1. **Step 1: Chosen NLP technique and rationale.**

This was comparatively easier, since top-performing LLMs are easily available. I did not have to do anything from scratch. NER models had limited 'generalization' ability, and bigger models performed better.

Tool/Model	Reason for Trying	Outcome	Decision
SpaCy	Lightweight, efficient features like NER and dependency parsing.	- Struggled with specific queries (e.g., object types, features).	Not suitable; lacked generality and customization.
Hugging Face Transformers (Flan-T5-small, Flan-T5-large)	Open Source. Wide range of models and performance.	- Flan-T5-small: Inconsistent, generic outputs. - Flan-T5-large: Performed ok after prompt refinement;	Chose Flan-T5-large for its flexibility and strong performance. Identified Llama-7-b as best, but didn't try due to storage, compute constraints. Others mentioned in code.
OpenAI GPT 4	Closed Source. Best performance.	- High-quality results. - Expensive for prototyping.	Did not use due to cost; should prioritize for production. Did predict correctly.

2. **Step 2: Building 3D Mesh**

I searched for multiple papers that could help me create a 3D mesh (useful for engineering , and not just stylistic). A list of papers is in Literature Review.

The final comparison, came between Point-E and CLIP-Forge. Used Point-E for quick prototyping, which uses diffusion based models.

If I had more time, I would have preferred to deep dive into CLIP-Forge architecture and build something similar, using a VAE trained on a large dataset (such as ShapeNet), and encode it into latent vectors. During reconstruction, I would get 3D shapes such as voxels, SDFs and I could have optimized the code more directly in the next 3 steps, since the gradients are more easily available.

On the other hand, in Point-E, the better model is when we divide into prompt -> image -> point cloud -> mesh, however I believe optimization/interpretability would be difficult, along multiple models.

So I used, a worse performing model : prompt -> point cloud -> mesh.

Criteria	Point-E	CLIP-Forge
Ease of Use	☆☆☆☆☆	☆☆☆
Output Quality	☆☆☆	☆☆☆☆
Resource Requirements	☆☆☆☆☆	☆☆☆
Suitability for Refinement	☆☆☆	☆☆☆☆
Setup Complexity	☆☆☆☆☆	☆☆
Overall Applicability	Great for quick prototypes.	Better for precise, detailed meshes.

Step 3: Optimizing based on User Feedback

We needed to convert point clouds to something more concrete for 'engineering', due to choosing Point-E.

The first decision was to choose between Voxels vs SDFs vs Meshes vs Parametric Surfaces (NURBS) VS CAD. **Chose voxels. SDFs would have also been a good prototype choice. CADs are difficult, but best for robust design.**

- **Voxels**: Great for volumetric representation but memory-intensive.
- **SDFs**: Compact and smooth but computationally demanding.
- **Meshes**: Most versatile for general-purpose 3D work; limited for smooth, high-precision surfaces.
- **NURBS/Parametric Surfaces**: Precise and compact, ideal for smooth curves; harder for general shapes.
- **CAD**: Best for engineering and manufacturing due to its precision and integration with real-world constraints.

This step is unfinished. The idea was to design basic 'code blocks' which can do specific things, such as, scale along a direction (to change mass), create holes, etc. Then maybe an NLP block could identify key user requests and parameters, and those can be used as inputs to these functions.

Example Flow : user says to reduce weight by 15% ->

NLP model from Step 1 extracts 'function':'reduce_weight', 'VALUE'=0.15 ->

NLP Similarity function matches it to most 'similar' function in our list and its parameter value ('reduce_weight':'scale_down(),'VALUE'='scale_down')->

Function runs scale_down(voxel, scale_down=VALUE)

The idea here is a mix between being 'general' and 'specific, but robust' and only entertaining a specific set of requests.

The more general approach would be to train the model directly on feedbacks through steps 2 to 4, with RL (maybe RLHF)