

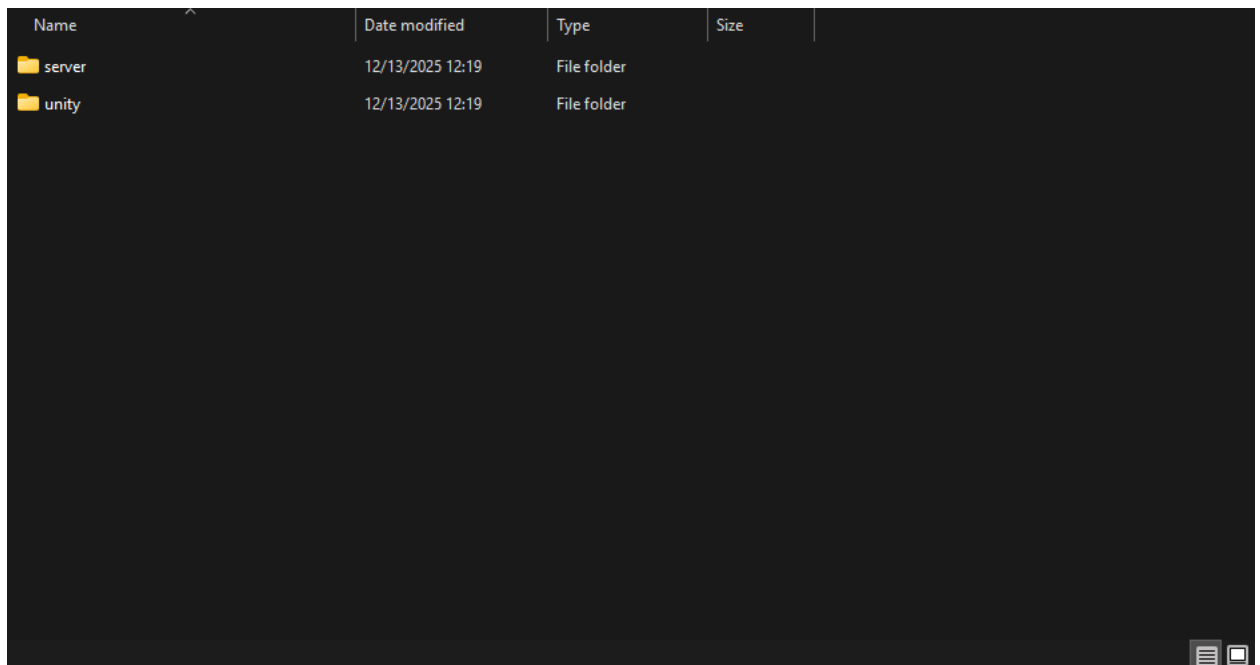
Installation Guide

Prerequisites

- **Unity Hub & Unity Editor** version **6000.0.62f1**
- **Python** version **3.13**
- **Google Gemini API Key**
- **Google Cloud Translation API Key File**

Instructions

1. First, users should download and decompress the source code. The directory should look like the screenshot below:



2. Next, using the command line, users should navigate to the server folder and download / install all required dependencies using the command “`pip install -r requirements.txt`”. It is important to note that this relies on pip being on the path. If it is not, the full command must refer to the python interpreter: “`(python path) -m pip install -r requirements.txt`”. After installation, a message resembling the following

should be displayed, indicating all dependencies have been installed:

```
Successfully installed annotated-doc-0.0.4 annotated-types-0.7.0 anyio-4.12.0 cachetools-6.2.3 certifi-2025.11.12 charset-normalizer-3.4.4 click-8.3.1 colorama-0.4.6 fastapi-0.121.3 google-api-core-2.28.1 google-api-python-client-2.187.0 google-auth-2.43.0 google-auth-http-2.0.2 googleapis-common-protos-1.72.0 h11-0.16.0 httplib2-0.31.0 idna-3.11 proto-plus-1.26.1 protobuf-5.29.5 pyasn1-0.6.1 pyasn1-modules-0.4.2 pydantic-2.12.5 pydantic-core-2.41.5 pyparsing-3.2.5 requests-2.32.5 rsa-4.9.1 starlette-0.50.0 typing-extensions-4.15.0 typing-inspection-0.4.2 uritemplate-4.2.0 urllib3-2.6.2 uvicorn-0.38.0
```

- Next, the computer side application must be configured for use with the appropriate API keys. Users should open the “translationsAndImages.py” in a text editor and change the “`environ["GOOGLE_API_KEY"] = "" # gemini api key here`” to where their Gemini API key is stored within the quotes. Additionally, the line “`environ["GOOGLE_APPLICATION_CREDENTIALS"] = "mimetic-sunset-479220-f9-868f285745aa.json"`” should have the Json file name updated to reflect the user’s key file.

```
9 # --- Configure Gemini ---
10 environ["GOOGLE_API_KEY"] = "" # gemini api key here
11 genai.configure(api_key=getenv("GOOGLE_API_KEY"))
12
13 # Custom search engine ID
14 CX_SEARCH_ENGINE = "8773d847969914505"
15
16 app = FastAPI(title="Bilingual Translation & Food Image API", version="1.0")
17
153 if __name__ == "__main__":
154     environ["GOOGLE_APPLICATION_CREDENTIALS"] = "mimetic-sunset-479220-f9-868f285745aa.json"
155     import uvicorn
156     uvicorn.run(app, host="0.0.0.0", port=8000)
157
```

- After both fields have been configured, the computer side server can now be run. Though this command varies by PATH configuration, the basic syntax is the path to the python interpreter followed by the application. If prompted to configure firewall click “Allow.” If the command ran successfully, the following should be displayed on the console, indicating Fast API is working:

```
INFO: Started server process [18688]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
```

- The user can now test the API by visiting the link or directly going to <http://127.0.0.1:8000/docs> on their web browser. This site shows all the endpoints

that are open for communication.

Bilingual Translation & Food Image API 1.0 OAS 3.1

/openapi.json

default

POST /general-translation General Translation

POST /translate-with-image Translate With Image

GET / Root

Schemas

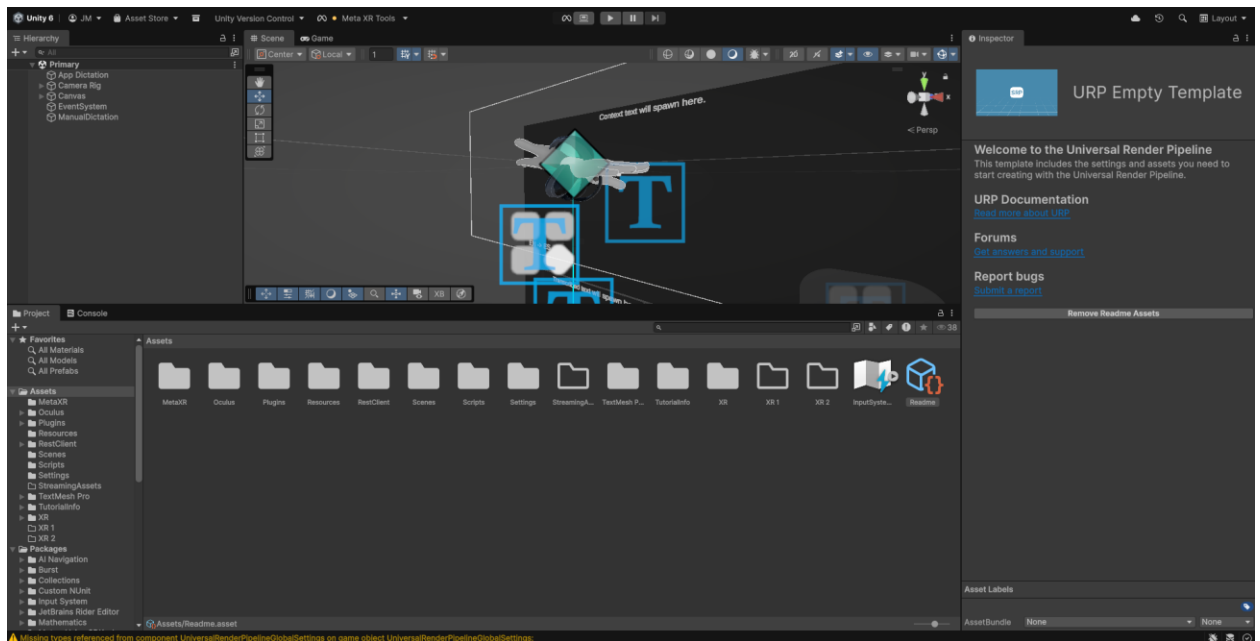
GeneralTranslationRequest > Expand all object

HTTPValidationError > Expand all object

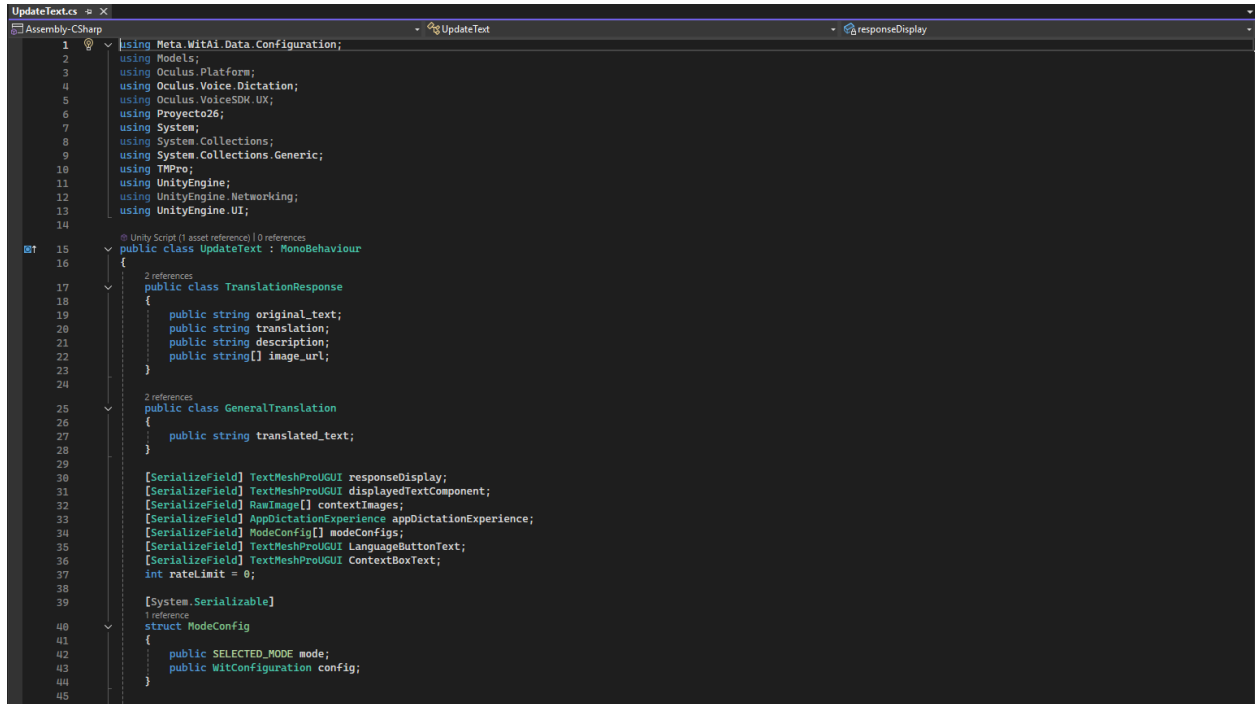
TranslationRequest > Expand all object

ValidationError > Expand all object

- Now the Unity side must be configured. Using the unity editor, the folder “unity” should be opened as a project. The current display may look different than the screenshot below, but the folder structure will be the same.



7. Navigate to scripts and open the script titled “UpdateText.cs”. The file could be opened in any text editor.



```
1  using Meta.WitAI.Data.Configuration;
2  using Models;
3  using Oculus.Platform;
4  using Oculus.Voice.Dictation;
5  using Oculus.VoiceSDK.UX;
6  using Proyecto26;
7  using System;
8  using System.Collections;
9  using System.Collections.Generic;
10 using TMPro;
11 using UnityEngine;
12 using UnityEngine.Networking;
13 using UnityEngine.UI;
14
15 public class UpdateText : MonoBehaviour
16 {
17     2 references
18     public class TranslationResponse
19     {
20         public string original_text;
21         public string translation;
22         public string description;
23         public string[] image_url;
24     }
25
26     3 references
27     public class GeneralTranslation
28     {
29         public string translated_text;
30     }
31
32     [SerializeField] TextMeshProUGUI responseDisplay;
33     [SerializeField] TextMeshProUGUI displayedTextComponent;
34     [SerializeField] RawImage[] contextImages;
35     [SerializeField] AppDictationExperience appDictationExperience;
36     [SerializeField] ModeConfig[] modeConfigs;
37     [SerializeField] TextMeshProUGUI languageButtonText;
38     [SerializeField] TextMeshProUGUI contextBoxText;
39     int rateLimit = 0;
40
41     [System.Serializable]
42     struct ModeConfig
43     {
44         public SELECTED_MODE mode;
45         public WitConfiguration config;
46     }
47 }
```

8. The next step varies by operating system. On windows, in a terminal, run “ipconfig” to retrieve the local IP address of the device running the server.



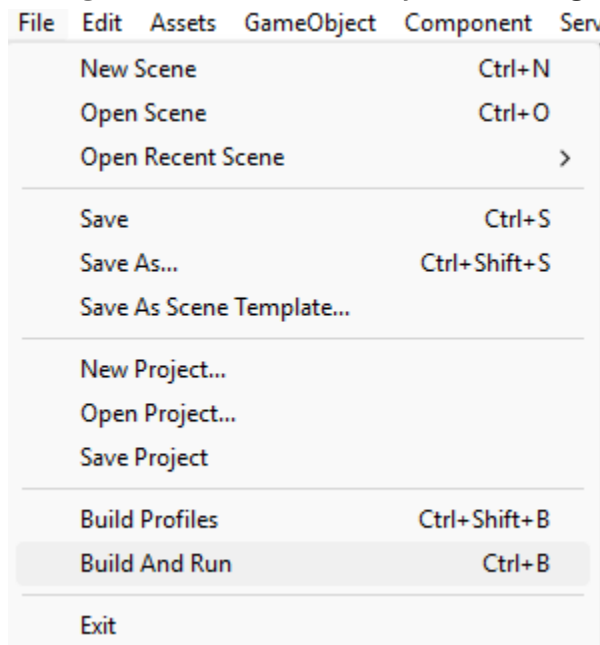
```
IPv4 Address. . . . . : 192.168.1.229
Subnet Mask . . . . . : 255.255.255.224
Default Gateway . . . . . : 192.168.1.225
```

9. The IP retrieved from the previous step should be substituted twice inside of the UpdateText.cs script on lines 60 and 79.



```
75     if (rateLimit == 1)
76     {
77         RequestHelper data = new RequestHelper
78         {
79             Uri = "http://192.168.1.229:8080/translate-with-image",
80             Headers = new Dictionary<string, string> {
81                 { "Content-Type", "application/json" }
82             },
83             BodyString = $"{{"text": \"{text}\", \"target_language\": \"{((currentMode == SELECTED_MODE.ESTOENUS) ? \"english\" : \"spanish\")}\"}}",
84             EnableDebug = true
85         };
86
87         RequestHelper otherData = new RequestHelper
88         {
89             Uri = "http://192.168.1.229:8080/general-translation",
90             Headers = new Dictionary<string, string> {
91                 { "Content-Type", "application/json" }
92             },
93             BodyString = (currentMode == SELECTED_MODE.ENUSTOES) ? $"{{"text": \"{text}\", \"source_language_code\": \"en-US\", \"target_language_code\": \"es\"}}"} : $"{{"text": \"{t
```

10. At this point, the application can be compiled for the headset. Save and close the UpdateText.cs script. Then, with the **Oculus Quest 2 headset inserted and in debug mode**, within the Unity Editor navigate to File > Build and Run.



11. The application should now compile and be loaded on to the headset, of which it can then be used and relaunched from the headset itself.