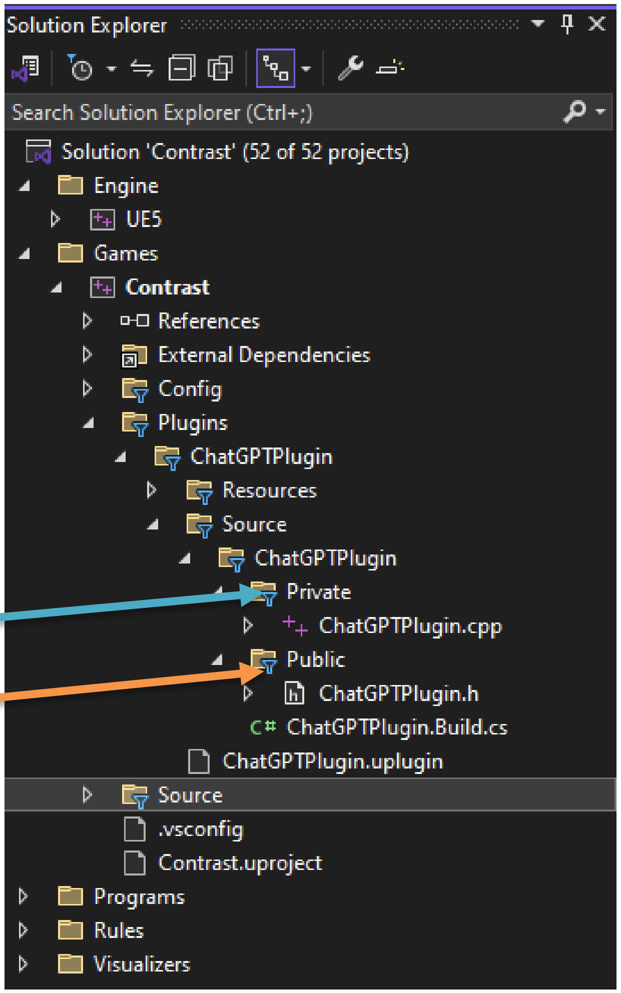
Descriptor Data
Author
Description
Advanced

✓ Show Content Directory

Practical Procedures (5)

After creating the plugin, verify the folder structure:

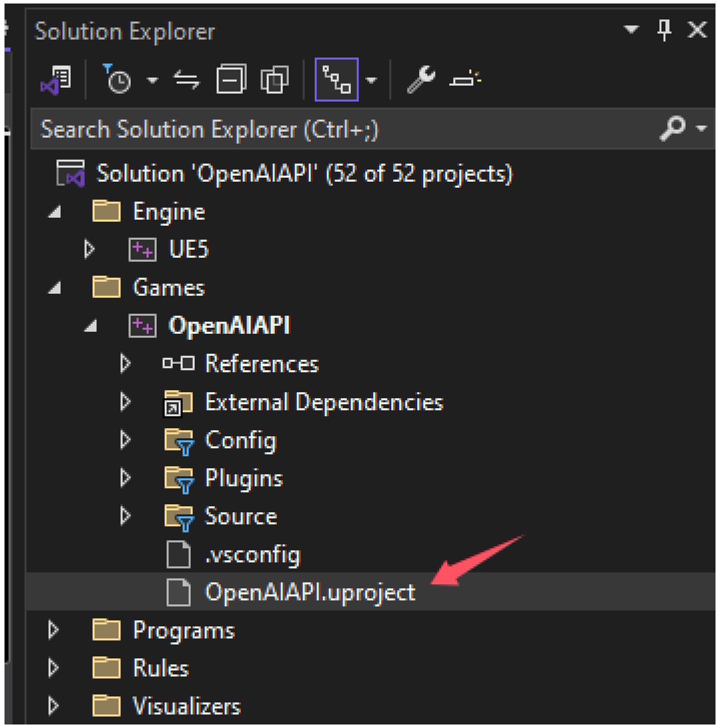
```
OpenAIAPI/
├── Plugins/
│   └── ChatGPTPlugin/
│       ├── Resources/
│       ├── Source/
│       │   └── ChatGPTPlugin/
│       │       ├── Private/
│       │       ├── Public/
│       │       └── ChatGPTPlugin.Build.cs
│       └── ChatGPTPlugin.uplugin
```



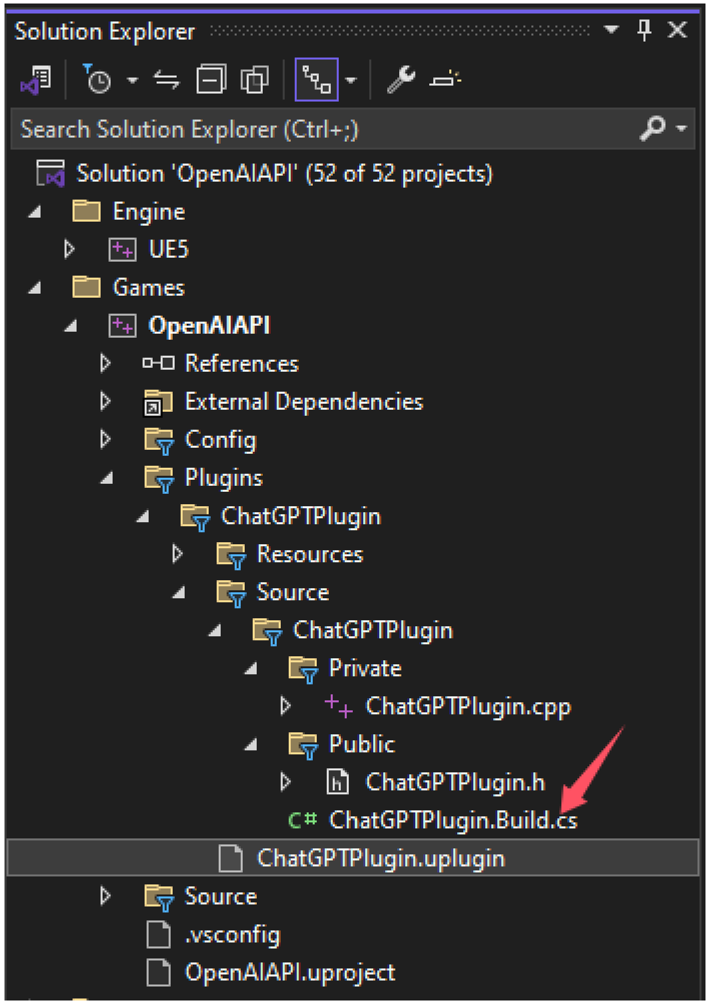
```
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'Contrast' (52 of 52 projects)
  Engine
    UE5
  Games
    Contrast
      References
      External Dependencies
      Config
      Plugins
        ChatGPTPlugin
          Resources
          Source
            ChatGPTPlugin
              Private
                ChatGPTPlugin.cpp
              Public
                ChatGPTPlugin.h
              ChatGPTPlugin.Build.cs
          ChatGPTPlugin.uplugin
      Source
      .vsconfig
      Contrast.uproject
  Programs
  Rules
  Visualizers
```

Practical Procedures (6)

Go to Visual Studio and open project and define the project and plugin metadata



```
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'OpenAIAPI' (52 of 52 projects)
  Engine
    UE5
  Games
    OpenAIAPI
      References
      External Dependencies
      Config
      Plugins
      Source
      .vsconfig
      OpenAIAPI.uproject
  Programs
  Rules
  Visualizers
```

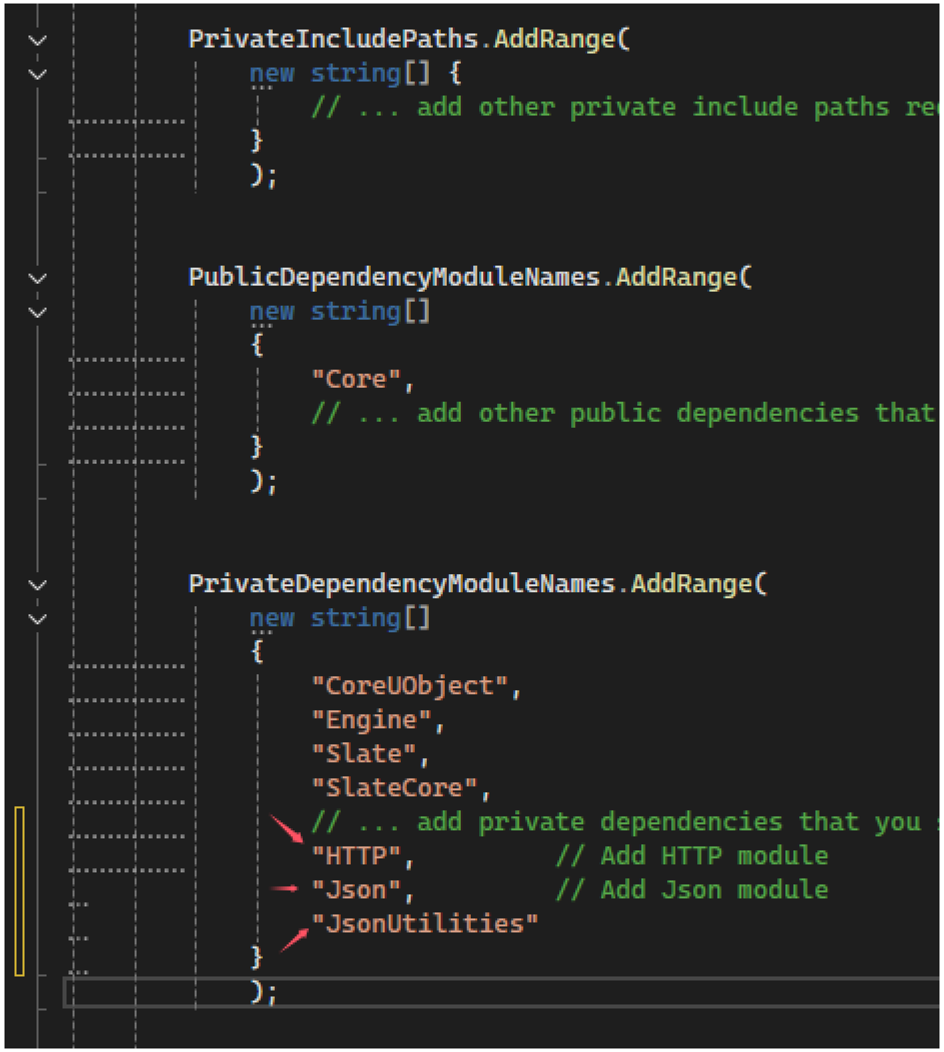


```
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'OpenAIAPI' (52 of 52 projects)
  Engine
    UE5
  Games
    OpenAIAPI
      References
      External Dependencies
      Config
      Plugins
        ChatGPTPlugin
          Resources
          Source
            ChatGPTPlugin
              Private
                ChatGPTPlugin.cpp
              Public
                ChatGPTPlugin.h
              ChatGPTPlugin.Build.cs
          ChatGPTPlugin.uplugin
      Source
      .vsconfig
      OpenAIAPI.uproject
```

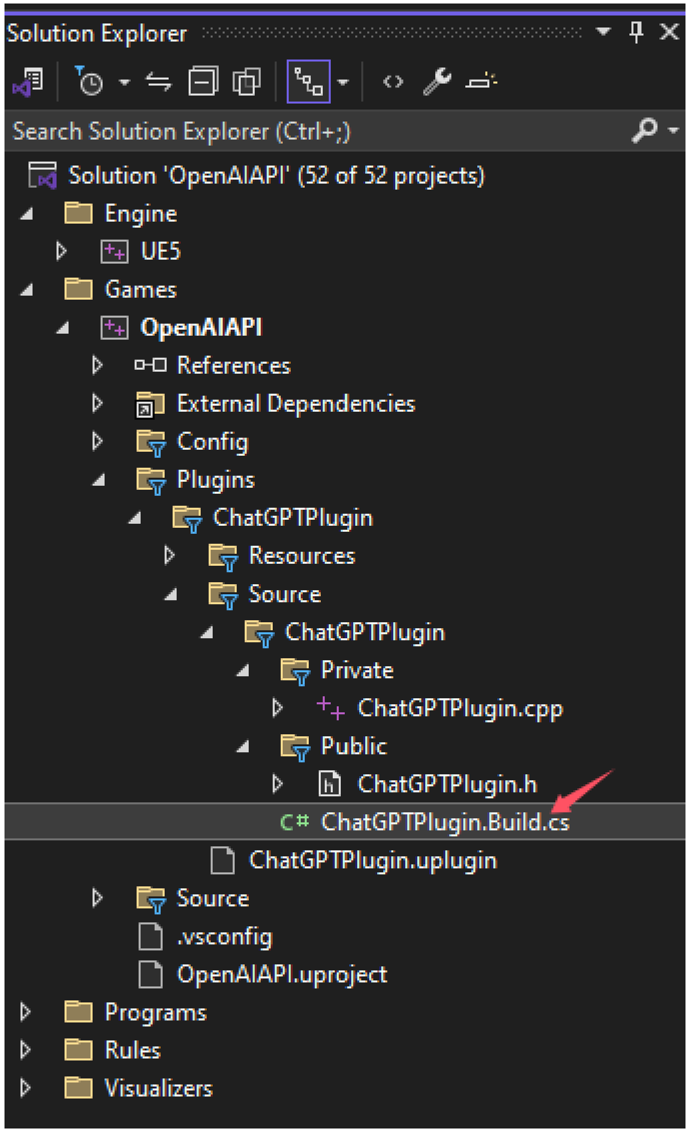
Use default setup for both

Practical Procedures (7)

Add required dependencies (Crucial Step) in ChatGPTPlugin.Build.cs



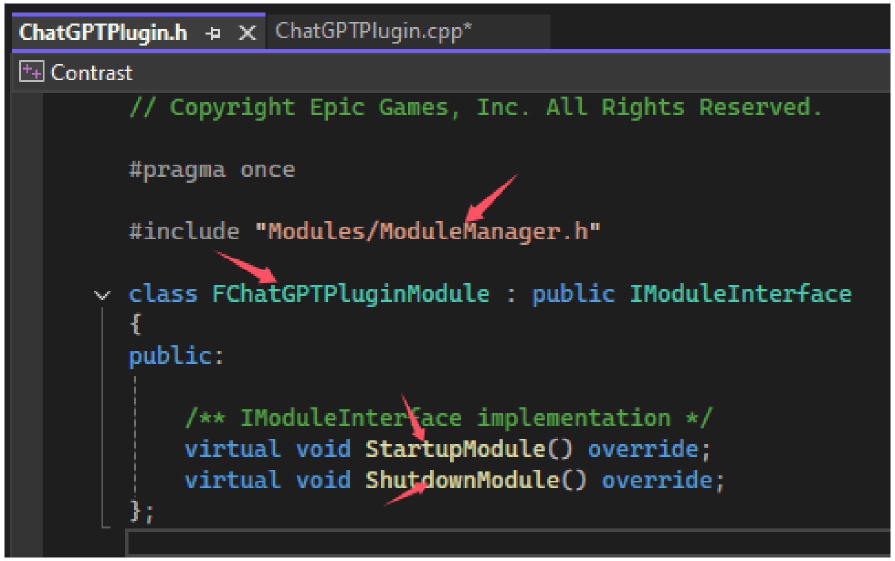
```
PrivateIncludePaths.AddRange(
    new string[] {
        // ... add other private include paths re
    }
    );


PublicDependencyModuleNames.AddRange(
    new string[]
    {
        "Core",
        // ... add other public dependencies that
    }
    );


PrivateDependencyModuleNames.AddRange(
    new string[]
    {
        "CoreUObject",
        "Engine",
        "Slate",
        "SlateCore",
        // ... add private dependencies that you 
        "HTTP",          // Add HTTP module
        "Json",          // Add Json module
        "JsonUtilities"
    }
    );
```



```
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'OpenAIAPI' (52 of 52 projects)
  Engine
    UE5
  Games
    OpenAIAPI
      References
      External Dependencies
      Config
      Plugins
        ChatGPTPlugin
          Resources
          Source
            ChatGPTPlugin
              Private
                ChatGPTPlugin.cpp
              Public
                ChatGPTPlugin.h
              ChatGPTPlugin.Build.cs
          ChatGPTPlugin.uplugin
      Source
      .vsconfig
      OpenAIAPI.uproject
  Programs
  Rules
  Visualizers
```

Practical Procedures (8)

How Unreal Engine Manages Modules: Header?



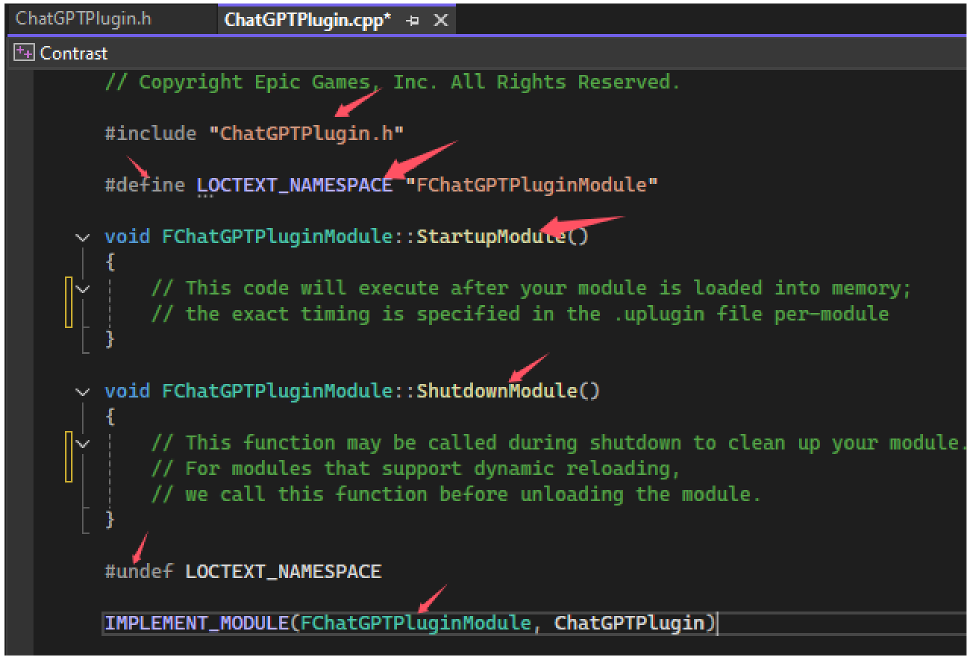
```
// Copyright Epic Games, Inc. All Rights Reserved.

#pragma once

#include "Modules/ModuleManager.h"

class FChatGPTPluginModule : public IModuleInterface
{
public:

    /** IModuleInterface implementation */
    virtual void StartupModule() override;
    virtual void ShutdownModule() override;
};
```

- ❑ Unreal Engine, modules (like FChatGPTPluginModule) managed by the module system (IModuleInterface). Engine starts → automatically loads and initializes all registered modules.
  - The declaration (ChatGPTPlugin.h) is part of the module header.
  - The implementation (ChatGPTPlugin.cpp) is instantiated at runtime by Unreal Engine.
  - StartupModdule() and ShutdownModule declared here.

- ❑ ***Keypoints***
  - *'FChatGPTPluginModule' declared normally as a class.*
  - *No instance created here.*
  - *Unreal Engine will instantiate this module at runtime and manage its lifecycle.*

Practical Procedures (9)

How Unreal Engine Manages Modules: Implementation file?



```
// Copyright Epic Games, Inc. All Rights Reserved.

#include "ChatGPTPlugin.h"

#define LOCTEXT_NAMESPACE "FChatGPTPluginModule"

void FChatGPTPluginModule::StartupModule()
{
	// This code will execute after your module is loaded into memory;
	// the exact timing is specified in the .uplugin file per-module
}

void FChatGPTPluginModule::ShutdownModule()
{
	// This function may be called during shutdown to clean up your module.
	// For modules that support dynamic reloading,
	// we call this function before unloading the module.
}

#undef LOCTEXT_NAMESPACE

IMPLEMENT_MODULE(FChatGPTPluginModule, ChatGPTPlugin)
```

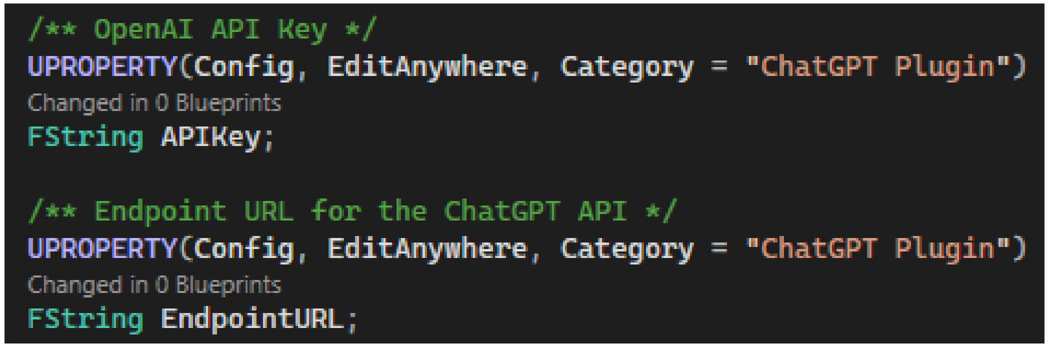
- ❑ Defines the module and registers it using Unreal's *IMPLEMENT_MODULE* macro
- ❑ ***Keypoints***
  - *IMPLEMENT_MODULE(FChatGPTPluginModule, ChatGPTPlugin) macro tells Unreal Engine:*
    - *"This is the module class for the plugin.“*
    - *"Instantiate it at runtime when Unreal Engine loads the plugin."*
  - *FChatGPTPluginModule is not manually instantiated in ChatGPTPlugin.h or ChatGPTPlugin.cpp because Unreal Engine does this automatically.*
  - *Function FModuleManager::Get().LoadModuleChecked<FChatGPTPluginModule> ("ChatGPTPlugin") can be used to get a reference to the instantiated module if needed.*

Practical Procedures (10)

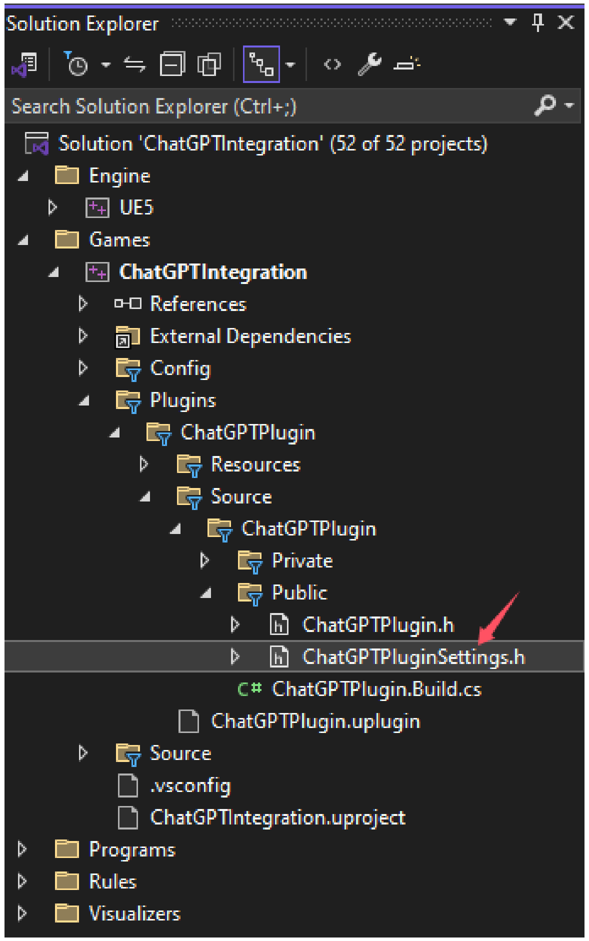
Implement core functionality in C++: header- ChatGPTPluginSettings.h (1)

ChatGPT Plugin Settings for API configuration:

- Dynamic API Key and Endpoint: Allows for runtime configuration through Unreal Editor's ***Project Settings***.
- Error Handling: Clear error messages for missing or invalid API keys and endpoints.
- Modular Parsing: Separate ***ParseResponse*** for clean JSON parsing.
- Performance: Minimized memory overhead by relying on settings when needed.



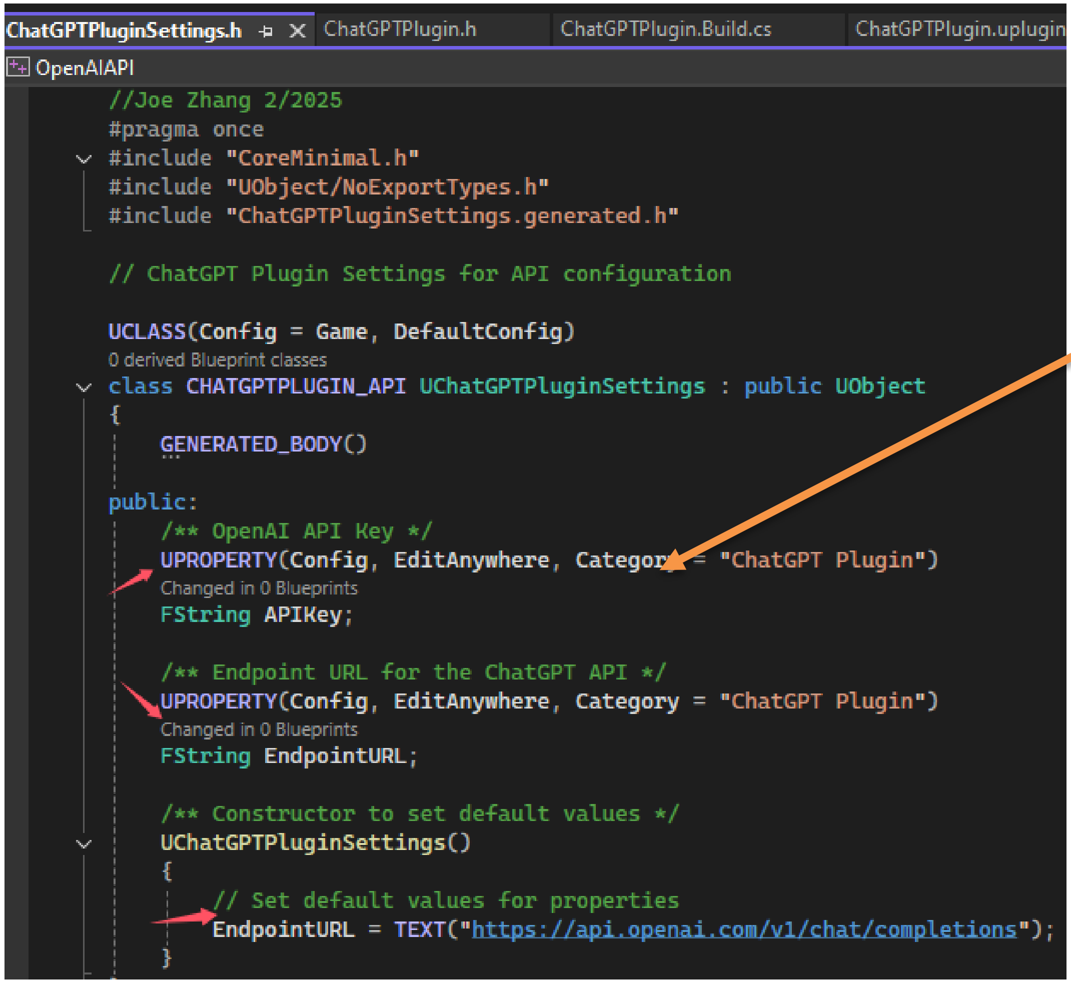
```
/** OpenAI API Key */
UPROPERTY(Config, EditAnywhere, Category = "ChatGPT Plugin")
Changed in 0 Blueprints
FString APIKey;

/** Endpoint URL for the ChatGPT API */
UPROPERTY(Config, EditAnywhere, Category = "ChatGPT Plugin")
Changed in 0 Blueprints
FString EndpointURL;
```



```
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'ChatGPTIntegration' (52 of 52 projects)
  Engine
    UE5
  Games
    ChatGPTIntegration
      References
      External Dependencies
      Config
      Plugins
        ChatGPTPlugin
          Resources
          Source
            ChatGPTPlugin
              Private
              Public
                ChatGPTPlugin.h
                ChatGPTPluginSettings.h
              ChatGPTPlugin.Build.cs
          ChatGPTPlugin.uplugin
      Source
      .vsconfig
      ChatGPTIntegration.uproject
  Programs
  Rules
  Visualizers
```

Practical Procedures (11)

Implement core functionality in C++: header- ChatGPTPluginSettings.h (2)



```
//Joe Zhang 2/2025
#pragma once
#include "CoreMinimal.h"
#include "UObject/NoExportTypes.h"
#include "ChatGPTPluginSettings.generated.h"

// ChatGPT Plugin Settings for API configuration

UCLASS(Config = Game, DefaultConfig)
0 derived Blueprint classes
class CHATGPTPLUGIN_API UChatGPTPluginSettings : public UObject
{
    GENERATED_BODY()

public:
    /** OpenAI API Key */
    UPROPERTY(Config, EditAnywhere, Category = "ChatGPT Plugin")
    Changed in 0 Blueprints
    FString APIKey;

    /** Endpoint URL for the ChatGPT API */
    UPROPERTY(Config, EditAnywhere, Category = "ChatGPT Plugin")
    Changed in 0 Blueprints
    FString EndpointURL;

    /** Constructor to set default values */
    UChatGPTPluginSettings()
    {
        // Set default values for properties
        EndpointURL = TEXT("https://api.openai.com/v1/chat/completions");
    }
```

UPROPERTY(Config, EditAnywhere, Category="ChatGPT Plugin"):

- The property will appear under a category labeled "***ChatGPT Plugin***" in the details panel.
- Best for grouping all plugin-related settings under a "ChatGPT Plugin" heading.

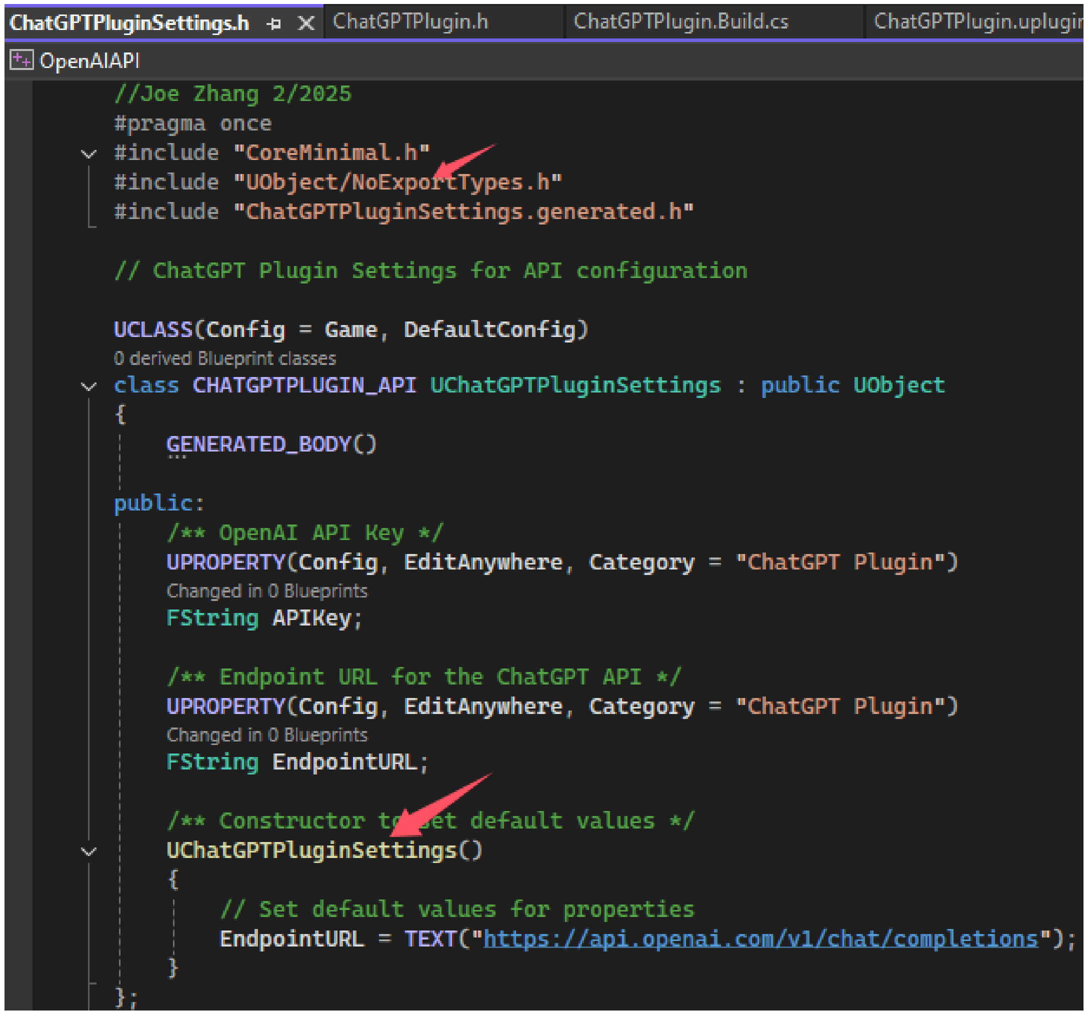
EditAnywhere: Enable visible and editable in the details panel in Unreal Editor;

Config: property should be loaded from or saved to a configuration file (e.g., DefaultGame.ini) and changes to this property in the Unreal Editor are persisted in the configuration file.

- Category: Defines how the property is organized in the details panel, and properties with the same category are grouped together in the editor.

Practical Procedures (12)

Implement core functionality in C++: header- ChatGPTPluginSettings.h (3)



```cpp
//Joe Zhang 2/2025
#pragma once
#include "CoreMinimal.h"
#include "UObject/NoExportTypes.h"
#include "ChatGPTPluginSettings.generated.h"

// ChatGPT Plugin Settings for API configuration

UCLASS(Config = Game, DefaultConfig)
class CHATGPTPLUGIN_API UChatGPTPluginSettings : public UObject
{
    GENERATED_BODY()

public:
    /** OpenAI API Key */
    UPROPERTY(Config, EditAnywhere, Category = "ChatGPT Plugin")
    FString APIKey;

    /** Endpoint URL for the ChatGPT API */
    UPROPERTY(Config, EditAnywhere, Category = "ChatGPT Plugin")
    FString EndpointURL;

    /** Constructor to set default values */
    UChatGPTPluginSettings()
    {
        // Set default values for properties
        EndpointURL = TEXT("https://api.openai.com/v1/chat/completions");
    }
};
```

Why Use the Constructor?

- ***Compatibility***: Unreal Engine automatically calls the constructor when creating instances of *UObject*-derived classes, ensuring your default values are properly initialized.
- ***Reflection safety***: Properties managed by UPROPERTY must adhere to Unreal's property system. Direct initialization in the header is not guaranteed to work properly with reflection and serialization.
- ***Ease of future modifications***: If you need to adjust the default value, it's centralized in the constructor, reducing potential conflicts or overlooked changes.

Practical Procedures (13)

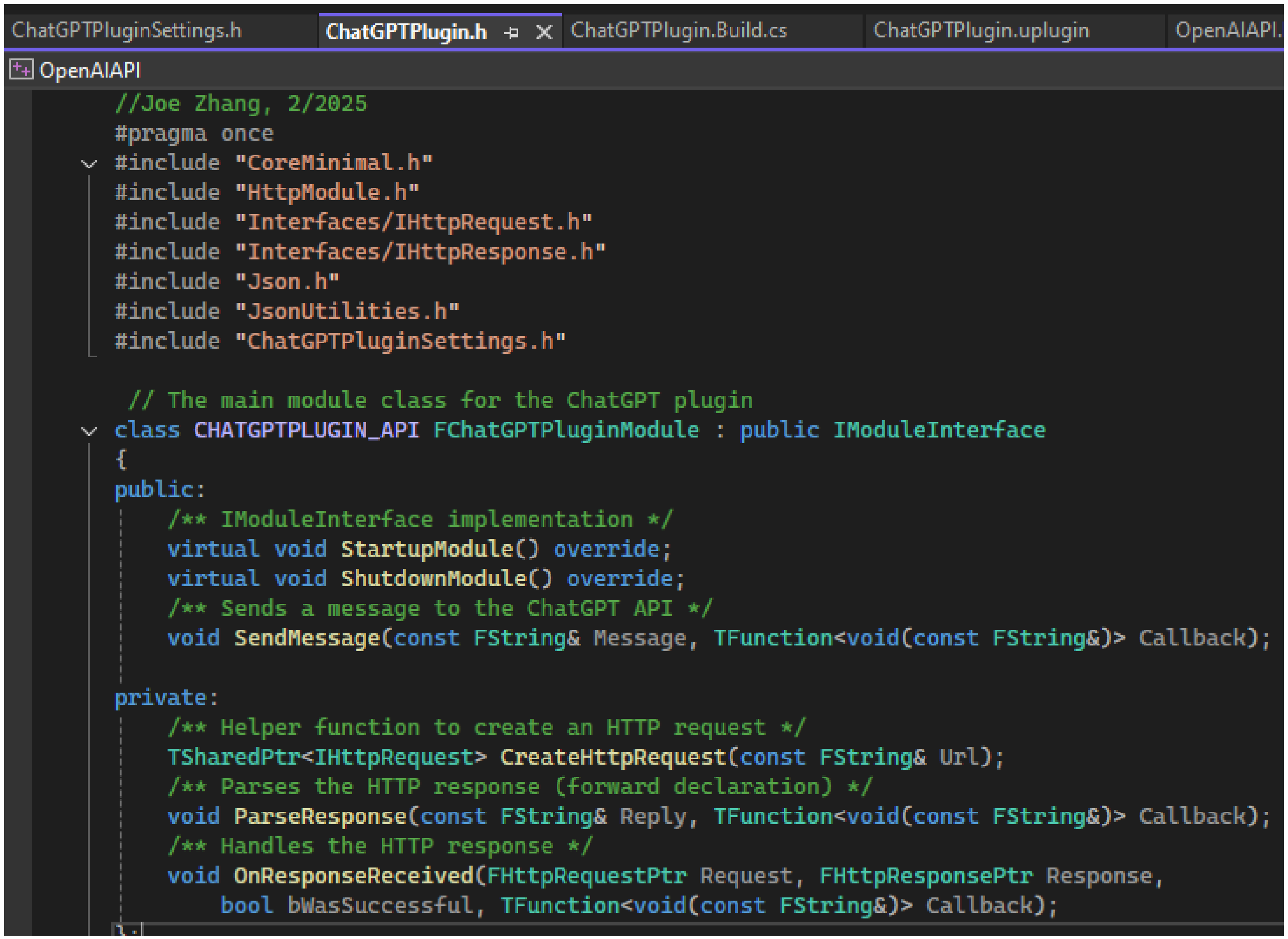
Core functionality in C++: header- ChatGPTPlugin.h (1)

Function Name	**Purpose**
StartupModule()	Initializes the plugin when loaded.
ShutdownModule()	Cleans up resources when the plugin is unloaded.
SendMessage()	Sends a user query to ChatGPT and handles the response.
CreateHttpRequest()	Creates and configures an HTTP request for the API.
OnResponseReceived()	Processes the API's HTTP response and triggers the callback.
ParseResponse()	Extracts useful data from the API's JSON response.

- Public functions (StartupModule, ShutdownModule, SendMessage) handle initialization, shutdown, and API interaction.
- Private functions (CreateHttpRequest, OnResponseReceived, ParseResponse) manage HTTP communication and response parsing.
- Ensures a ***modular, structured, and efficient API*** interaction within Unreal Engine.

Practical Procedures (14)

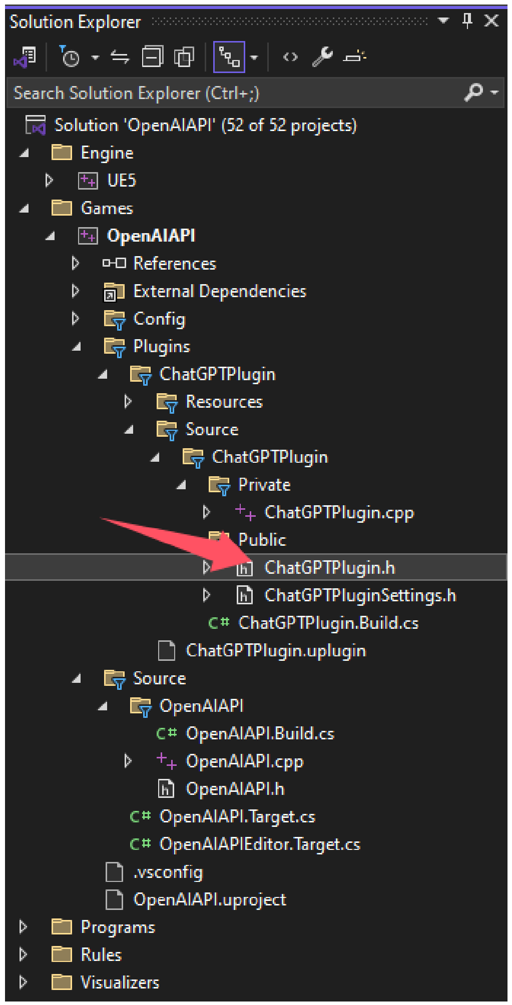
Core functionality in C++: header- ChatGPTPlugin.h (2)



ChatGPTPluginSettings.h | ChatGPTPlugin.h | ChatGPTPlugin.Build.cs | ChatGPTPlugin.uplugin | OpenAIAPI.

OpenAIAPI

```cpp
//Joe Zhang, 2/2025
#pragma once
#include "CoreMinimal.h"
#include "HttpModule.h"
#include "Interfaces/IHttpRequest.h"
#include "Interfaces/IHttpResponse.h"
#include "Json.h"
#include "JsonUtilities.h"
#include "ChatGPTPluginSettings.h"

 // The main module class for the ChatGPT plugin
class CHATGPTPLUGIN_API FChatGPTPluginModule : public IModuleInterface
{
public:
    /** IModuleInterface implementation */
    virtual void StartupModule() override;
    virtual void ShutdownModule() override;
    /** Sends a message to the ChatGPT API */
    void SendMessage(const FString& Message, TFunction<void(const FString&)> Callback);

private:
    /** Helper function to create an HTTP request */
    TSharedPtr<IHttpRequest> CreateHttpRequest(const FString& Url);
    /** Parses the HTTP response (forward declaration) */
    void ParseResponse(const FString& Reply, TFunction<void(const FString&)> Callback);
    /** Handles the HTTP response */
    void OnResponseReceived(FHttpRequestPtr Request, FHttpResponsePtr Response,
        bool bWasSuccessful, TFunction<void(const FString&)> Callback);
}
```



Solution Explorer

Search Solution Explorer (Ctrl+;)

```
Solution 'OpenAIAPI' (52 of 52 projects)
  Engine
    UE5
  Games
    OpenAIAPI
      References
      External Dependencies
      Config
      Plugins
        ChatGPTPlugin
          Resources
          Source
            ChatGPTPlugin
              Private
                ChatGPTPlugin.cpp
              Public
                ChatGPTPlugin.h
                ChatGPTPluginSettings.h
              ChatGPTPlugin.Build.cs
          ChatGPTPlugin.uplugin
      Source
        OpenAIAPI
          OpenAIAPI.Build.cs
          OpenAIAPI.cpp
          OpenAIAPI.h
        OpenAIAPI.Target.cs
        OpenAIAPIEditor.Target.cs
      .vsconfig
      OpenAIAPI.uproject
  Programs
  Rules
  Visualizers
```

Practical Procedures (15)

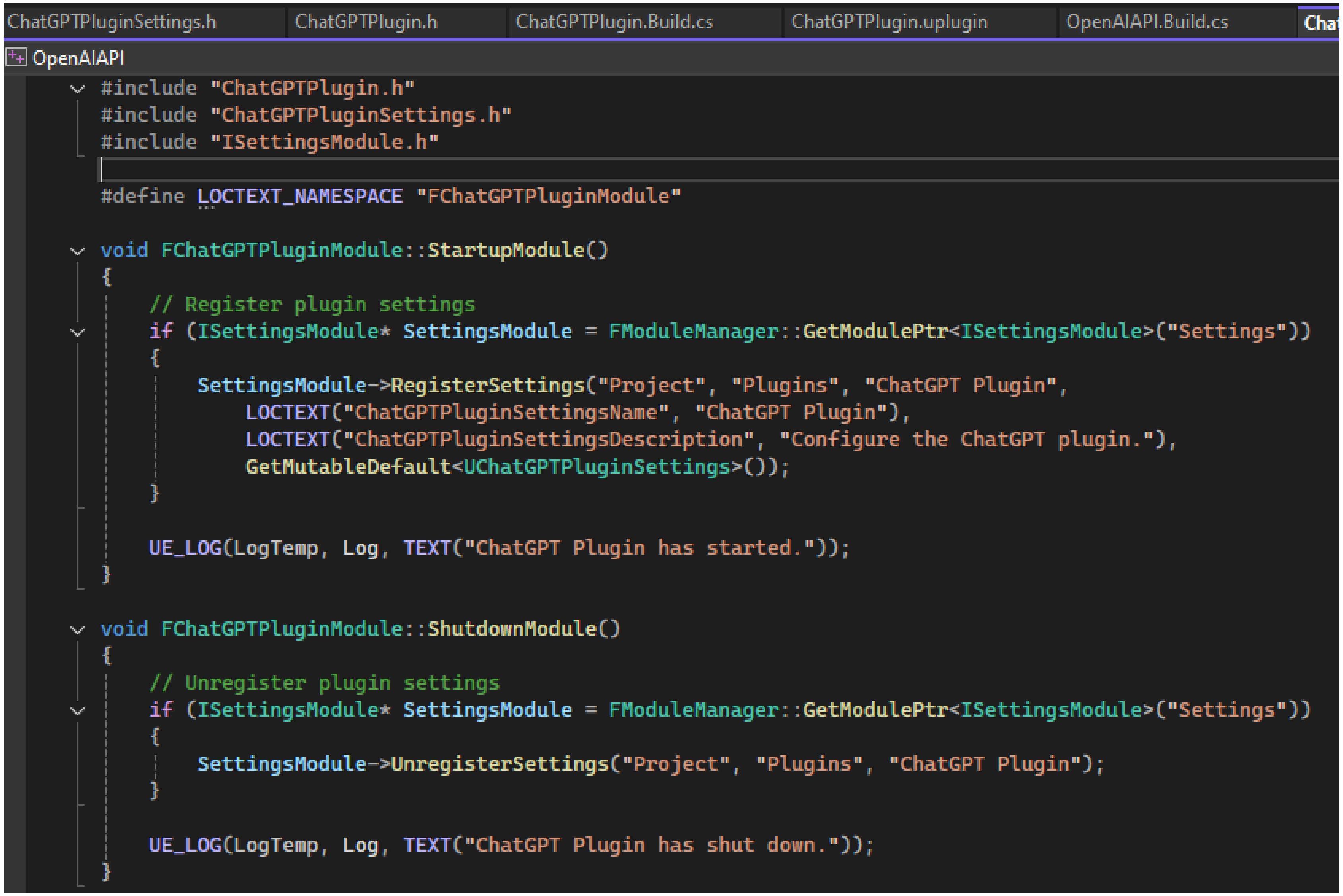
Implement core functionality in C++: Implementation file - ChatGPTPlugin.cpp (1)

Function Name	Purpose
StartupModule()	Initializes the plugin and registers settings.
ShutdownModule()	Cleans up the module when the plugin is unloaded.
SendMessage()	Retrieves the API key & endpoint, then sends an HTTP request.
CreateHttpRequest()	Creates and configures the HTTP request with headers.
OnResponseReceived()	Handles the HTTP response and passes data to ParseResponse().
ParseResponse()	Extracts relevant text from the JSON response and invokes the callback.

- StartupModule and ShutdownModule handle initialization and cleanup.
- SendMessage sends a request to ChatGPT.
- CreateHttpRequest configures HTTP requests.
- OnResponseReceived processes API responses.
- ParseResponse extracts useful content from JSON.
- Ensures clean, modular, and efficient communication with OpenAI's API in Unreal Engine.

Practical Procedures (15)

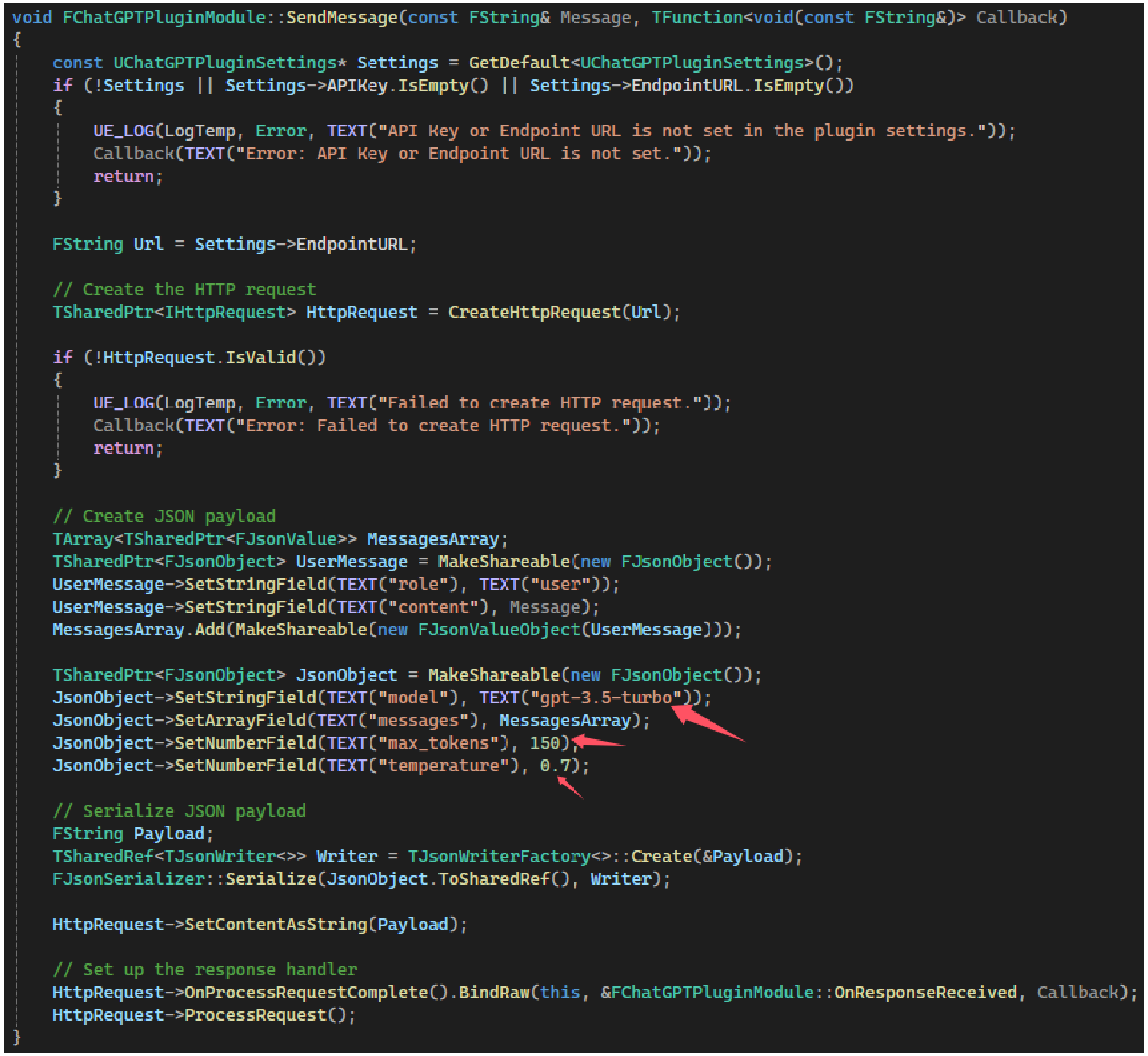
Implement core functionality in C++: Implementation file - ChatGPTPlugin.cpp (2)



```
#include "ChatGPTPlugin.h"
#include "ChatGPTPluginSettings.h"
#include "ISettingsModule.h"

#define LOCTEXT_NAMESPACE "FChatGPTPluginModule"

void FChatGPTPluginModule::StartupModule()
{
    // Register plugin settings
    if (ISettingsModule* SettingsModule = FModuleManager::GetModulePtr<ISettingsModule>("Settings"))
    {
        SettingsModule->RegisterSettings("Project", "Plugins", "ChatGPT Plugin",
            LOCTEXT("ChatGPTPluginSettingsName", "ChatGPT Plugin"),
            LOCTEXT("ChatGPTPluginSettingsDescription", "Configure the ChatGPT plugin."),
            GetMutableDefault<UChatGPTPluginSettings>());
    }

    UE_LOG(LogTemp, Log, TEXT("ChatGPT Plugin has started."));
}

void FChatGPTPluginModule::ShutdownModule()
{
    // Unregister plugin settings
    if (ISettingsModule* SettingsModule = FModuleManager::GetModulePtr<ISettingsModule>("Settings"))
    {
        SettingsModule->UnregisterSettings("Project", "Plugins", "ChatGPT Plugin");
    }

    UE_LOG(LogTemp, Log, TEXT("ChatGPT Plugin has shut down."));
}
```

Practical Procedures (16)

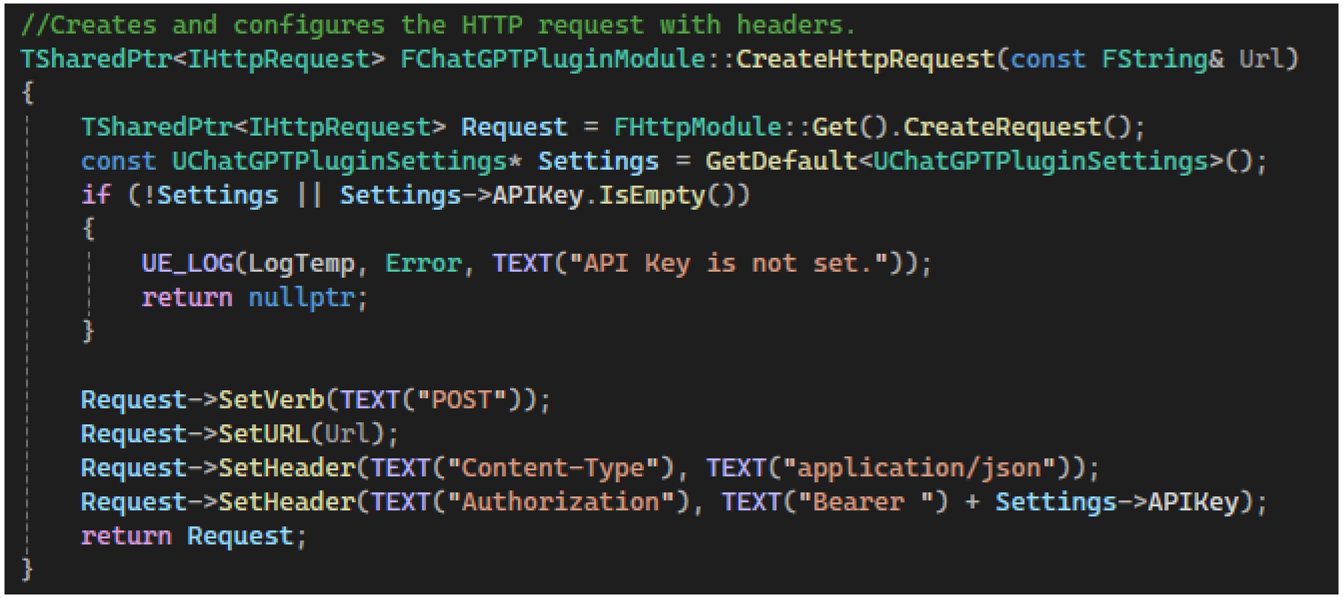
Implement core functionality in C++: Implementation file - ChatGPTPlugin.cpp (3)



```cpp
void FChatGPTPluginModule::SendMessage(const FString& Message, TFunction<void(const FString&)> Callback)
{
    const UChatGPTPluginSettings* Settings = GetDefault<UChatGPTPluginSettings>();
    if (!Settings || Settings->APIKey.IsEmpty() || Settings->EndpointURL.IsEmpty())
    {
        UE_LOG(LogTemp, Error, TEXT("API Key or Endpoint URL is not set in the plugin settings."));
        Callback(TEXT("Error: API Key or Endpoint URL is not set."));
        return;
    }

    FString Url = Settings->EndpointURL;

    // Create the HTTP request
    TSharedPtr<IHttpRequest> HttpRequest = CreateHttpRequest(Url);

    if (!HttpRequest.IsValid())
    {
        UE_LOG(LogTemp, Error, TEXT("Failed to create HTTP request."));
        Callback(TEXT("Error: Failed to create HTTP request."));
        return;
    }

    // Create JSON payload
    TArray<TSharedPtr<FJsonValue>> MessagesArray;
    TSharedPtr<FJsonObject> UserMessage = MakeShareable(new FJsonObject());
    UserMessage->SetStringField(TEXT("role"), TEXT("user"));
    UserMessage->SetStringField(TEXT("content"), Message);
    MessagesArray.Add(MakeShareable(new FJsonValueObject(UserMessage)));

    TSharedPtr<FJsonObject> JsonObject = MakeShareable(new FJsonObject());
    JsonObject->SetStringField(TEXT("model"), TEXT("gpt-3.5-turbo"));
    JsonObject->SetArrayField(TEXT("messages"), MessagesArray);
    JsonObject->SetNumberField(TEXT("max_tokens"), 150);
    JsonObject->SetNumberField(TEXT("temperature"), 0.7);

    // Serialize JSON payload
    FString Payload;
    TSharedRef<TJsonWriter<>> Writer = TJsonWriterFactory<>::Create(&Payload);
    FJsonSerializer::Serialize(JsonObject.ToSharedRef(), Writer);

    HttpRequest->SetContentAsString(Payload);

    // Set up the response handler
    HttpRequest->OnProcessRequestComplete().BindRaw(this, &FChatGPTPluginModule::OnResponseReceived, Callback);
    HttpRequest->ProcessRequest();
}
```

Practical Procedures (17)

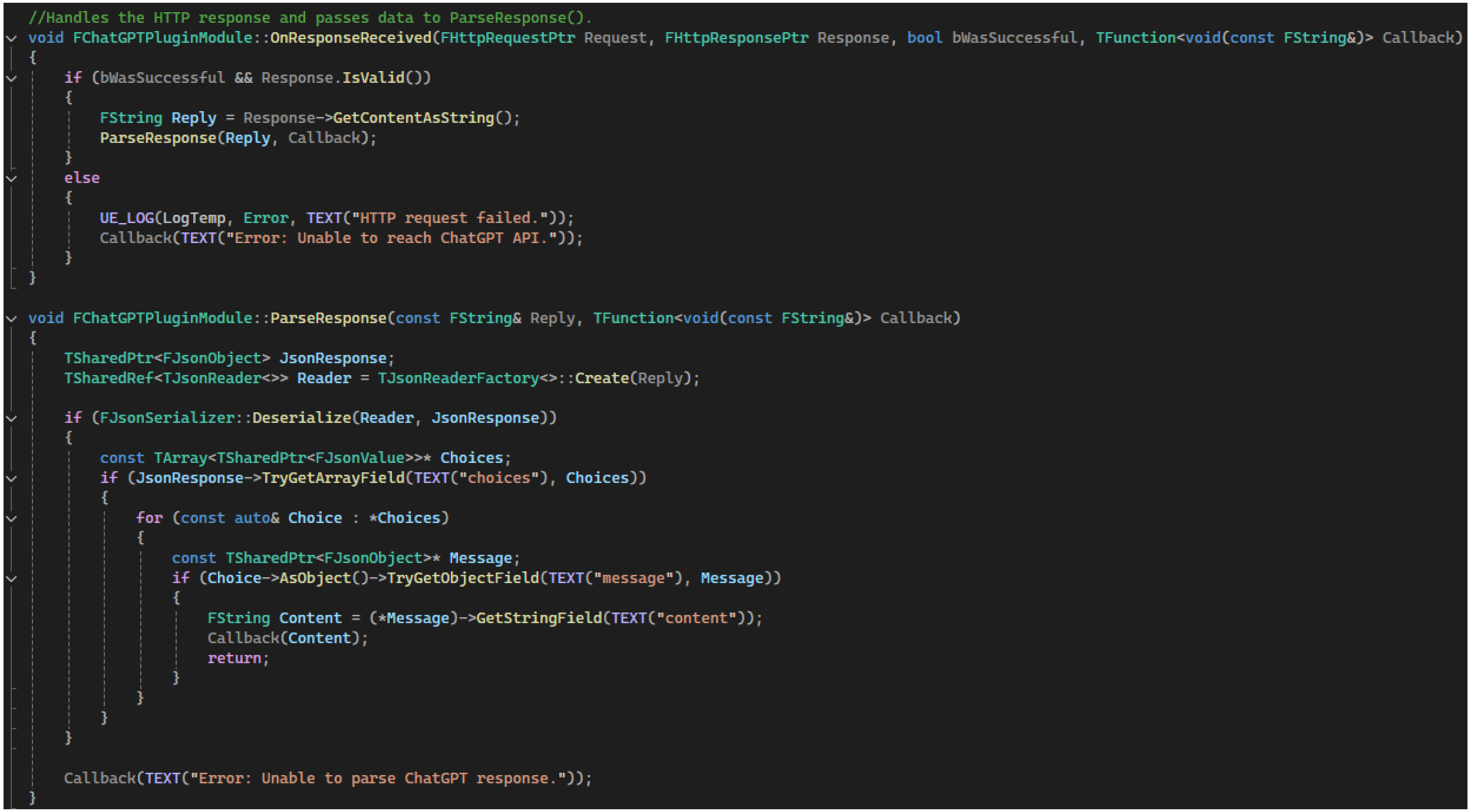
Implement core functionality in C++: Implementation file - ChatGPTPlugin.cpp (4)



```cpp
//Creates and configures the HTTP request with headers.
TSharedPtr<IHttpRequest> FChatGPTPluginModule::CreateHttpRequest(const FString& Url)
{
    TSharedPtr<IHttpRequest> Request = FHttpModule::Get().CreateRequest();
    const UChatGPTPluginSettings* Settings = GetDefault<UChatGPTPluginSettings>();
    if (!Settings || Settings->APIKey.IsEmpty())
    {
        UE_LOG(LogTemp, Error, TEXT("API Key is not set."));
        return nullptr;
    }

    Request->SetVerb(TEXT("POST"));
    Request->SetURL(Url);
    Request->SetHeader(TEXT("Content-Type"), TEXT("application/json"));
    Request->SetHeader(TEXT("Authorization"), TEXT("Bearer ") + Settings->APIKey);
    return Request;
}
```

Practical Procedures (18)

Implement core functionality in C++: Implementation file - ChatGPTPlugin.cpp (5)

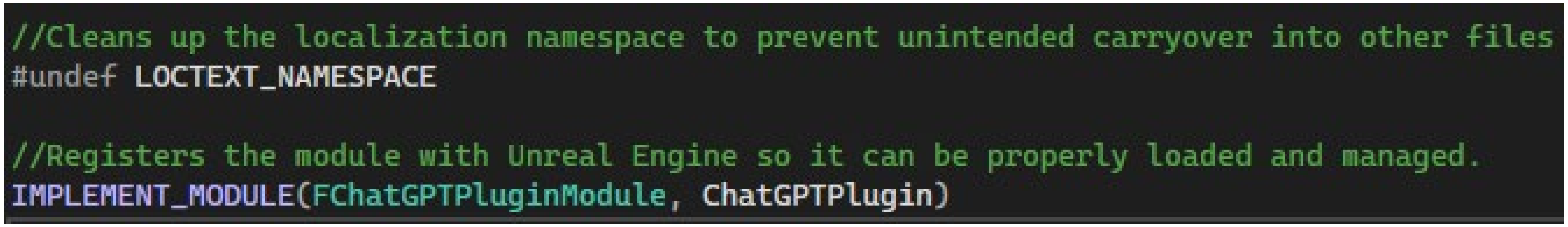


```cpp
//Handles the HTTP response and passes data to ParseResponse().
void FChatGPTPluginModule::OnResponseReceived(FHttpRequestPtr Request, FHttpResponsePtr Response, bool bWasSuccessful, TFunction<void(const FString&)> Callback)
{
    if (bWasSuccessful && Response.IsValid())
    {
        FString Reply = Response->GetContentAsString();
        ParseResponse(Reply, Callback);
    }
    else
    {
        UE_LOG(LogTemp, Error, TEXT("HTTP request failed."));
        Callback(TEXT("Error: Unable to reach ChatGPT API."));
    }
}

void FChatGPTPluginModule::ParseResponse(const FString& Reply, TFunction<void(const FString&)> Callback)
{
    TSharedPtr<FJsonObject> JsonResponse;
    TSharedRef<TJsonReader<>> Reader = TJsonReaderFactory<>::Create(Reply);

    if (FJsonSerializer::Deserialize(Reader, JsonResponse))
    {
        const TArray<TSharedPtr<FJsonValue>>* Choices;
        if (JsonResponse->TryGetArrayField(TEXT("choices"), Choices))
        {
            for (const auto& Choice : *Choices)
            {
                const TSharedPtr<FJsonObject>* Message;
                if (Choice->AsObject()->TryGetObjectField(TEXT("message"), Message))
                {
                    FString Content = (*Message)->GetStringField(TEXT("content"));
                    Callback(Content);
                    return;
                }
            }
        }
    }

    Callback(TEXT("Error: Unable to parse ChatGPT response."));
}
```

Practical Procedures (19)

Implement core functionality in C++: Implementation file - ChatGPTPlugin.cpp (6)

Don NOT forget



```
//Cleans up the localization namespace to prevent unintended carryover into other files
#undef LOCTEXT_NAMESPACE

//Registers the module with Unreal Engine so it can be properly loaded and managed.
IMPLEMENT_MODULE(FChatGPTPluginModule, ChatGPTPlugin)
```

If #undef LOCTEXT_NAMESPACE is missing and another file forgets to define a new LOCTEXT_NAMESPACE, any LOCTEXT calls in that file might incorrectly inherit the namespace from this file. This could result in:

- Wrong translations being applied.
- Debugging headaches when the wrong text gets localized.

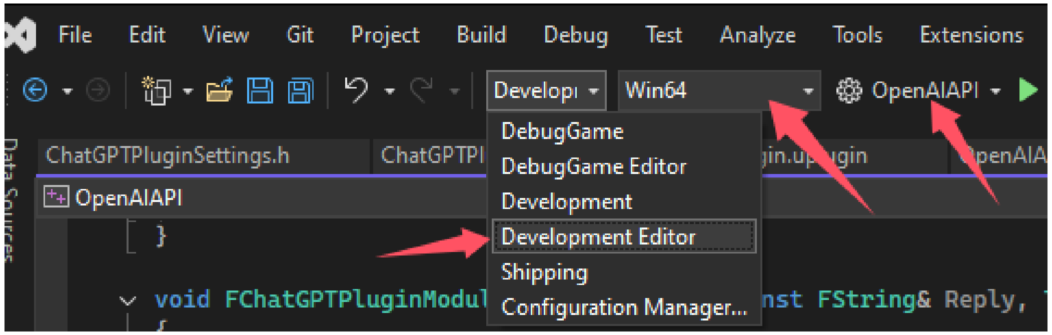
What Happens If You Forget IMPLEMENT_MODULE?

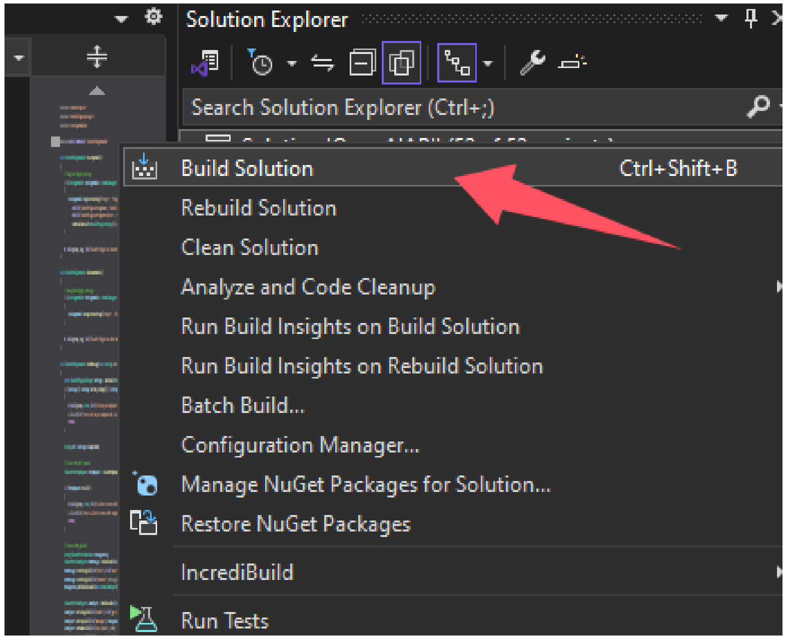
- Your plugin module will NOT be recognized or loaded by Unreal Engine!
  - Unreal won't call StartupModule(), meaning your plugin won't initialize.
  - Unreal won't call ShutdownModule(), meaning cleanup won't happen.

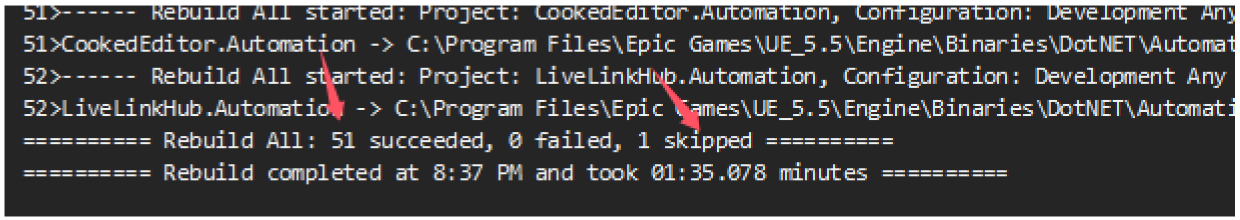
Always use IMPLEMENT_MODULE at the end of your module's .cpp file.

Practical Procedures (20)

Build and Get First ChatGPT Plugin: Configuration




Solution Explorer
Search Solution Explorer (Ctrl+;)
Build Solution Ctrl+Shift+B
Rebuild Solution
Clean Solution
Analyze and Code Cleanup
Run Build Insights on Build Solution
Run Build Insights on Rebuild Solution
Batch Build...
Configuration Manager...
Manage NuGet Packages for Solution...
Restore NuGet Packages
IncrediBuild
Run Tests



```
51>------ Rebuild All started: Project: CookedEditor.Automation, Configuration: Development Any
51>CookedEditor.Automation -> C:\Program Files\Epic Games\UE_5.5\Engine\Binaries\DotNET\Automat
52>------ Rebuild All started: Project: LiveLinkHub.Automation, Configuration: Development Any
52>LiveLinkHub.Automation -> C:\Program Files\Epic Games\UE_5.5\Engine\Binaries\DotNET\Automati
========== Rebuild All: 51 succeeded, 0 failed, 1 skipped ==========
========== Rebuild completed at 8:37 PM and took 01:35.078 minutes ==========
```

Practical Procedures (21)

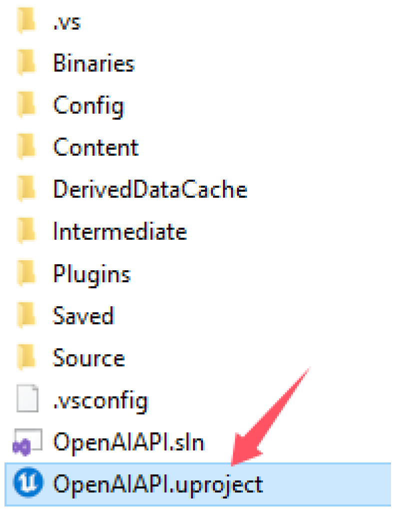
Build and Get First ChatGPT Plugin: Directory Structure after Building

```
ChatGPTPlugin/
├── Binaries/
├── Content/          <-- This folder stores plugin assets
├── Source/           <-- Source code files
├── Resources/        <-- Icon and other metadata
├── ChatGPTPlugin.uplugin
```

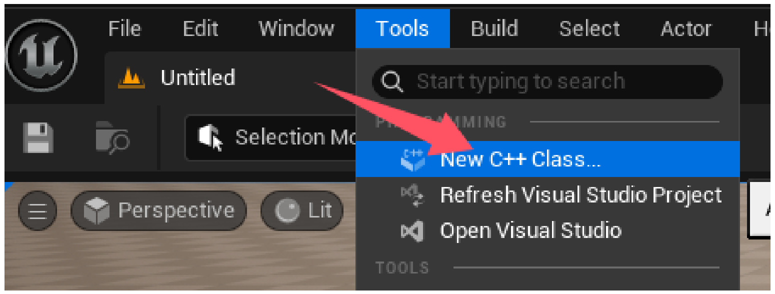
Practical Procedures (22)

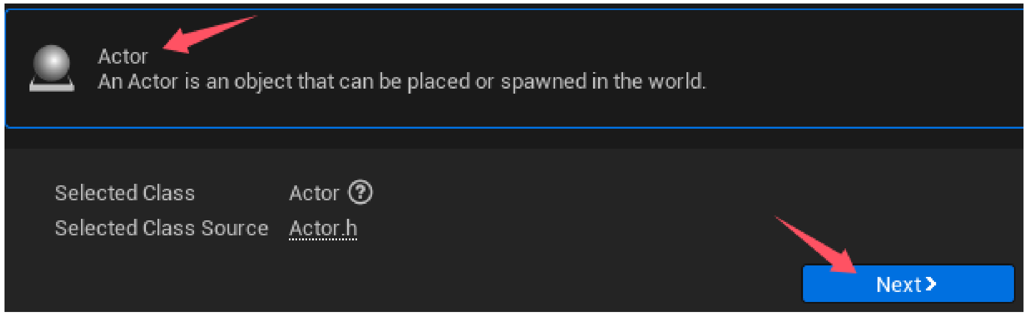
Test the Plugin: Create a New Actor

→ Open Unreal Editor (if promoted to



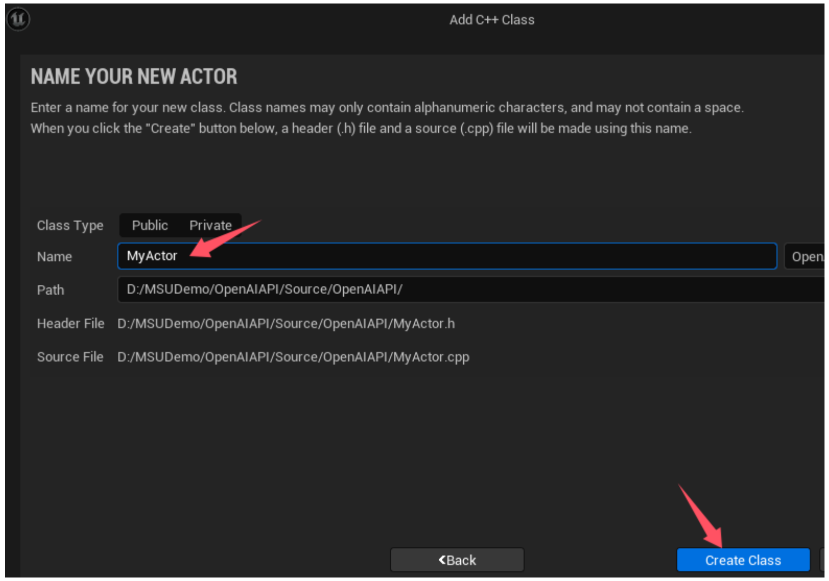
→ Create a test actor





An Actor is an object that can be placed or spawned in the world.

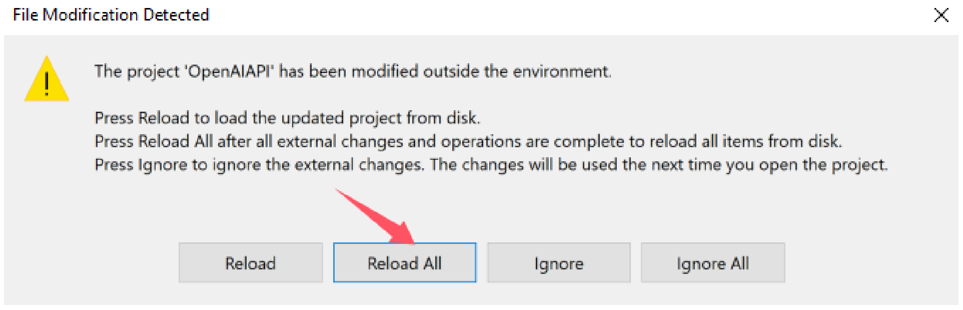
Selected Class Actor
Selected Class Source Actor.h



NAME YOUR NEW ACTOR

Enter a name for your new class. Class names may only contain alphanumeric characters, and may not contain a space.
When you click the "Create" button below, a header (.h) file and a source (.cpp) file will be made using this name.

Class Type	Public Private
Name	MyActor
Path	D:/MSUDemo/OpenAIAPI/Source/OpenAIAPI/
Header File	D:/MSUDemo/OpenAIAPI/Source/OpenAIAPI/MyActor.h
Source File	D:/MSUDemo/OpenAIAPI/Source/OpenAIAPI/MyActor.cpp



File Modification Detected

The project 'OpenAIAPI' has been modified outside the environment.

Press Reload to load the updated project from disk.
Press Reload All after all external changes and operations are complete to reload all items from disk.
Press Ignore to ignore the external changes. The changes will be used the next time you open the project.

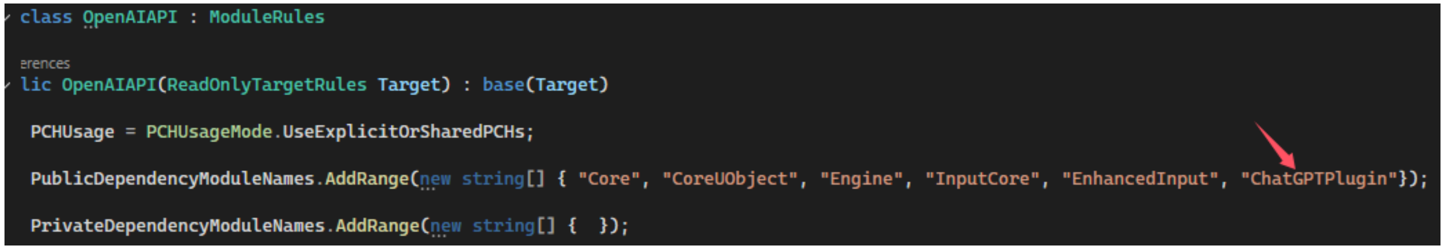
Practical Procedures (23)

Test the Plugin: Directory Structure with New Actor

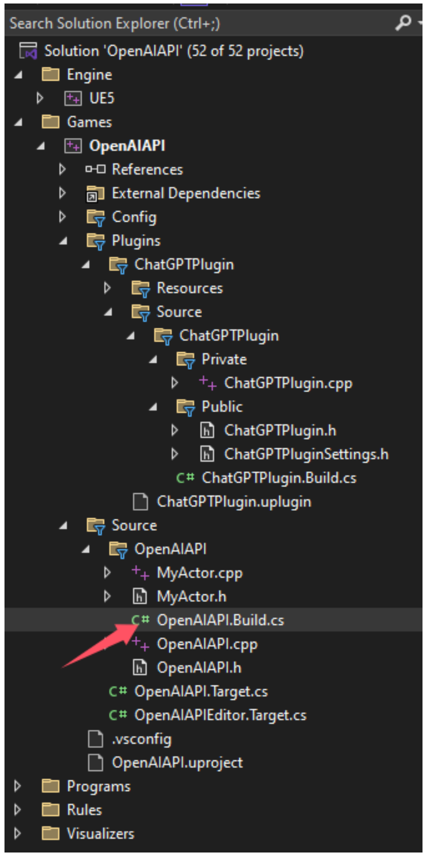
```
/OpenAIAPI/
│ ── Source/
│    ├── OpenAIAPI/              # Main Project Source Folder
│    │    ├── Public/            # Public headers
│    │    │    ├── MyActor.h
│    │    ├── Private/            # Private source files
│    │    │    ├── MyActor.cpp
│    │    ├── OpenAIAPI.Build.cs     # Build configuration for the main project
│
│ ── Plugins/
│    ├── ChatGPTPlugin/              # Plugin Folder
│    │    ├── Source/
│    │    │    ├── ChatGPTPlugin/      # Plugin Module
│    │    │    │    ├── Public/         # Public headers
│    │    │    │    │    ├── ChatGPTPlugin.h
│    │    │    │    │    ├── ChatGPTPluginSettings.h
│    │    │    │    ├── Private/         # Private implementation files
│    │    │    │    │    ├── ChatGPTPlugin.cpp
│    │    │    │    ├── ChatGPTPlugin.Build.cs  # Plugin build script
│    │    ├── ChatGPTPlugin.uplugin    # Plugin descriptor file
│
│ ── OpenAIAPI.uproject            # Unreal Engine project file
```

Practical Procedures (24)

Test the Plugin: Add new plugin dependency in OpenAIAPI.Build.cs



```
class OpenAIAPI : ModuleRules

erences
lic OpenAIAPI(ReadOnlyTargetRules Target) : base(Target)

 PCHUsage = PCHUsageMode.UseExplicitOrSharedPCHs;

 PublicDependencyModuleNames.AddRange(new string[] { "Core", "CoreUObject", "Engine", "InputCore", "EnhancedInput", "ChatGPTPlugin"});

 PrivateDependencyModuleNames.AddRange(new string[] {  });
```



```
Search Solution Explorer (Ctrl+;)
Solution 'OpenAIAPI' (52 of 52 projects)
  Engine
    UE5
  Games
    OpenAIAPI
      References
      External Dependencies
      Config
      Plugins
        ChatGPTPlugin
          Resources
          Source
            ChatGPTPlugin
              Private
                ChatGPTPlugin.cpp
              Public
                ChatGPTPlugin.h
                ChatGPTPluginSettings.h
              ChatGPTPlugin.Build.cs
          ChatGPTPlugin.uplugin
      Source
        OpenAIAPI
          MyActor.cpp
          MyActor.h
          OpenAIAPI.Build.cs
          OpenAIAPI.cpp
          OpenAIAPI.h
        OpenAIAPI.Target.cs
        OpenAIAPIEditor.Target.cs
      .vsconfig
      OpenAIAPI.uproject
  Programs
  Rules
  Visualizers
```

Practical Procedures (25)

Test the Plugin: MyActor.h (1)

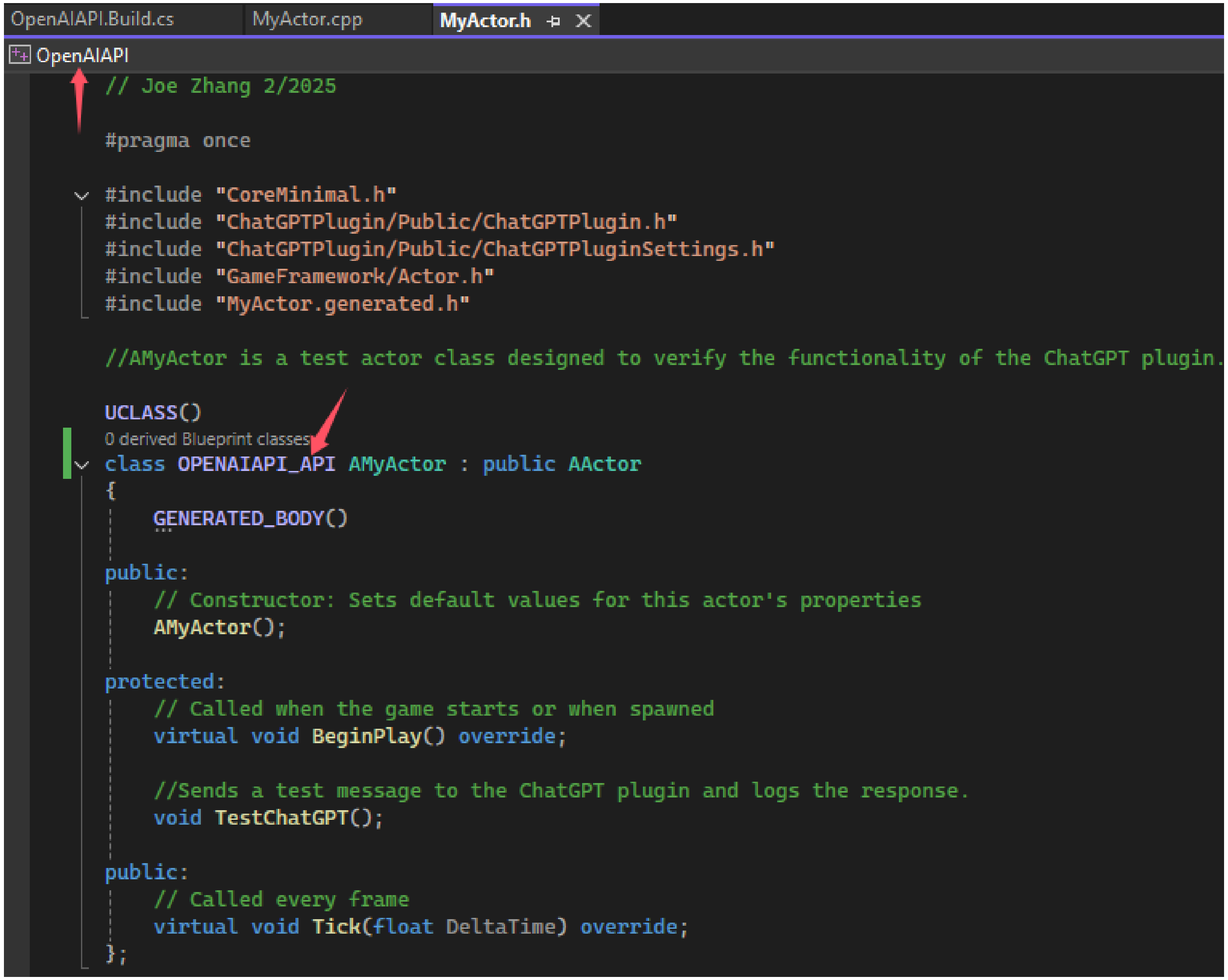
- AMyActor is an actor class that tests the functionality of ChatGPTPlugin in Unreal Engine.
- It sends a request to the ChatGPT API when the game starts and logs the response.

Function	Purpose
AMyActor()	Constructor – Initializes default values.
BeginPlay()	Called when the game starts, triggers TestChatGPT().
TestChatGPT()	Sends a test message to ChatGPT and logs the response.
Tick(float)	Called every frame (not currently modified).

Included Headers	Purpose
"CoreMinimal.h"	Essential Unreal Engine types.
"ChatGPTPlugin/Public/ChatGPTPlugin.h"	Access to FChatGPTPluginModule.
"ChatGPTPlugin/Public/ChatGPTPluginSettings.h"	Access to API settings.
"GameFramework/Actor.h"	Base class for actors in Unreal Engine.

Practical Procedures (26)

Test the Plugin: MyActor.h (2)



```
// Joe Zhang 2/2025

#pragma once

#include "CoreMinimal.h"
#include "ChatGPTPlugin/Public/ChatGPTPlugin.h"
#include "ChatGPTPlugin/Public/ChatGPTPluginSettings.h"
#include "GameFramework/Actor.h"
#include "MyActor.generated.h"

//AMyActor is a test actor class designed to verify the functionality of the ChatGPT plugin.

UCLASS()
class OPENAIAPI_API AMyActor : public AActor
{
    GENERATED_BODY()

public:
    // Constructor: Sets default values for this actor's properties
    AMyActor();

protected:
    // Called when the game starts or when spawned
    virtual void BeginPlay() override;

    //Sends a test message to the ChatGPT plugin and logs the response.
    void TestChatGPT();

public:
    // Called every frame
    virtual void Tick(float DeltaTime) override;
};
```

Practical Procedures (27)

Test the Plugin: MyActor.cpp (1)

Implements the logic for AMyActor, as a test actor to verify the functionality of plugin

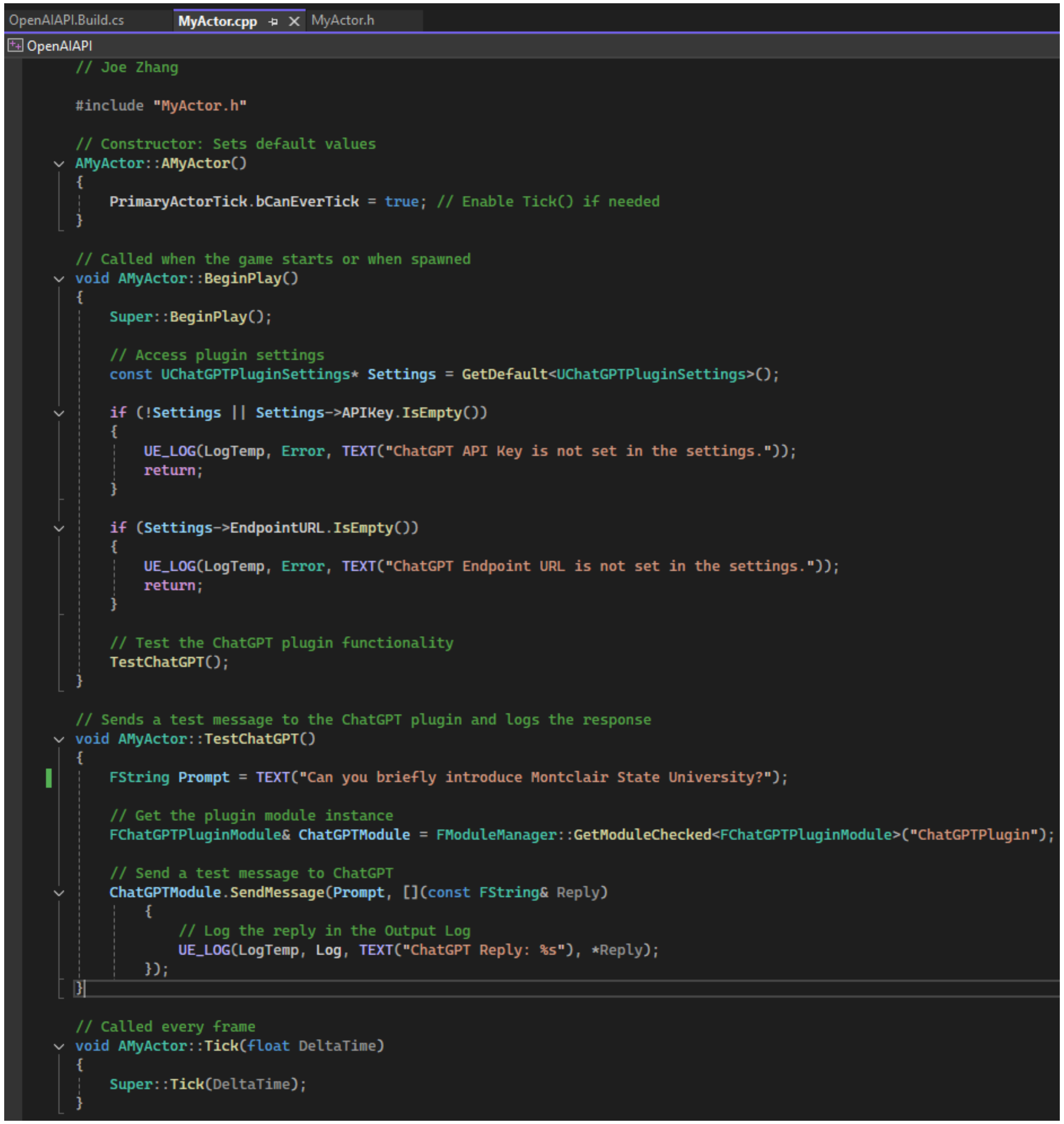
- AMyActor initializes at game start, checks the ChatGPT API key and endpoint, and sends a test request to OpenAI's API.
- The response is logged in Unreal Engine's output console.

Function	Purpose
AMyActor()	Constructor – Initializes default values and enables Tick().
BeginPlay()	Called when the game starts, retrieves API settings, and calls TestChatGPT().
TestChatGPT()	Sends a predefined prompt to ChatGPT and logs the response.
Tick(float)	Currently does nothing, but can be used for updates.

Included Headers	Purpose
"MyActor.h"	Includes the class definition.
"ChatGPTPlugin/Public/ChatGPTPlugin.h"	Allows access to FChatGPTPluginModule for sending messages.
"ChatGPTPlugin/Public/ChatGPTPluginSettings.h"	Retrieves API key and endpoint settings.

Practical Procedures (28)

Test the Plugin: MyActor.cpp (2)



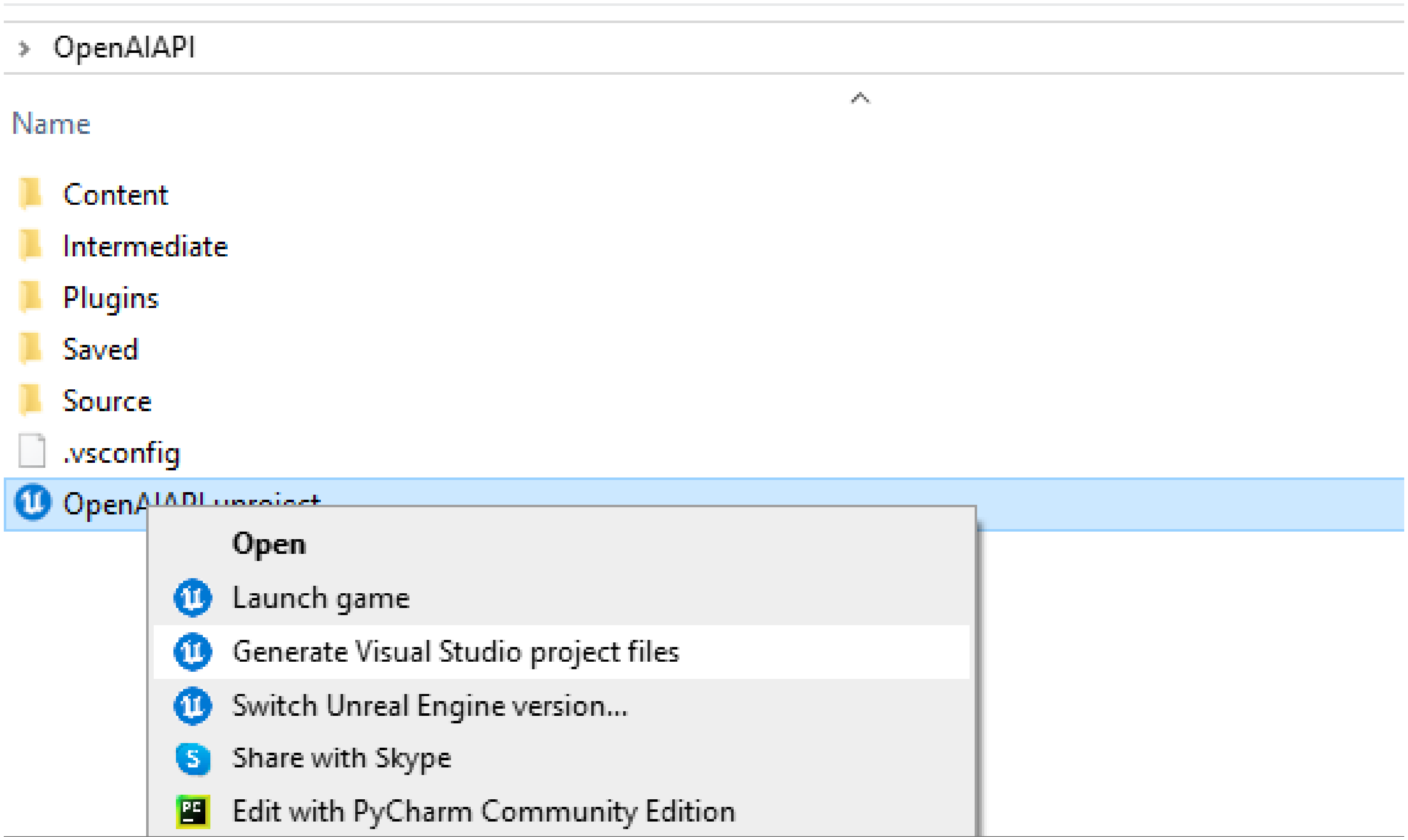
OpenAIAPI.Build.cs | MyActor.cpp | MyActor.h

OpenAIAPI

```cpp
// Joe Zhang

#include "MyActor.h"

// Constructor: Sets default values
AMyActor::AMyActor()
{
    PrimaryActorTick.bCanEverTick = true; // Enable Tick() if needed
}

// Called when the game starts or when spawned
void AMyActor::BeginPlay()
{
    Super::BeginPlay();

    // Access plugin settings
    const UChatGPTPluginSettings* Settings = GetDefault<UChatGPTPluginSettings>();

    if (!Settings || Settings->APIKey.IsEmpty())
    {
        UE_LOG(LogTemp, Error, TEXT("ChatGPT API Key is not set in the settings."));
        return;
    }

    if (Settings->EndpointURL.IsEmpty())
    {
        UE_LOG(LogTemp, Error, TEXT("ChatGPT Endpoint URL is not set in the settings."));
        return;
    }

    // Test the ChatGPT plugin functionality
    TestChatGPT();
}

// Sends a test message to the ChatGPT plugin and logs the response
void AMyActor::TestChatGPT()
{
    FString Prompt = TEXT("Can you briefly introduce Montclair State University?");

    // Get the plugin module instance
    FChatGPTPluginModule& ChatGPTModule = FModuleManager::GetModuleChecked<FChatGPTPluginModule>("ChatGPTPlugin");

    // Send a test message to ChatGPT
    ChatGPTModule.SendMessage(Prompt, [](const FString& Reply)
        {
            // Log the reply in the Output Log
            UE_LOG(LogTemp, Log, TEXT("ChatGPT Reply: %s"), *Reply);
        });
}

// Called every frame
void AMyActor::Tick(float DeltaTime)
{
    Super::Tick(DeltaTime);
}
```

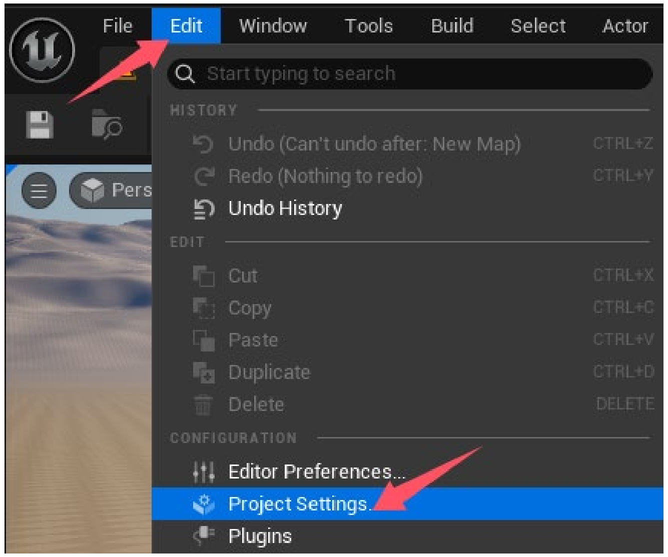
Practical Procedures (29)

Test the Plugin: Try to Generate Visual Studio project files if build failed

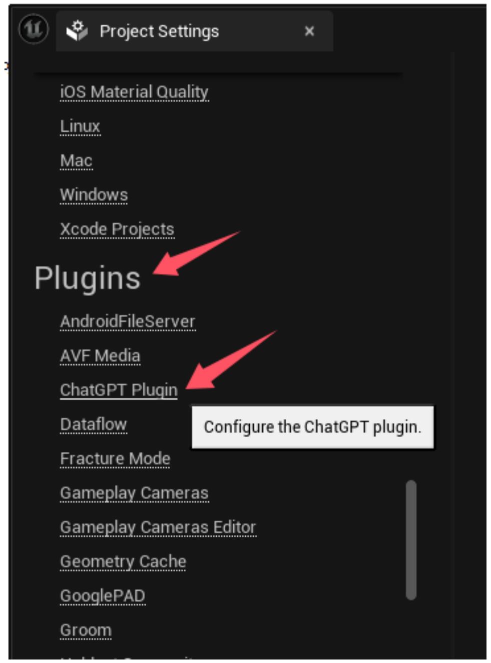


Practical Procedures (30)

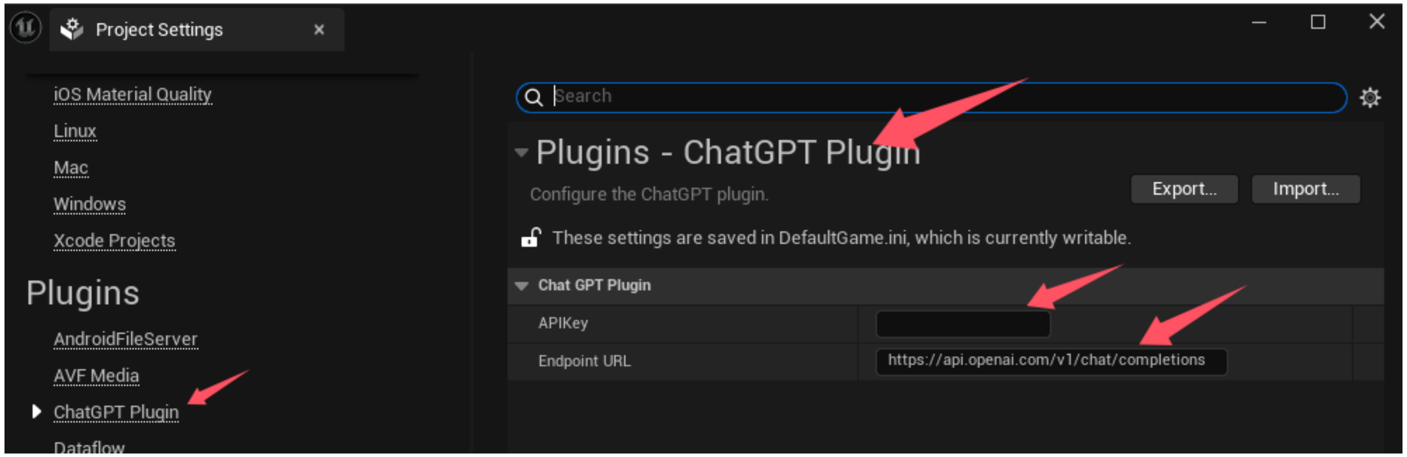
Run Unreal Editor Again
→Edit → Project Settings



→ Plugins → ChatGPTPlugin



→ Input API key and URL



Plugins - ChatGPT Plugin

Configure the ChatGPT plugin.

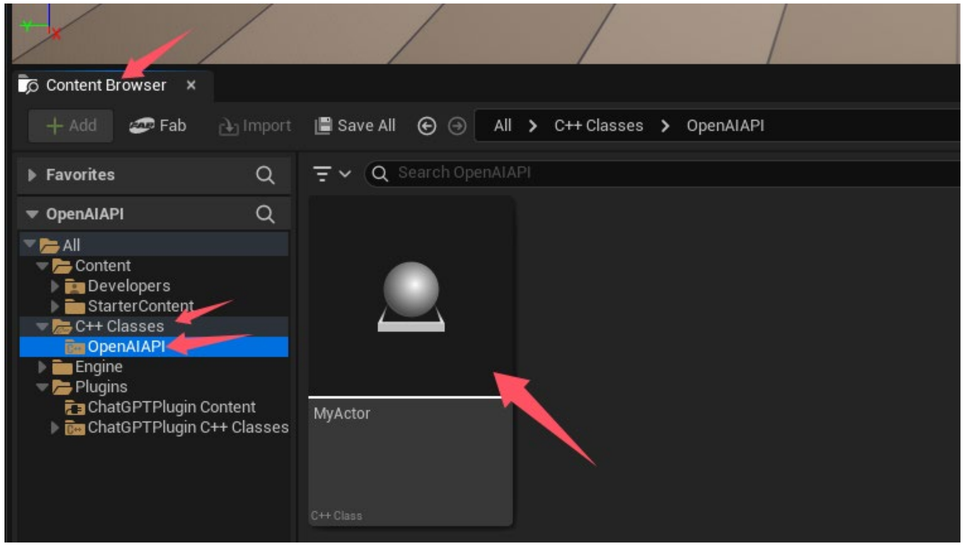
These settings are saved in DefaultGame.ini, which is currently writable.

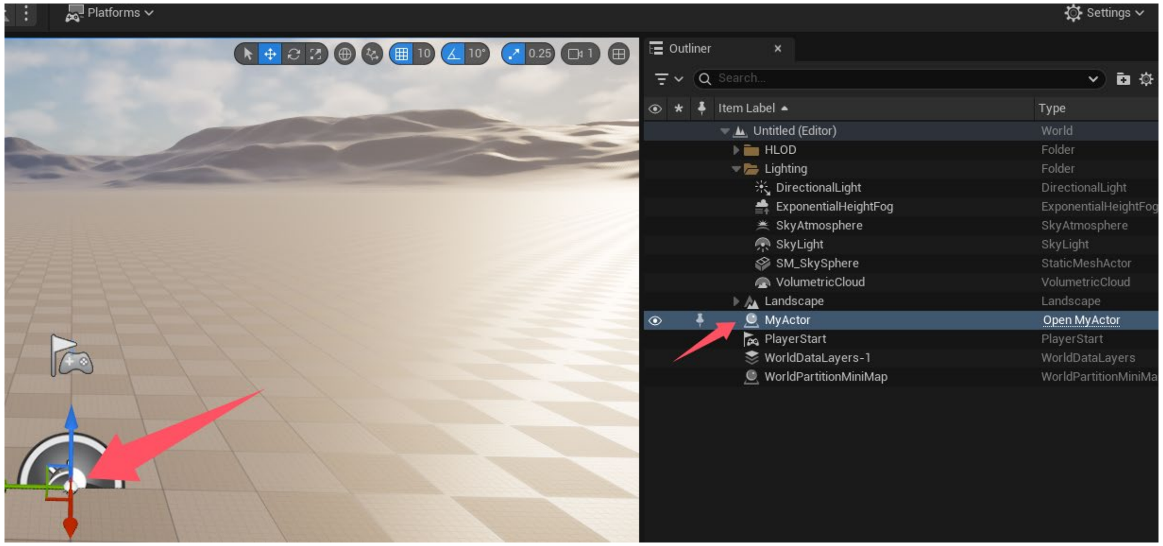
Chat GPT Plugin

APIKey	
Endpoint URL	https://api.openai.com/v1/chat/completions

Practical Procedures (31)

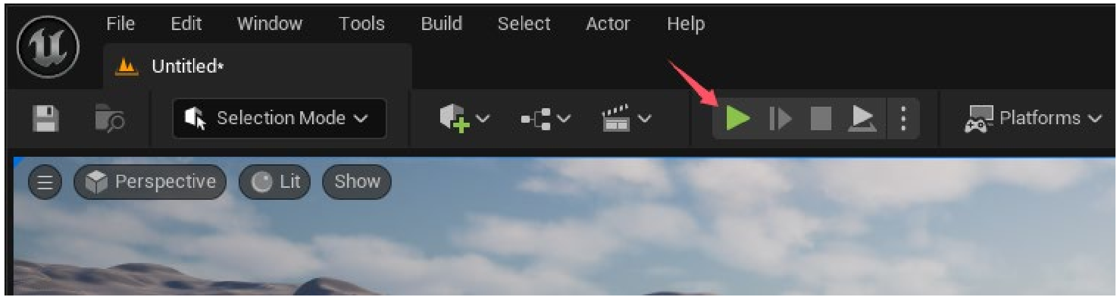
Place MyActor in the Level:
→ Content Browser→ C++ Classes → OpenAIAPI → MyActor → Drag MyActor to the Level

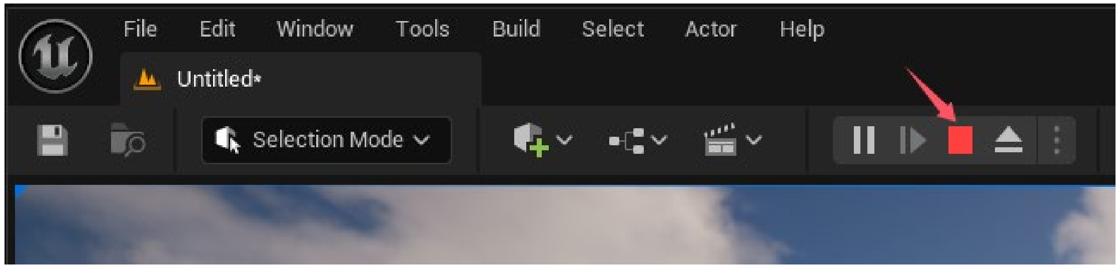


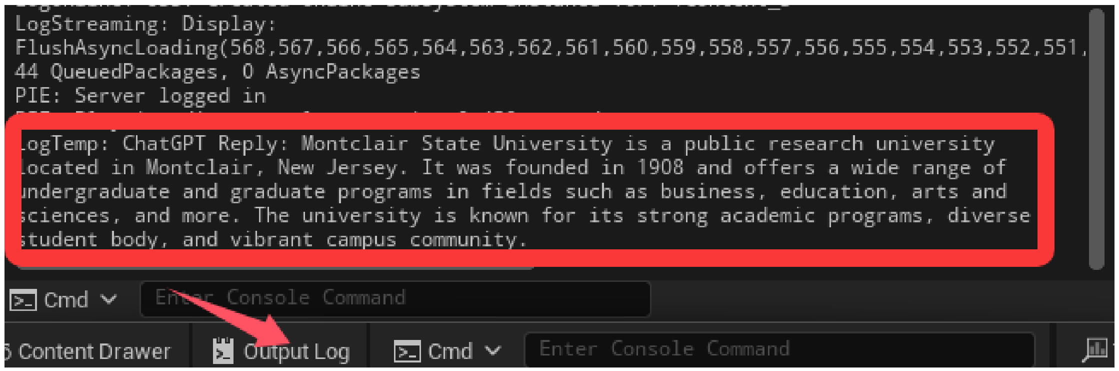

Platforms
Settings
Outliner
Search...
Item Label
Type
Untitled (Editor) World
HLOD Folder
Lighting Folder
DirectionalLight DirectionalLight
ExponentialHeightFog ExponentialHeightFog
SkyAtmosphere SkyAtmosphere
SkyLight SkyLight
SM_SkySphere StaticMeshActor
VolumetricCloud VolumetricCloud
Landscape Landscape
MyActor Open MyActor
PlayerStart PlayerStart
WorldDataLayers-1 WorldDataLayers
WorldPartitionMiniMap WorldPartitionMiniMa

Practical Procedures (32)

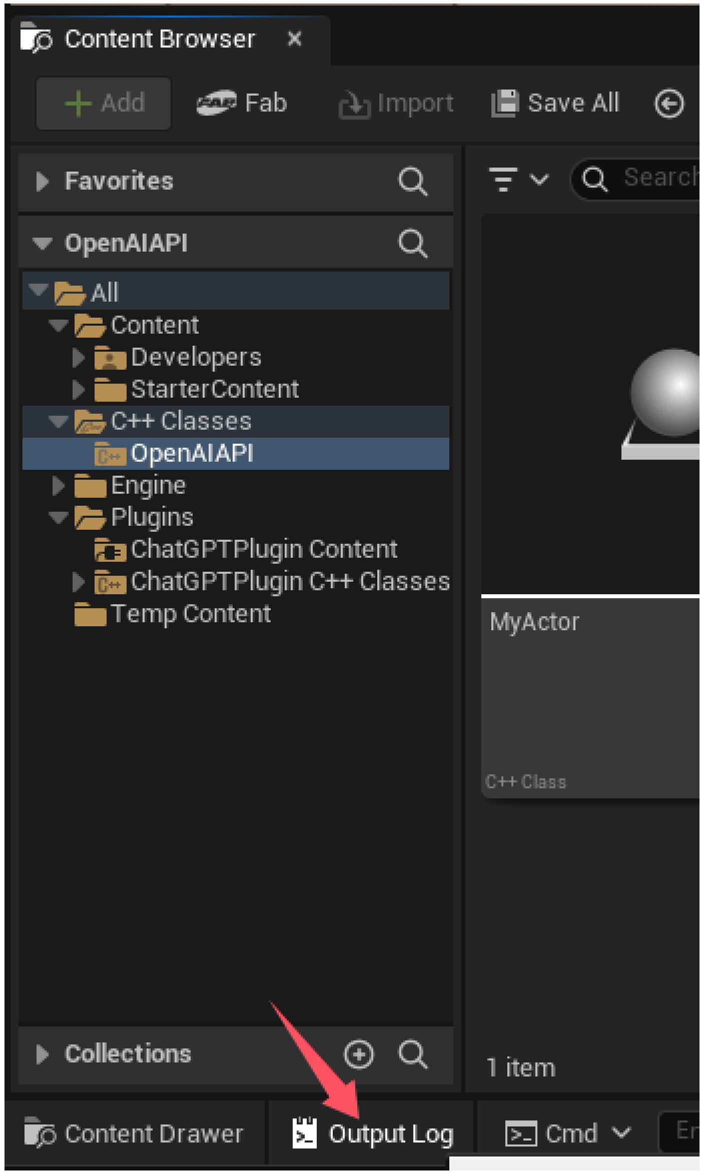
Play this Level and Check Output Log:







```
LogStreaming: Display:
FlushAsyncLoading(568,567,566,565,564,563,562,561,560,559,558,557,556,555,554,553,552,551,
44 QueuedPackages, 0 AsyncPackages
PIE: Server logged in
LogTemp: ChatGPT Reply: Montclair State University is a public research university
located in Montclair, New Jersey. It was founded in 1908 and offers a wide range of
undergraduate and graduate programs in fields such as business, education, arts and
sciences, and more. The university is known for its strong academic programs, diverse
student body, and vibrant campus community.
```



Recap

- ❑ Plugin creation in Unreal Engine
- ❑ OpenAI API integration steps
- ❑ Potential AI-powered game applications