# User Interface Code Generation from Hand-drawn Sketch


**Team Members**

Manoj Paudel (THA077BCT025)
Prince Poudel (THA077BCT036)
Ronish Shrestha (THA077BCT040)
Sonish Poudel (THA077BCT042)

**Supervised By:**

Er. Devendra Kathayat

Department of Electronics and Computer Engineering
Institute of Engineering, Thapathali Campus


August, 2024

# Presentation Outline

- Motivation
- Problem Statement and Objectives of Project
- Scope of Project
- Project Applications
- Methodology
- Results
- Discussion of Results
- Remaining Tasks
- References

# Motivation

- Converts sketches to code which saves development time.
- Enables non-programmers to create software easily.
- Facilitates rapid prototyping and creative iteration.

# Problem Statement

- Conversion of sketches into function GUI code is time-consuming, error-prone process which requires significant technical expertise.

# Objectives of Project

- To construct a model able to generate quick User Interface.
- Creation of intuitive User interface for customization and styling of generated code.

# Scope of Project

- Accurately interpret and process hand-drawn sketches.
- Convert recognized sketches into functional code automatically.
- Develop a user-friendly platform for sketch-to-code conversion.
- Ensure compatibility with various coding environments.
- Enhance efficiency and accuracy of the generated code.

# Project Applications

- It can be used in rapid prototyping for various industries like healthcare, finance and retails etc.
- It can be used in hackathons to create functional prototypes quickly.
- It can be used in startups to create functional prototypes and MVPs quickly.

# Methodology - [1] (Datasets)

- Dataset is a wireframe sketch and associated DSL code.
- We were not aware of any dataset which contained wireframes sketches and DSL code
- We will create our own dataset.

# Methodology - [2] (DSL)

- Specialized language designed to address specific aspect or needs of a particular language.
- Designed the simple lightweight DSL to describe the GUI.
- Elements in DSL will be categorized into different hierarchical structures.

# Methodology - [3] (DSL elements)

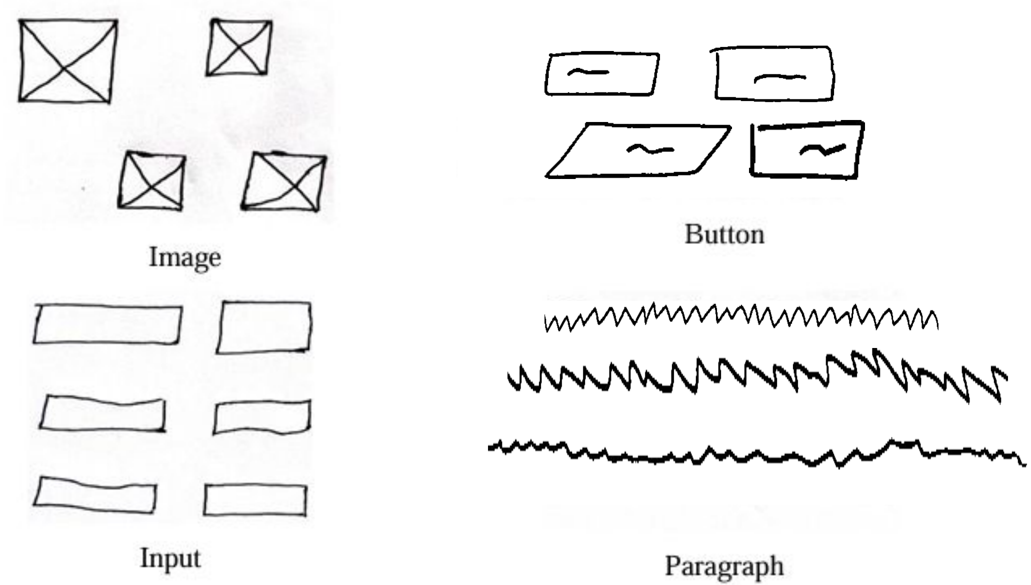
- Body
- Root
- Header
- Nav
- Navlink
- Logodiv
- Container
- Row
- Div-3
- Div-6
- Div-9
- Div-12
- Flex
- Flex-sb
- Flex-c
- Flex-r
- Text
- Text-c
- Text-r
- Paragraph
- Image
- Card
- Input
- Button
- Button-c
- Button-r
- Footer

# Methodology - [4] (DSL code)

```
header{
	flex-sb{
		logodiv{
			image
		}
		nav{
			navlink
			navlink
			navlink
		}
	}
}
container{
	row{
		div-3{
			card{
				button-c
				input
			}
		}
		div-3{
			button
			paragraph
		}
	}
}
```

Figure: DSL code

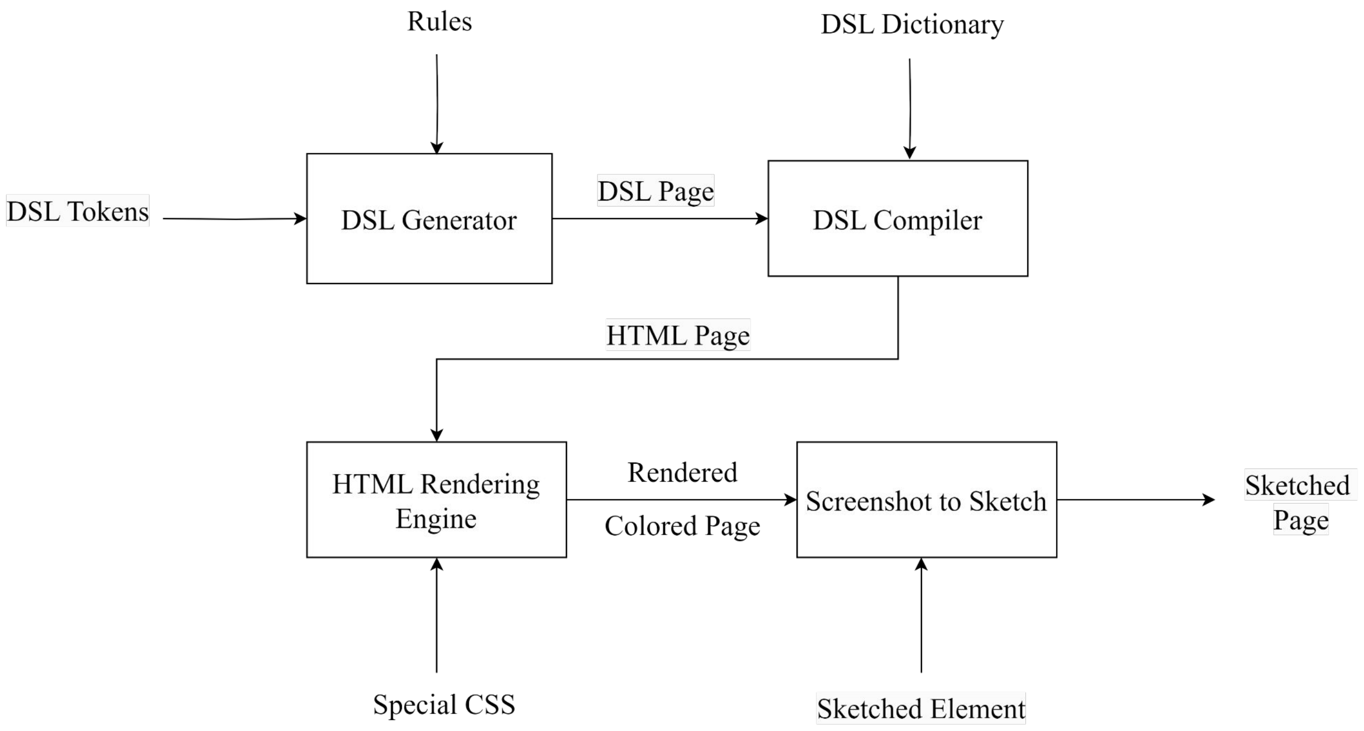
# Methodology - [5] (Sample elements)



# Methodology - [6] (Dataset Generation Process)

- Dataset is the hand-drawn sketch and it's associated dsl.
- There is no existing dataset for it.
- Dataset generator is created to create as many dataset as required.

# Methodology - [7] (Dataset generator)



# Methodology - [8] (Random DSL Generator)

- First step is to generate a DSL randomly.
- It denotes the elements in the user interface.
- DSL is generated by mixing the different possible combination of the element.

# Methodology - [9] (Compiling the DSL code)

- Randomly generated DSL code is mapped into the corresponding HTML tag.

# Methodology - [10] (Rendering the Produced HTML)

- Mapped HTML file is rendered into the webpage with the special CSS file.
- CSS file helps to denote the different element with separate colour.

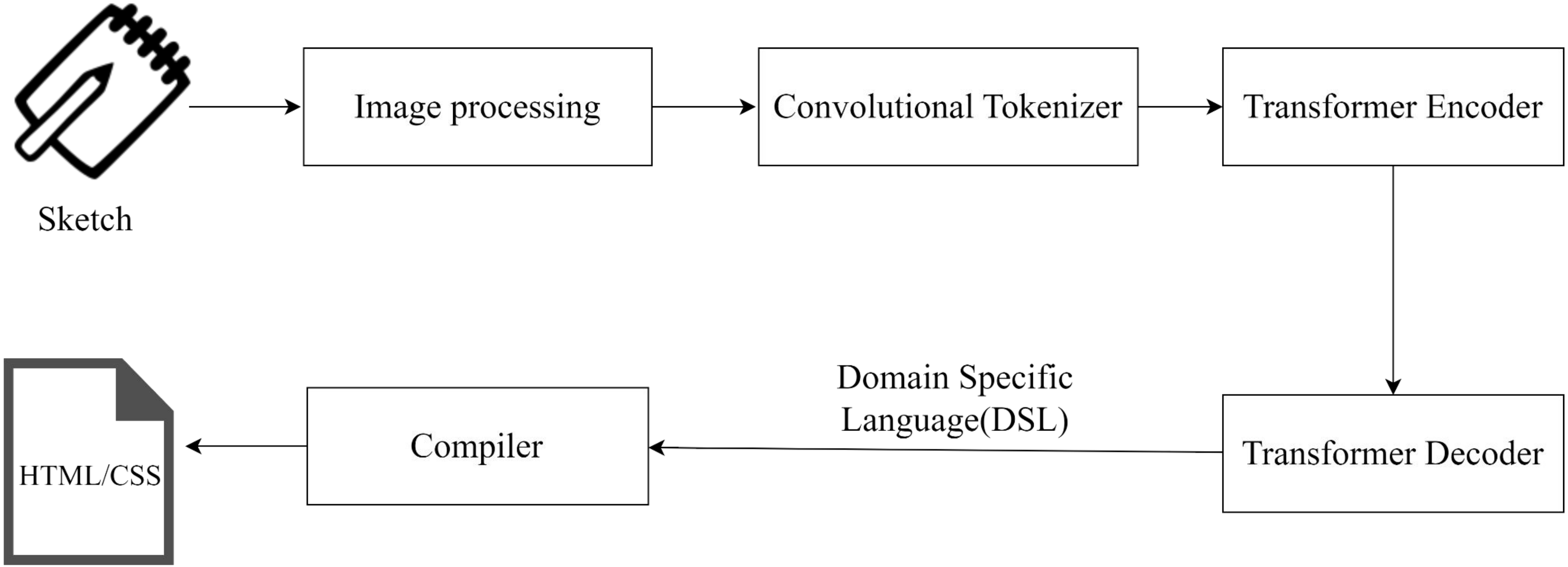
# Methodology - [11] (Finding outline of elements)

- Contour detection is applied to find the boundary of all the elements
- Then, the position is identified.

# Methodology - [12] (Placing hand-drawn sketch of element)

- Hand-drawn sketch of all the element is place at the position identified from above step.
- At last, sketch is obtained with its associated DSL code.

# Methodology - [13] (System Block Diagram)



# Methodology - [14] (Convolutional Tokenizer)

- Convert images into token sequences
- Apply convolutions → Pooling → Feature maps → Flatten
- Key Components are Convolutional layers, Pooling layers
- Output is Sequence of vectors representing image features
- Bridges visual data to sequence models

# Methodology - [15] (Transformer Encoder)

- Designed for NLP, now also used in computer vision.
- Multiple identical layers with self-attention and feed-forward sub-layers
- Layer normalization and residual connections for improved training

# Methodology - [16] (Transformer Decoder)

- Consists of four identical transformer blocks.
- Each block has masked self-attention, cross attention, and feed-forward sublayers.
- Adds positional information to word embeddings.
- Uses the output of the last decoder block to predict the next word via a linear layer.

# Methodology - [17] (Self-Attention)

- Captures dependencies and relationships within input sequences.
- Used in natural language processing and computer vision.
- Model evaluates the importance of different input components.
- Utilizes query (q), key (k), and value (v) vectors from the same input.

# Methodology - [18] (Multi-Head Attention)

- Contains multiple attention head with different learned linear projections.
- Contains three vectors queries, keys and values derived from input embedding through linear transformation.
- The attention is given by:

$$Attention(Q,K,V) = softmax\left(\frac{Q.K^T}{\sqrt{d_k}}\right).V$$

Where $d_k$ is the dimension of keys and queries

# Methodology - [19] (Positional Embeddings)

- The model is actually uninformed of the token's spatial relationship.
- Used to add spatial information to the image data.
- Usually, involves assigning tokens weights derived from two high-frequency sine waves.

# Methodology - [20] (Compiler)

- A computer program that translates computer code written in one programming language into another language.
- Translate the Domain Specific language into a HTML/CSS code.

# Methodology - [21] (Customization)

- Transforming the sketch to HTML code is not enough.
- Component must be styled accordingly for good look.
- Some customizing by theme selection, color selection, font-selection, style selection etc.
- Contains randomly generated text can be edited.
- Can add image from link or file.

# Results - [1]
# (Model Training)

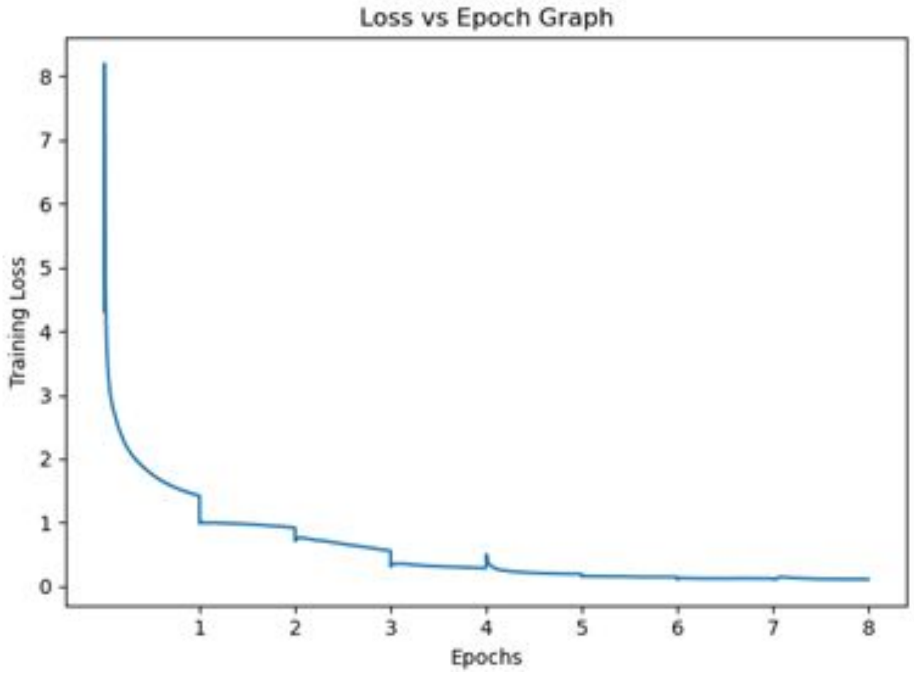



**Figure: Epoch vs Loss graph for Training Data**

# Results - [2] (Model Training)

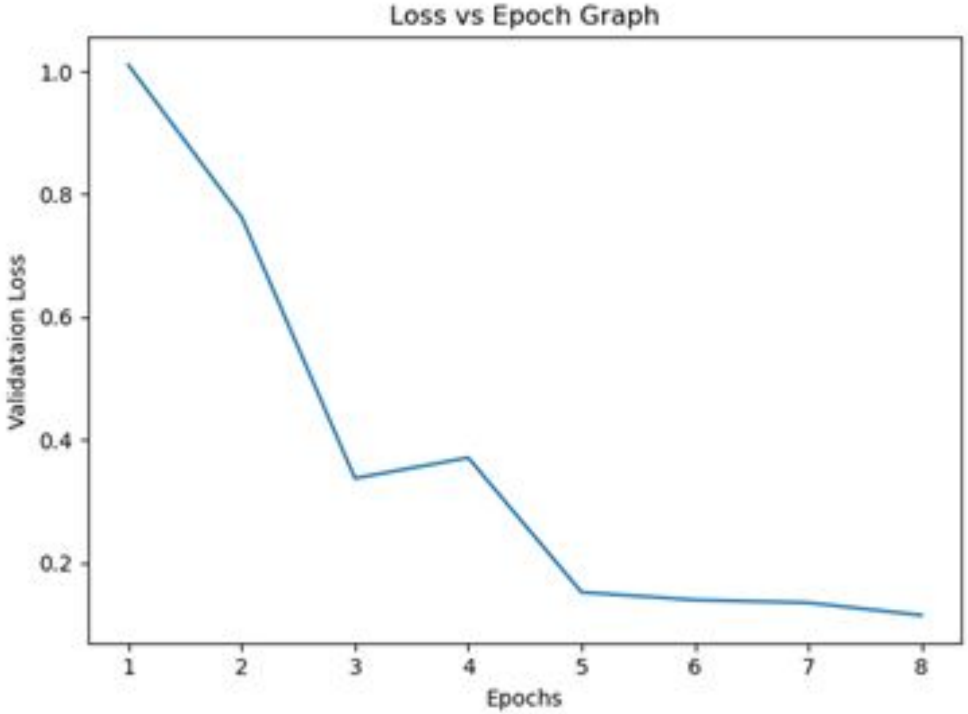



**Figure: Epoch vs Loss graph for Validation Data**

# Results - [3] (Model Evaluation: BLEU)

The average BLEU score of n-grams 1, 2, 5 and 10 for testing data are as follows:

- 1-grams BLEU Score: 0.9544114683715649
- 2-grams BLEU Score: 0.9378854888760932
- 5-grams BLEU Score: 0.8867411989388778
- 10-grams BLEU Score: 0.7982317463855622

# Results - [4]
# (Model Evaluation: BLEU)

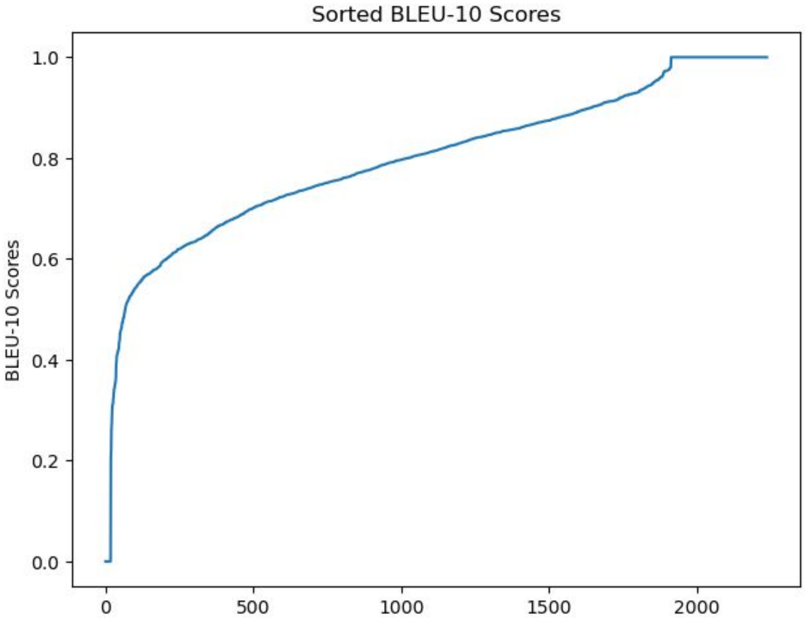



**Figure:BLEU-10 Scores in sorted order**

# Results - [5]
# (Model Evaluation: ROUGE)

**Table: ROUGE Scores for test data**

| Metric | Precision | Recall | F1-Score |
|---|---|---|---|
| ROUGE-1 | 0.9481 | 0.9846 | 0.9652 |
| ROUGE-5 | 0.7806 | 0.8112 | 0.7948 |
| ROUGE-L | 0.9428 | 0.9791 | 0.9598 |

# Results - [6] (Model Evaluation: ROUGE)

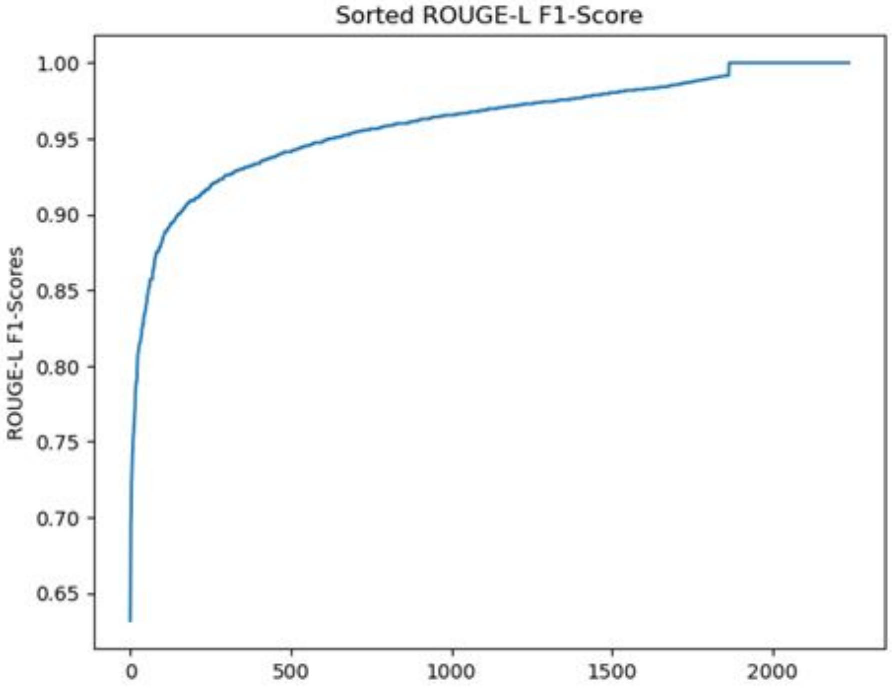



**Figure:ROUGE-L F1-Scores in sorted order**

# Results - [7]
# (Model Evaluation for Hand Drawn Sketch)

The dsl code for 100 hand drawn images were generated using the model. The BLEU Score is obtained as:

- 1-grams BLEU Score: 0.6543238343196423
- 2-grams BLEU Score: 0.5798785348251052
- 5-grams BLEU Score: 0.36159844471049923
- 10-grams BLEU Score: 0.10595463254487267

# Results - [8]
# (Model Evaluation for Hand Drawn Sketch)

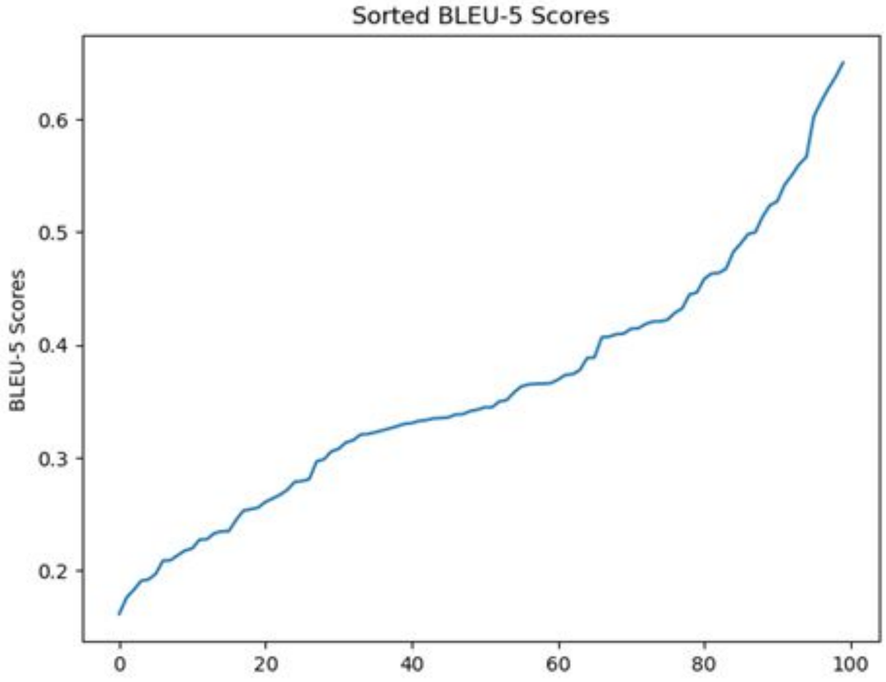
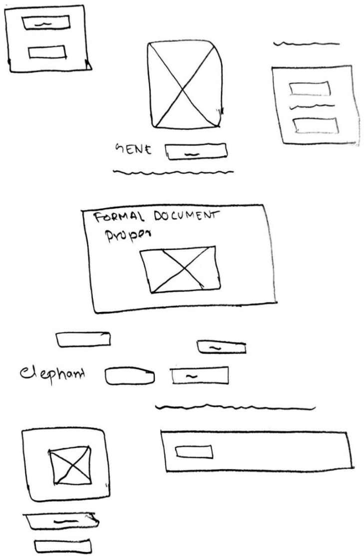
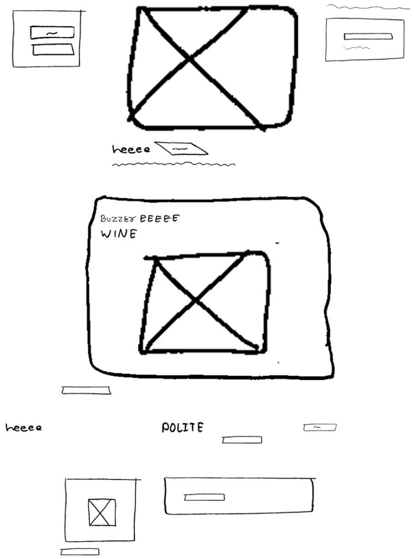



**Figure:BLEU-5 Scores in sorted order**

# Results - [9]
# (Model Evaluation for Hand Drawn Sketch)




Input

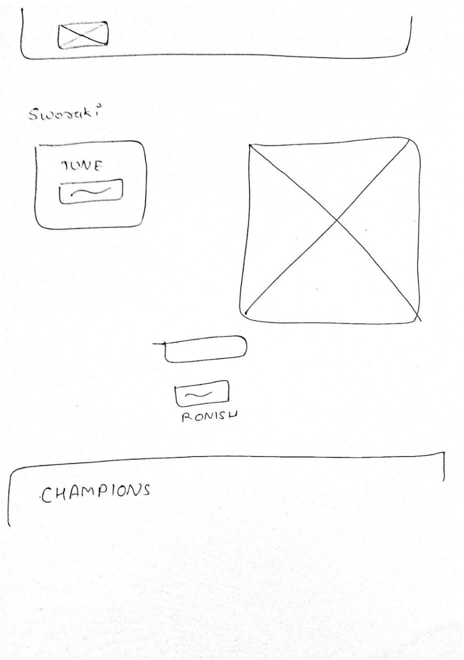


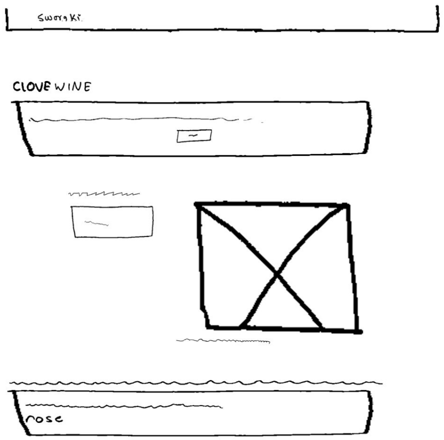

Predicted



**ROUGE-5 Score: 0.6068**

# Results - [10]
# (Model Evaluation for Hand Drawn Sketch)




**Input** **Predicted**

**ROUGE-5 Score: 0.1527**

# Discussion of Results

- Trained using 8000 images generated by dataset generator for 8 epochs.
- 2000 images generated images were evaluated by trained model.
- The BLEU and ROUGE score obtained showed good performance of model.
- But, for hand drawn data the scores are lower than for generated data.

# Remaining Tasks

- Optimizing Dataset generation process to match real world data.
- Collect more Hand-drawn data.
- Fine-tuning the model for better performance.
- Improving the architecture of model.
- Improve the preprocessing steps for hand-drawn sketches.
- Creation of intuitive User interface for customization and styling of generated code.

# References-[1]


1. Tony Beltramelli,"Pix2code: Generating Code from a Graphical User Interface Screenshot," 2017. [Online]. Available: https://www.researchgate.net/publication/325920827_pix2code_Generating_Code_from_a_Graphical_User_Interface_Screenshot. [Accessed: June 2024].
2. Sarah Suleri, Vinoth Pandian,Svetlana Shiskovets,Matthias Jarke,"Eve: A sketch-based Software Prototyping Workbench," 2019. Available: https://www.researchgate.net/publication/332777261_Eve_A_Sketch-based_Software_Prototyping_Workbench. [Accessed: June 2024].
3. Biniam Behailu Adefris, Ayalew Belay Habtie, Yesuneh Getachew Taye, "Automatic Code Generation from Low Fidelity Graphical User Interface Sketches Using Deep Learning," 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9971204. [Accessed: June 2024].
4. Daniel Baulé, Christiane Gresse von Wangenheim, Aldo von Wangenheim, Jean C. R. Hauck and Edson C. Vargas Júnior, "Automatic Code Generation from Sketches of Mobile Applications in End-User Development Using Deep Learning," [Online]. Available: https://www.researchgate.net/publication/349963791_Automatic_code_generation_from_sketches_of_mobile_applications_in_enduser_development_using_Deep_Learning . [Accessed: June 2024].

# References-[2]


5. Jia Li, Yongmin Li, Ge Li, Zhi Jin, Yiyang Hao, and Xing Hu, "STC (Sketch To Code) - An Enhanced HTML & CSS Autocode Generator from Handwritten Text and Image Using Deep Learning," IEEE, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10537336. [Accessed: June 2024].
6. A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," [Online]. Available: https://arxiv.org/abs/2010.11929. [Accessed: June 2024].
7. "Canny edge detector," Wikipedia, The Free Encyclopedia. [Online]. Available: https://en.wikipedia.org/wiki/Canny_edge_detector. [Accessed: 17-Jun-2024].
8. Q. Xin, Y. Zhang and B. Tan, "Image Captioning with Vision/Text Transformers," [Online]. Available: https://qi-xin.github.io/image%20caption%20generation.pdf. [Accessed: June 2024].