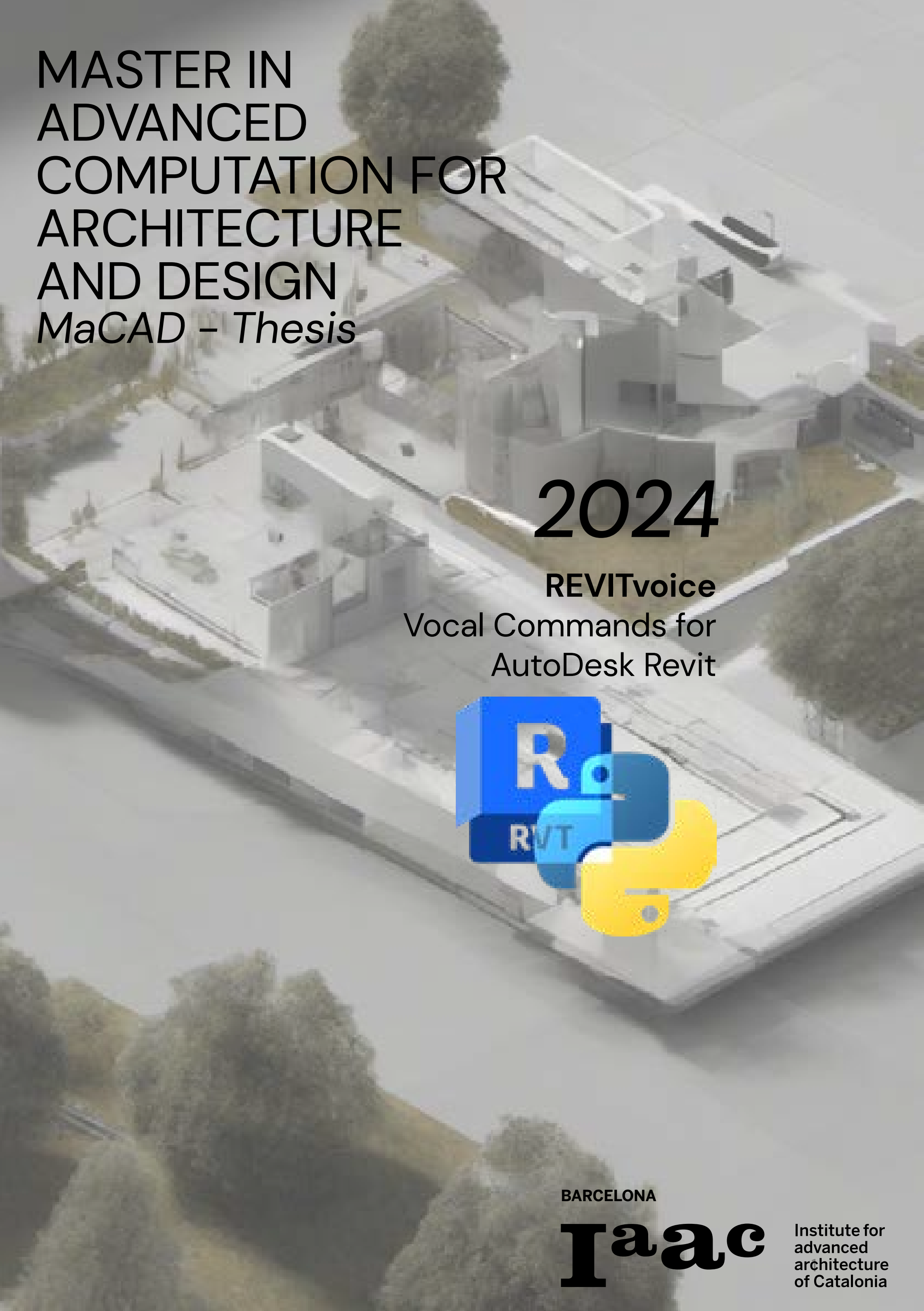
MASTER IN ADVANCED COMPUTATION FOR ARCHITECTURE AND DESIGN

MaCAD - Thesis



2024

REVITvoice

Vocal Commands for AutoDesk Revit

BARCELONA

Iaac Institute for advanced architecture of Catalonia

MASTER IN ADVANCED COMPUTATION
FOR ARCHITECTURE AND DESIGN

Project Title: REVIT Automated

MaCAD 2024 – Thesis

Faculty: David Andres Leon

Faculty Assistant: Gabriella Rossi

Dominic Large

James Foo

INDEX

Abstract

Thesis

In the past several years, Building Information Modeling (BIM) has continued to expand and transform the architectural and structural design worlds. This has been achieved through the use of software like AutoDesk Revit, ArchiCAD, and Vectorworks, however, the introduction of technology like machine learning (ML) and artificial intelligence (AI) have begun to make these programs out-dated in terms of efficiency and productivity. One major development in this regard is the advancing proficiency of speech recognition technology through the use of coding languages such as Python, C#, JavaScript and HTML. These advancements offer the potential to eliminate manual tasks, optimize workflows, and make BIM processes more widely accessible and user friendly.

This research project aims to understand the potential benefits a coding language like Python, in combination with both speech recognition technology and existing AutoDesk Revit software, might offer the architectural and design community. By using speech recognition to automate manual computer aided design (CAD), designers can accelerate their efficiency and production of future projects.

Research Question & Objective

The main question driving this research project is: *Can machine learning through the use of Python be used to create a trained model that successfully and quickly recognizes and implements commands within Revit Family parameters using only the Human Voice and a prompt?*

In order to answer this proposed research question, the main objective will be to explore the most proficient and accessible ways to implement speech recognition technology into a Revit friendly plug-in. This plug-in will be designed to automate specific tasks; Specifically, creating and / or modifying Revit family elements.

Theis Milestones

In order to realize this project, three major milestones must be met; The first being **Voice-to-Text Conversion.** This milestone will require training a machine learning model to accurately transcribe spoken prompts and use grammatical practices to identify both specific words that refer to Revit family parameters and their corresponding values. A quick path to achieving this will likely be to use a natural language processing tool (NLP) that can be run within Python on virtually any machine with a microphone.

The second major milestone will be a **Text-to-Revit Command** process. Once prompts are recorded, they must be broken down and translated into a form that Revit will be able to understand and use as commands. The simplest route to achieve this will most likely be converting identified parameters and their values into a JSON file that can then be read by Revit usit PyRevit.

Finally, successfully deploying a plug-in with a **Human-Centered User Interface** (UI). Ensuring that this tool is easy to use and can run efficiently for other designers will complete the scope of this project and prove the reason to use the tool and continue its expansion.

Future Opportunity

Due to the minimal time frame of the project, the initial focus will remain on successfully translating spoken words into viable commands within Revit, there will continue to be opportunity to expand the thesis.

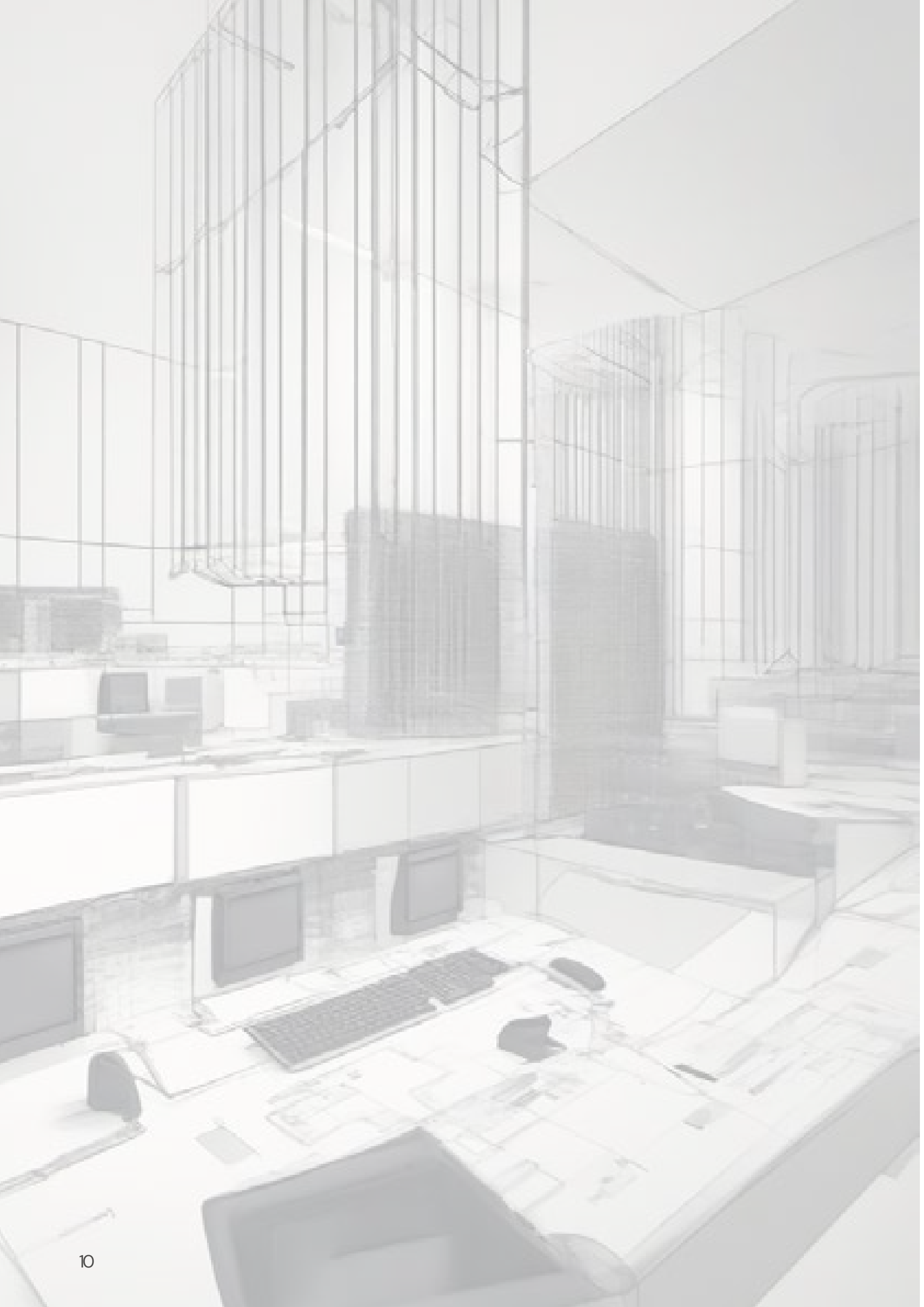
In addition to modifying single types of elements within a family and updating their parameters, the opportunity to expand **Workflow**

Automation would streamline user experience even farther. Expanding the intelligence of the model through further Python exploration, the model would be able to handle entire chains of command rather than being limited to single-sentence spoken prompts.

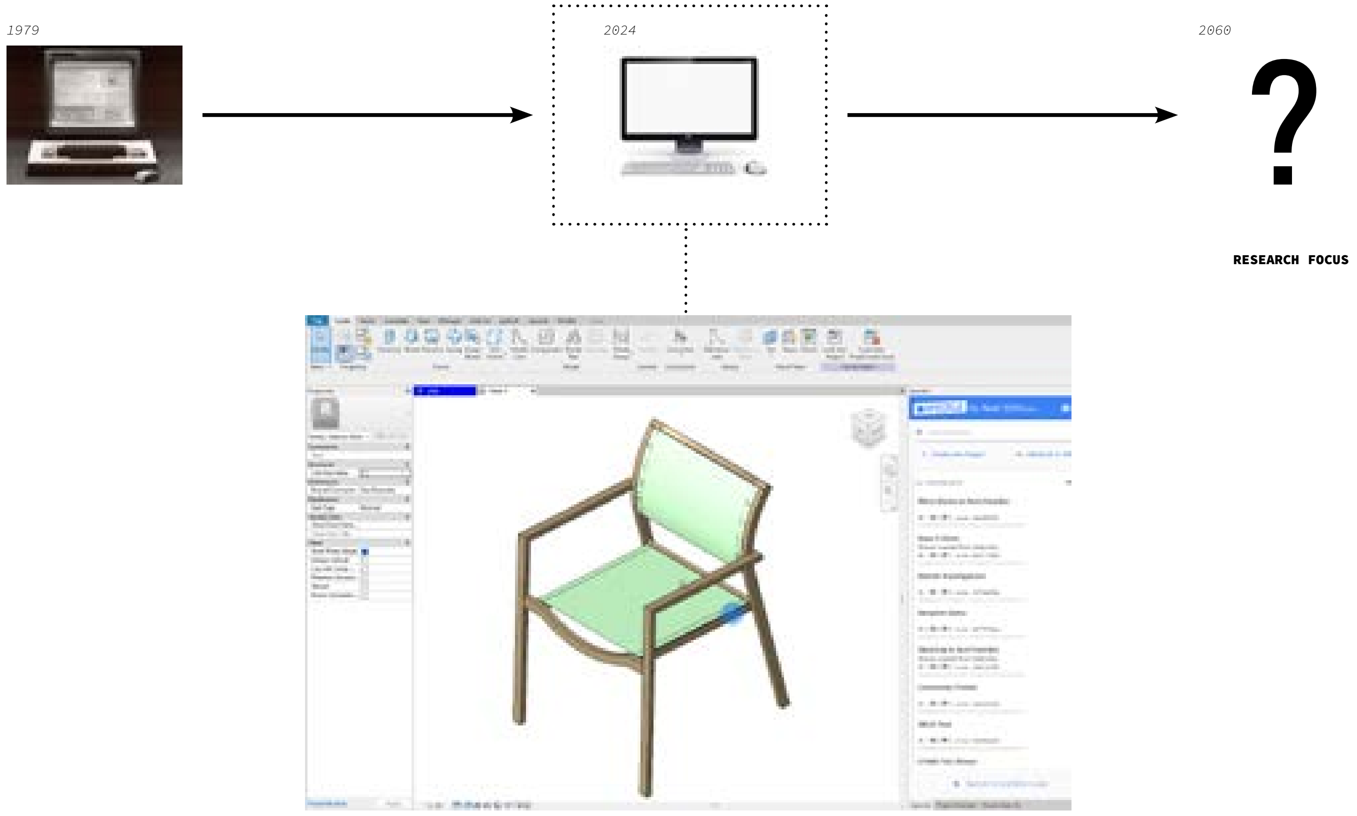
Finally, the task of **Scheduled Maintenance.** Keeping with regular updates to keep concurrent with adapting technologies and softwares will be essential in further developments of the tool.

Conclusion

The integration of Speech Recognition within Revit, powered by Python and ML, will dramatically increase the efficiency of designers. Not only will the opportunity to let Human-Centered UI drive advancements, designers will also be able to streamline their projects, saving time and money.



# 01 PROBLEM



Unchanging User Experience

Despite decades of technological advancement, the core user experience between man and computer has remained nearly unchanged.

This tedious and repetitive process requires designers to spend extreme amounts of time that could be saved through updated methods.

This Thesis explores a potential future for computer user experience, utilizing AI language models to interpret and execute commands.

# 02 SOLUTION

Voice 2 Command

Instead of the traditional method of manual CAD, the proposed combination of Speech Recognition, Python Coding, and AutoDesk Revit can **provide an alternative and more effective method of automated CAD.**

The automation of Revit commands through spoken prompts will lead to reduced design time required by Revit users. This will in turn produce an accelerated iteration process and faster turn around on future projects.

Speech Recognition + Python + Revit

STATE OF THE ART

Voice-to-Command

When considering current available technologies to consider as precedents for this research thesis, successful uses of Spoken Prompts translated into Computer Commands will be crucial. One impressive existing method is the Voice2CAD program. This is a stand alone software that allows users to program their own "spoken shortcuts" or "hot keys". While Voice2CAD does successfully interpret spoken prompts, the software specializes in dimension conversion; For example, instead of typing out "L2 [1/2 x 1 1/2 x 48] x (3)" to specify the dimensions of 3 members all labeled "L2", the user can simply speak that command: " Change the three L2 members to be one half by one and one half by 48." While this does save time in manual typing, this is a very specific case that has many limitations.

Another example of voice powered commands is the offering of Voice Recognition by AutoDesk within Fusion360. However, this product is still in its infancy with many limitations and inefficient results.

Natural Language Processing

In addition to Speech Recognition technologies offered by third party software, it is important to understand the Natural There are many NLP systems that can be utilized within Python code for Speech Recognition based on needs and uses. However, for the scope of this thesis, we will focus on three popular and effective NLP systems: nltk, spaCy, and Google Speech-Recognition.

While Google's Speech Recognition technology is extremely effective, the use of the product requires an active Internet connection as well as importing of specific Google dependencies. Because the goal of this product is to use the minimum required outlying dependencies and servies, this avenue will be effective but not the main implementation to allow for maximum ease during user experience.

After researching the NLP spaCy, it became clear that this system was not only capable of easily running within a Python environment, but also extremely useful in terms of handling large amounts of information. However, due to its large amount of requires dependencies to be installed within the Python environment, the use of spaCy would likely lead to a drastic increase of CPU usage and time required to run.

Preliminary research and testing with the nltk NLP seemed to yield the best results. Because nltk requires minimal installed dependencies

AI-Enhanced Modeling

Finally, when looking at precedents for artificial intelligence (AI) influenced programs that produce CAD models, AutoDesk's Fusion360 once again offers an example. While Fusion360 boasts Generative Design aided by AI within the native UI, it comes with its own limitations: The generative design tool is limited to basic functions and commands unless the user pays for additional subscriptions. In addition, the process remains reliant on the out dated manual CAD process.

Precedent Findings

When considering the current availabilities to designers, Speech Recognition and AI are being utilized but in very specific use cases with limited capabilities.

Taking these precedents and expanding upon their capacities will not only prove useful to this thesis, but future research as well.

# 03 METHODOLOGY

Divided Technology Stack

Dom - Python / Voice Tokenizer

Speech-to-Text

Audio / Prompt Recording

Text to Variables

Traditional coding method

Speech Tokenizers

Testing the performance of various speech tokenizers

Rhino.Inside Tests

Testing methodologies utilizing Rhino. Inside to execute the commands

James - Revit / LLM

Revit API

Controls Revit application through Custom Scripting

Preliminary Research

PyRevit

Running Python inside of the Revit environment

LLM / RAG

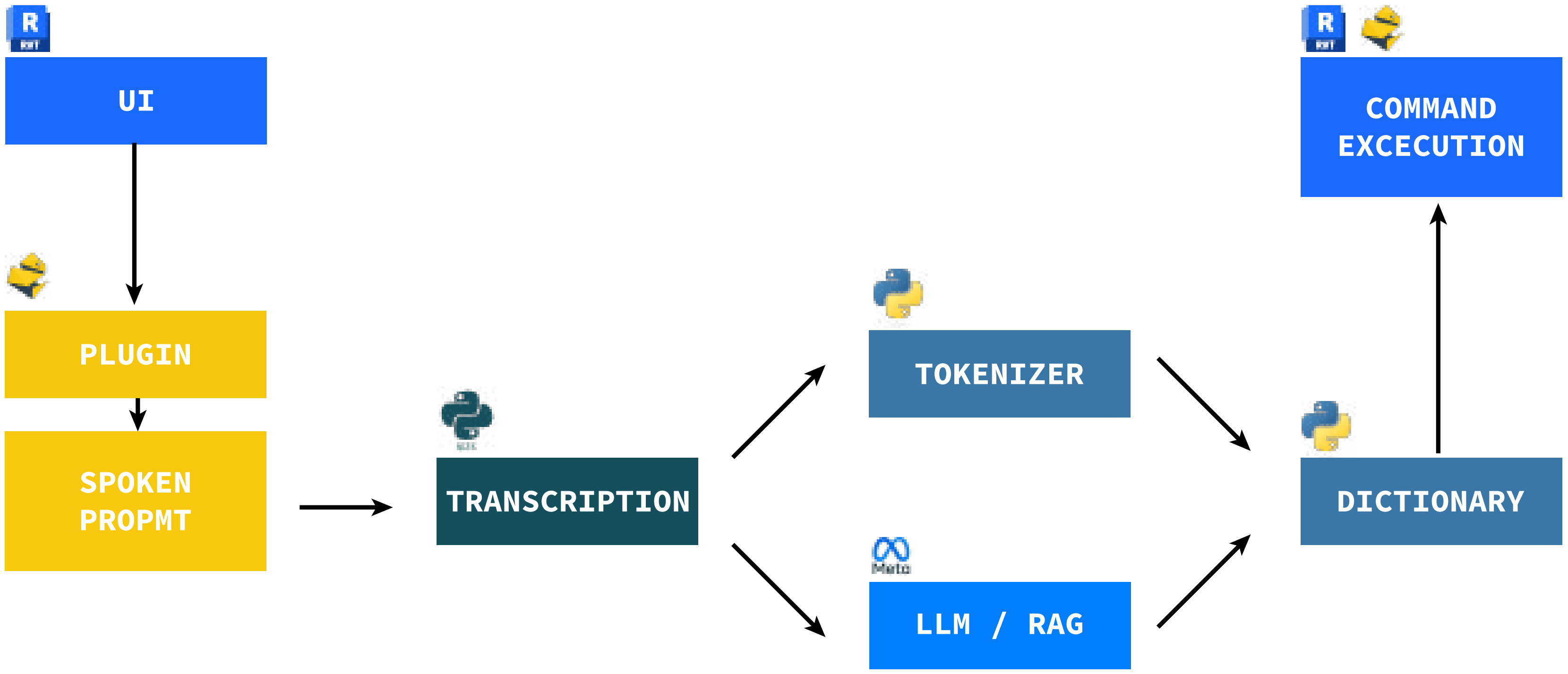
Encoding Revit file for LLM

RAG retrieval of elements

Due to the scope of the thesis, encompassing both manually written Python code and Revit API knowledge, we determined that our time would most effectively be spent divided.

With both team members work on opposite ends of the technology stack, task items could quickly be accomplished alongside one another while working toward a common goal. This also allowed for effective collaboration to determine best paths forward for both James and Dom during weekly catch up meetings.

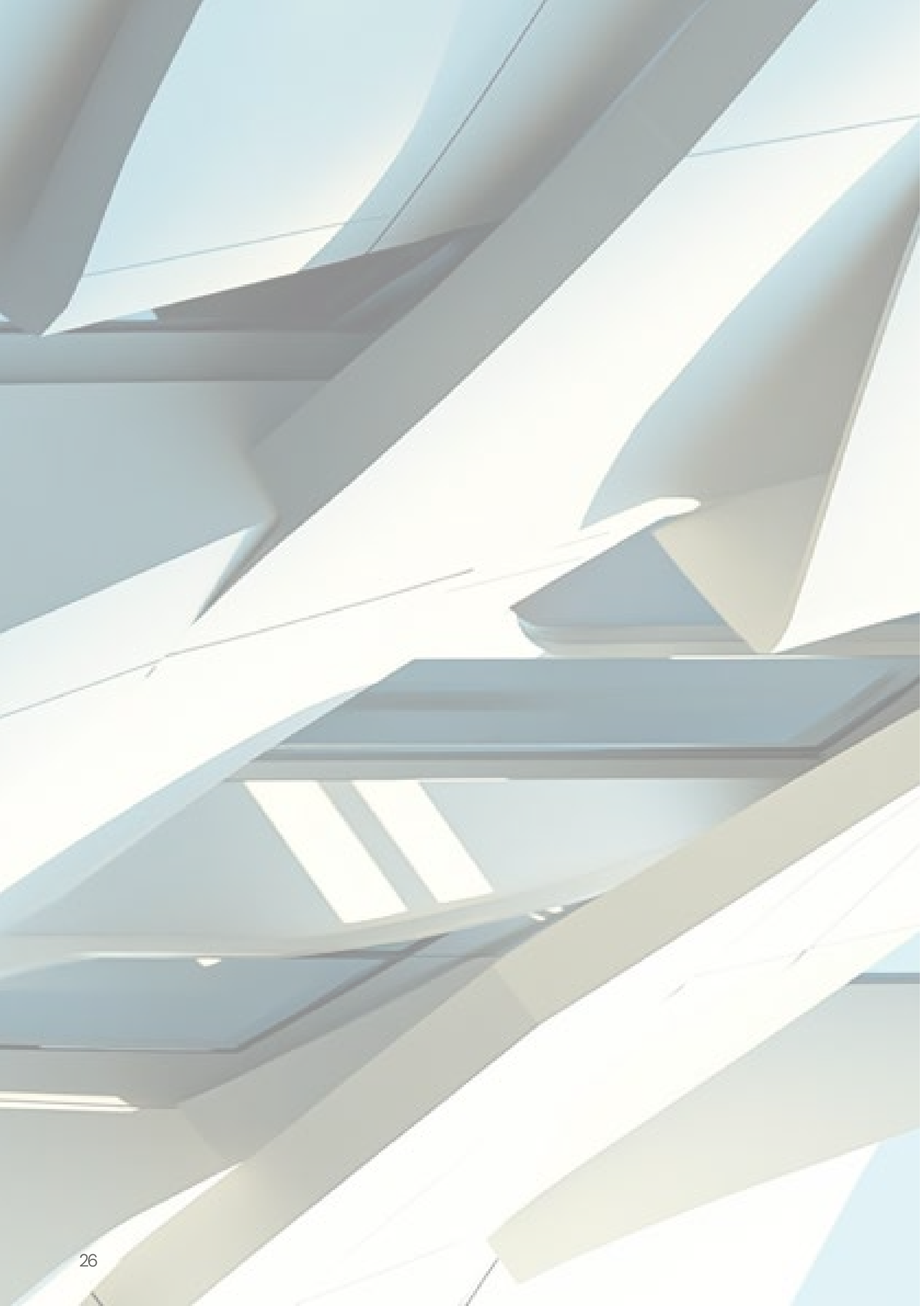
Workflow Pseudo-Code



The diagram above shows the intended workflow process that will be executed during the final process. While being run inside the Revit environment, first the plug-in is installed via placing the dependent Python Notebooks into the Revit install directory. Once installed, the user will be able to easily record their prompt using their machine's microphone.

After successful installation and prompt recording, the plug-in then utilized the use of the nltk NLP in order to transcribe the user's spoken prompt. Once transcribed, one of two separate actions can be performed to create a usable Revit Dictionary: either Python is used to tokeninze specific phrases that are predefined, or a large language model (LLM) can be used in combination with a retrieval augmented generation (RAG) search to handle larger amounts of information.

Once the dictionary is created via one of the two processes, Revit is able to receive the dictionary and interpret its contents as commands and execute them.

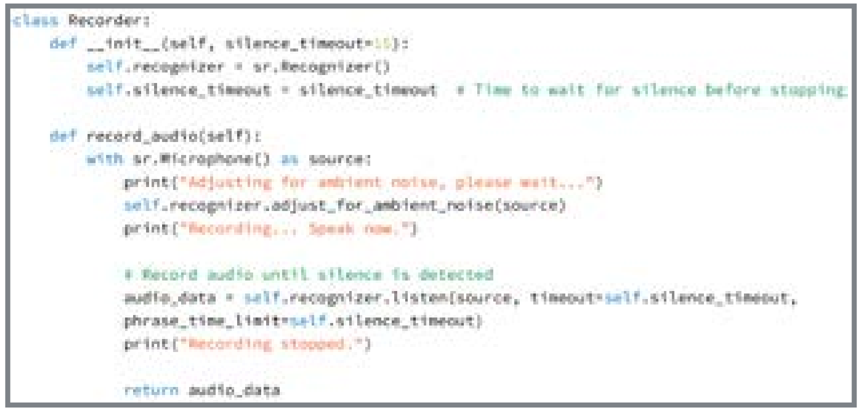


# 04 Transcription

NLTK

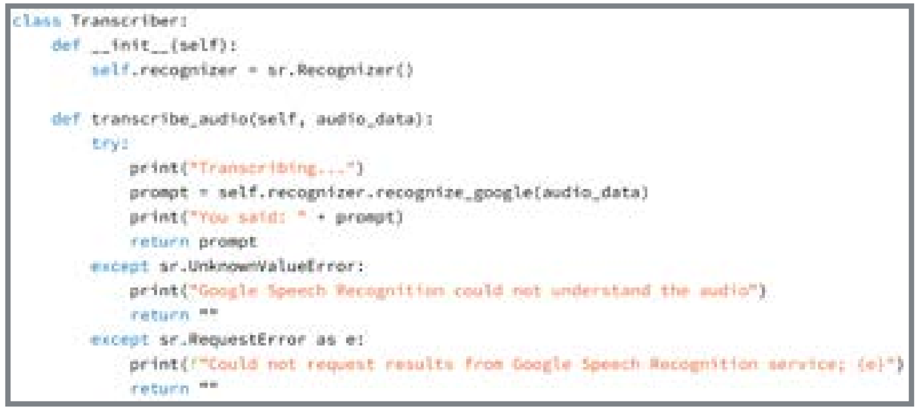
Transcription via NLTK

Recorder Class



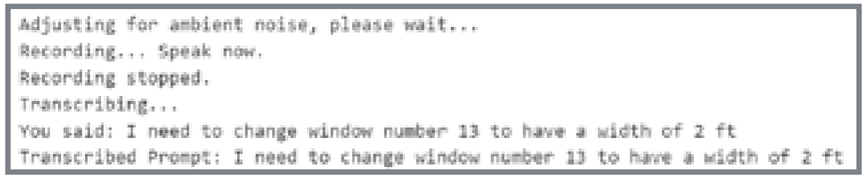
```python
class Recorder:
    def __init__(self, silence_timeout=15):
        self.recognizer = sr.Recognizer()
        self.silence_timeout = silence_timeout  # Time to wait for silence before stopping

    def record_audio(self):
        with sr.Microphone() as source:
            print("Adjusting for ambient noise, please wait...")
            self.recognizer.adjust_for_ambient_noise(source)
            print("Recording... Speak now.")

            # Record audio until silence is detected
            audio_data = self.recognizer.listen(source, timeout=self.silence_timeout,
            phrase_time_limit=self.silence_timeout)
            print("Recording stopped.")

            return audio_data
```

Transcriber Class



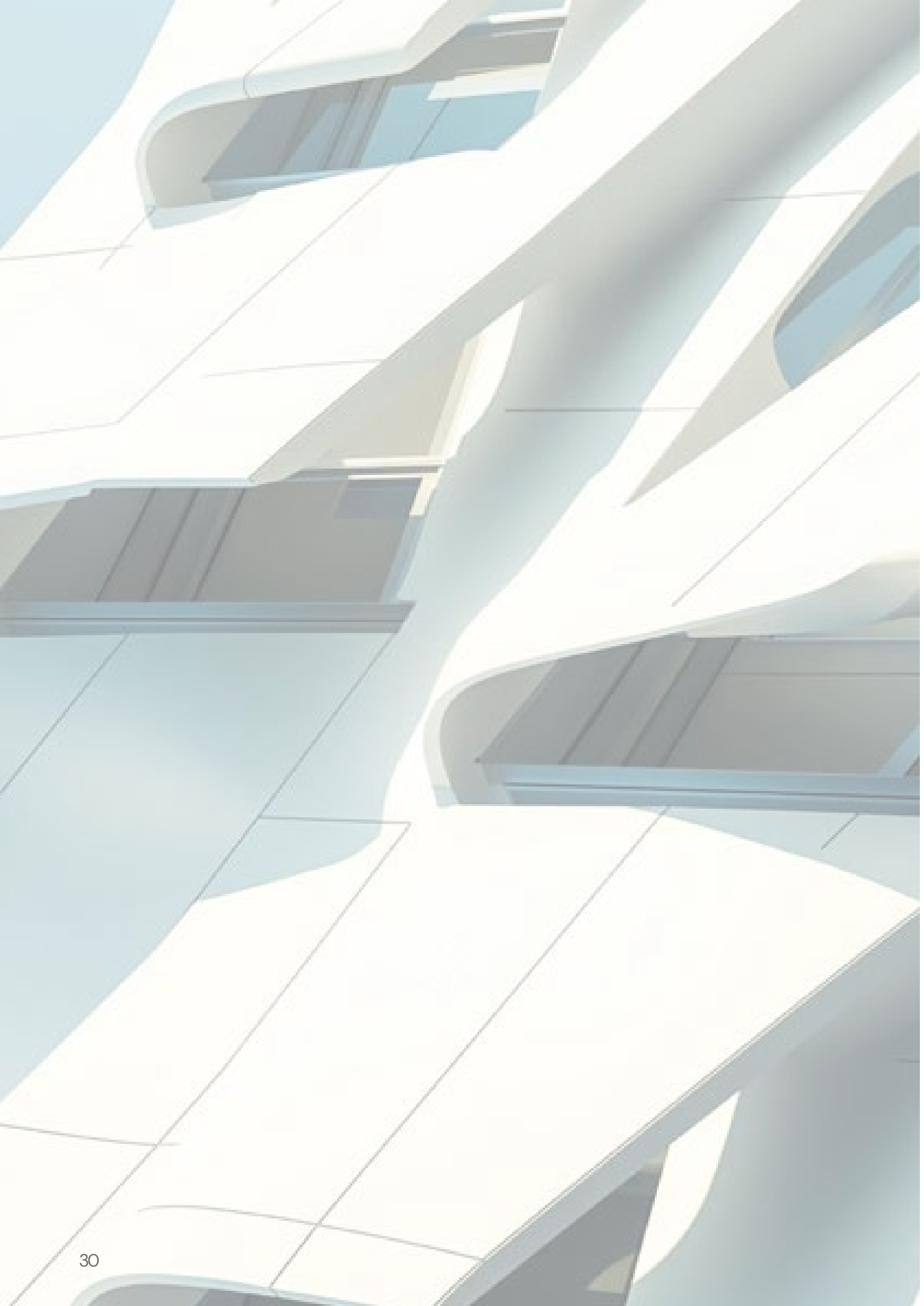
```python
class Transcriber:
    def __init__(self):
        self.recognizer = sr.Recognizer()

    def transcribe_audio(self, audio_data):
        try:
            print("Transcribing...")
            prompt = self.recognizer.recognize_google(audio_data)
            print("You said: " + prompt)
            return prompt
        except sr.UnknownValueError:
            print("Google Speech Recognition could not understand the audio")
            return ""
        except sr.RequestError as e:
            print(f"Could not request results from Google Speech Recognition service; {e}")
            return ""
```

Transcription Output



```
Adjusting for ambient noise, please wait...
Recording... Speak now.
Recording stopped.
Transcribing...
You said: I need to change window number 13 to have a width of 2 ft
Transcribed Prompt: I need to change window number 13 to have a width of 2 ft
```

The diagram above shows the intended workflow process that will be executed during the final process. While being run inside the Revit environement, first the plug-in is installed via placing the dependent Python Notebooks into the Revit install directory. Once installed, the user will be able to easily record their prompt using their machine's microphone.



# 05 Tokenizer

Tokenizer Process Outline

DEPENDENCIES

nltk
pyaudio / pydub
SpeechRecognition
word2number

RECORDER CLASS

Ambient Noise Detection

“Recording... Speak Now”

“Recording Stopped”

TRANSCRIBER CLASS

Speech Recognition
“You Said ________”

SPOKEN PROMPT

Adjust for Ambient Noise

You Said: “I need to change window number 13 to have a width of two feet instead of three”

EXTRACT VARIABLES

“I need to **change window** numbier 13 to have a **width of two feet** instead of three.”

DICTIONARY

```
{
“Modify”
“Window: 13”
“Width: 2 ft”
}
```

The outline above illustrates the order of the Python code created to prioritize tokenizing specific words within a spoken prompt. This process ensures that viable parameters and their values can be pulled from the prompt and used within Revit. First, the necessary Dependencies are installed, enabling nltk and the corresponding requirements for Speech Recognition.

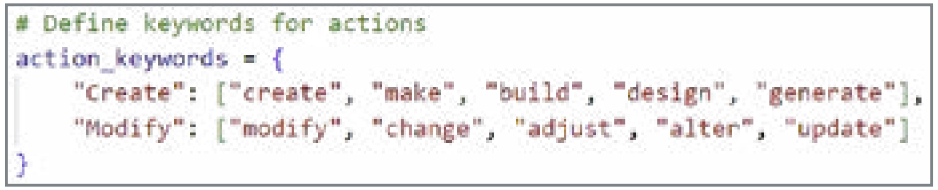
Next, the Recorder and Transcriber classes are defined; these classes allow for the Python environment to utilize the machine’s microphone in order to interpret what the user has spoken, and then translate that to a coded variable. The classes even offer options to update User Experience (UI) prompts such as: “Recording… Speak now” and “Recording Stopped”.

Once the two classes are defined, the user is able to proceed with speaking there prompt for their Revit command. After being activated, the code prompts the user to be silent to account for ambient noise before allowing the recording to begin. The model is also trained to know when the user has stopped speaking according to comparison with the measured ambient noise. Finally, the variable “prompt” is displayed and cached in the code, ready to be referenced.

After the variable is created, the tokenized values are compared to the variable to identify the parameters that need to be adjusted, as well as their corresponding values. Finally, all of this information is organized and cached within a JSON dictionary that is readable by Revit.

Variable Identification

Create or Modify



```
# Define keywords for actions
action_keywords = {
    "Create": ["create", "make", "build", "design", "generate"],
    "Modify": ["modify", "change", "adjust", "alter", "update"]
}
```

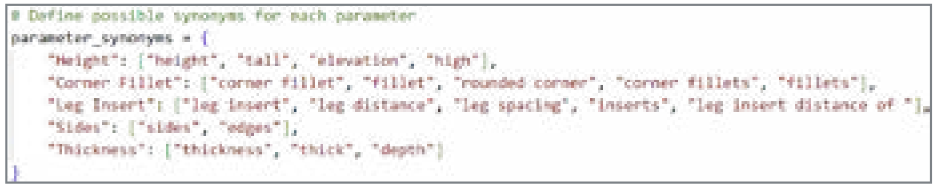
Input:
Processing text: “I need to update tables 5, 11, and 17”

Output:
Detected Action: “Modify”
Modified IDs: [5, 11, 17]

Because the approach uses ‘token’ words to identify variables and their values, those variables need to be identified.

The Python code above shows the defined token words to identify wether an item is to be Created or Modified. However, in order to expand the capabilities of the code, additional token words are included for each of the two options. This ensures that users do not have to adhere to the same prompt each time a command is spoken. Of course, these values are able to be updated for further expansion.

Variable / Value Identification



```
# Define possible synonyms for each parameter
parameter_synonyms = {
    "Height": ["height", "tall", "elevation", "high"],
    "Corner Fillet": ["corner fillet", "fillet", "rounded corner", "corner fillets", "fillets"],
    "Leg Insert": ["leg insert", "leg distance", "leg spacing", "inserts", "leg insert distance of "],
    "Sides": ["sides", "edges"],
    "Thickness": ["thickness", "thick", "depth"]
}
```

Input:
Processing text: “I need to update tables 5, 11, and 17”

Output:

```
{
        “Height”: “3 feet”,
        “Corner Fillet”: “5.25 inches”,
        “Leg Insert”: “1.25 inches”,
        “Sides”: “3.25 feet”,
        “Thickness”: “0.75 inches”
}
```

Once the code identifies wether or not elements are to be created or modified, the variables themselves are identified. Similarly to the action keywords, parameter keywords are defined as well as their possible synonyms.

Once identified with their values, Parameters are organized in an easy to understand format, both for humans and Revit.

Dictionary Creation

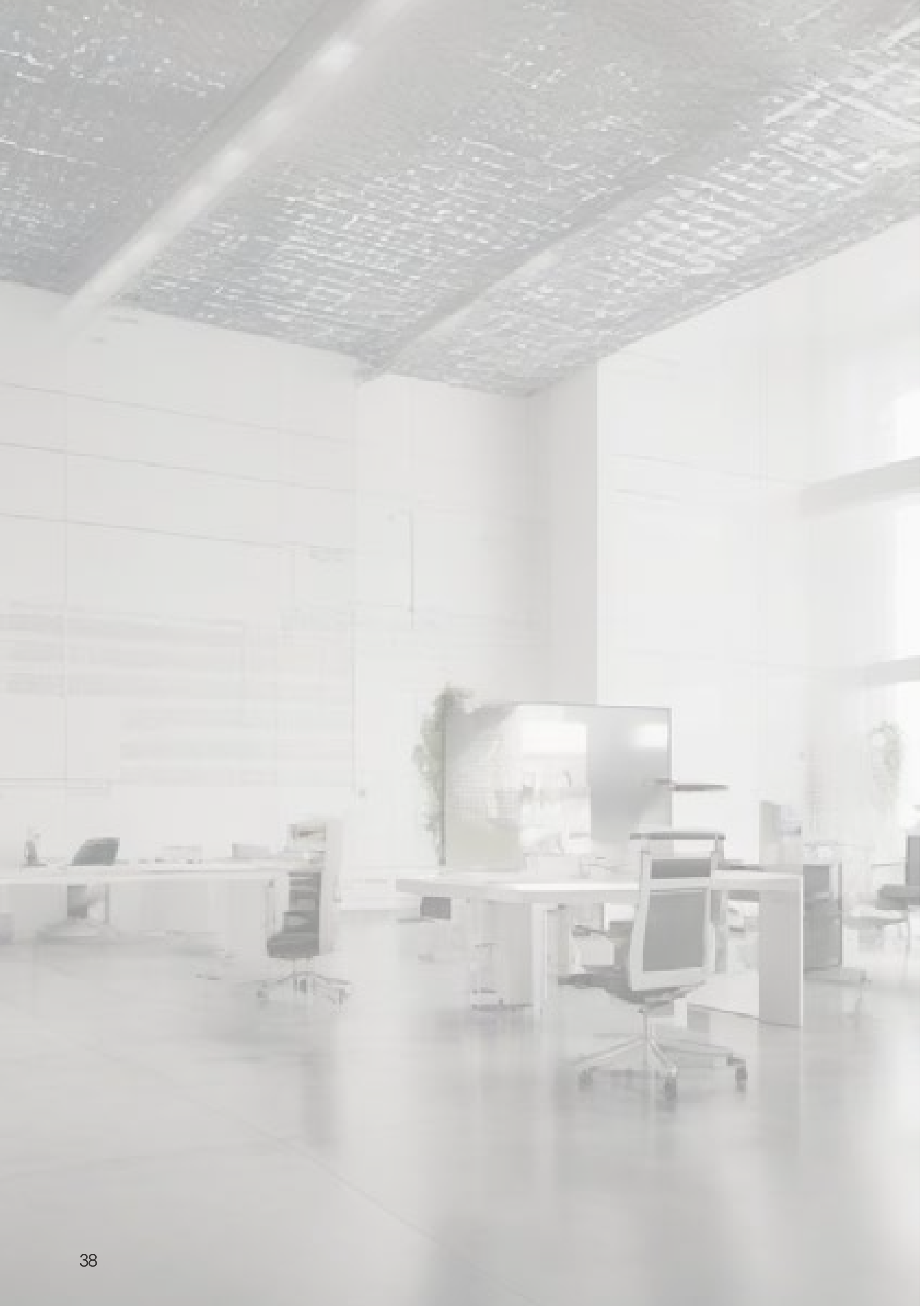
Create

```
{
        “Action”: “Create”, ..............................................
        “Table IDs”: [],....................................................
        “Parameters”: {.....................................................
                “Height”: {
                “ParameterName”: “Height”,
                “Value”: “3”,
                “Unit”: “feet”
        },
        “Thickness”: {
                “ParameterName”: “Thickness”,
                “Value”: “0.75”,
                “Unit”: “inches”
        },
        “Sides”: {
                “ParameterName”: “Number of
Sides”,
                “Value”: “3.25”,
                “Unit”: “feet”
        },
        “Leg Insert”: {
                “ParameterName”: “Leg Insert”,
                “Value”: “1.25”,
                “Unit”: “inches”
        },
        “Corner Fillet”: {
                “ParameterName”: “Corner
Fillet”,
                “Value”: “5.25”,
                “Unit”: “inches”
        }
}
```

Modify

```
{
    “Action”: “Modify”, ........................... ACTION
    “Table IDs”: [5, 11, 17], ..................... ELEMENT IDS
    “Parameters”: {................................ PARAMETERS
                “Height”: {
                “ParameterName”: “Height”,
                “Value”: “3”,
                “Unit”: “feet”
        },
        “Thickness”: {
                “ParameterName”: “Thickness”,
                “Value”: “0.75”,
                “Unit”: “inches”
        },
        “Sides”: {
                “ParameterName”: “Number of
Sides”,
                “Value”: “3.25”,
                “Unit”: “feet”
        },
        “Leg Insert”: {
                “ParameterName”: “Leg Insert”,
                “Value”: “1.25”,
                “Unit”: “inches”
        },
        “Corner Fillet”: {
                “ParameterName”: “Corner
Fillet”,
                “Value”: “5.25”,
                “Unit”: “inches”
        }
}
```

Once all the information is collected and ready, the Python code is then trained to organize and export the informatin in a Revit friendly dictionary. This ensures that the commands within Revit will successfully receive the information necessary to make updates based on the Spoken Prompt.



# 06 LLM / RAG

Create

```
{
        “Action”: “Create”, ..........................................
        “Table IDs”: [], .............................................
        “Parameters”: { ..............................................
                “Height”: {
                “ParameterName”: “Height”,
                “Value”: “3”,
                “Unit”: “feet”
        },
        “Thickness”: {
                “ParameterName”: “Thickness”,
                “Value”: “0.75”,
                “Unit”: “inches”
        },
        “Sides”: {
                “ParameterName”: “Number of
Sides”,
                “Value”: “3.25”,
                “Unit”: “feet”
        },
        “Leg Insert”: {
                “ParameterName”: “Leg Insert”,
                “Value”: “1.25”,
                “Unit”: “inches”
        },
        “Corner Fillet”: {
                “ParameterName”: “Corner
Fillet”,
                “Value”: “5.25”,
                “Unit”: “inches”
        }
}
```

Modify

```
{
    “Action”: “Modify”, ........................ ACTION
    “Table IDs”: [5, 11, 17], .................. ELEMENT IDS
    “Parameters”: { ............................ PARAMETERS
                “Height”: {
                “ParameterName”: “Height”,
                “Value”: “3”,
                “Unit”: “feet”
        },
        “Thickness”: {
                “ParameterName”: “Thickness”,
                “Value”: “0.75”,
                “Unit”: “inches”
        },
        “Sides”: {
                “ParameterName”: “Number of
Sides”,
                “Value”: “3.25”,
                “Unit”: “feet”
        },
        “Leg Insert”: {
                “ParameterName”: “Leg Insert”,
                “Value”: “1.25”,
                “Unit”: “inches”
        },
        “Corner Fillet”: {
                “ParameterName”: “Corner
Fillet”,
                “Value”: “5.25”,
                “Unit”: “inches”
        }
}
```

Once all the information is collected and ready, the Python code is then trained to organize and export the informatin in a Revit friendly dictionary. This ensures that the commands within Revit will successfully receive the information necessary to make updates based on the Spoken Prompt.

# 07 Revit

list of references, books, people, ins-titutions, web links etc related to the project

for fast and correct form of referencing, follow this link:

https://www.citethisforme.com/harvard-referencing

# 08 Conclusions

CONCLUSION

TITLE

am quos et qui adias utectat ullati verae laccuptam fugit, odit et parum rempos et aborernam et et as quam, voluptios poreperum rerat quassum que nihilit aepedit iatur, ulparum accus.
To doloribuscim non parita quost eumReris conseque imi, corrovid quid quid unt, to ipis aut et elest quae eatia nobis maxima quiata quiam, sus audipid que cum con eostium quiam faccuscient alicatur autas eicienditiis doluptat pero omnis doloris cipiendi dio di aciam ento eum nulliaectio. Uptatum aut eati nulpa conseque de nat.
Debitas repro qui desequam enis aut es sitin rehenientem et mo mo derum imoluptae porite labo. Ovid quam nis sim repe pelestisit fugit offic tem que rerspic iasped qui dolorate num quamus corerum is qui as millendam, alia de nat.
Sequistrum nobis iur? Ecatet fugit ipsanis aut que volessita doluptatur, quae explis et, suntur sinvel idunt harchicidunt volorum erferib

am quos et qui adias utectat ullati verae laccuptam fugit, odit et parum rempos et aborernam et et as quam, voluptios poreperum rerat quassum que nihilit aepedit iatur, ulparum accus.
To doloribuscim non parita quost eumReris conseque imi, corrovid quid quid unt, to ipis aut et elest quae eatia nobis maxima quiata quiam, sus audipid que cum con eostium quiam faccuscient alicatur autas eicienditiis doluptat pero omnis doloris cipiendi dio di aciam ento eum nulliaectio. Uptatum aut eati nulpa conseque de nat.
Debitas repro qui desequam enis aut es sitin rehenientem et mo mo derum imoluptae porite labo. Ovid quam nis sim repe pelestisit fugit offic tem que rerspic iasped qui dolorate num quamus corerum is qui as millendam, alia de nat.
Sehicidunt volorum erferiberumque qui tem etur aut quidebissi

REFERENCES

am quos et qui adias utectat ullati verae laccuptam fugit, odit et parum rempos et aborernam et et as quam, voluptios poreperum rerat quassum que nihilit aepedit iatur, ulparum accus. To doloribuscim non parita quost eumReris conseque imi, corrovid quid quid unt, to ipis aut et elest quae eatia nobis maxima quiata quiam, sus audipid que cum con eostium quiam fac cuscient alicatur autas eicienditiis doluptat pero omnis doloris cipiendi dio di aciam ento eum nulliaectio. Uptatum aut eati nulpa conseque de nat. Debitas repro qui desequam enis aut es sitin rehenientem et mo mo derum imoluptae porite labo. Ovid quam nis sim repe pelestisit fugit offic tem que rerspic iasped qui dolorate num quamus corerum is qui as millendam, alia de nat. Sequistrum nobis iur? Ecatet fugit ipsanis aut que volessita doluptatur, quae explis et, suntur sinvel idunt harchicidunt volorum erferib uscimusam quiandio. Lor re, quatentempor aut aut quati tota essitia nihil ium ariosse nderspe lluptat umquam, et maximax imincim quia nos parciendus essumendi conserf erumque qui tem etur aut quidebissi

MASTER IN ADVANCED COMPUTATION FOR ARCHITECTURE AND DESIGN

MaCAD - Thesis

2024

REVITvoice

Vocal Commands for AutoDesk Revit

BARCELONA

Iaac

Institute for advanced architecture of Catalonia