MAJOR PROJECT
8TH SEMESTER

# Image to CODE

AUTOMATIC HTML CODE GENERATION FROM IMAGES USING DEEP LEARNING BASED MODEL

UNDER THE GUIDANCE OF:
DR. ABHISHEK VAISH

## Contents

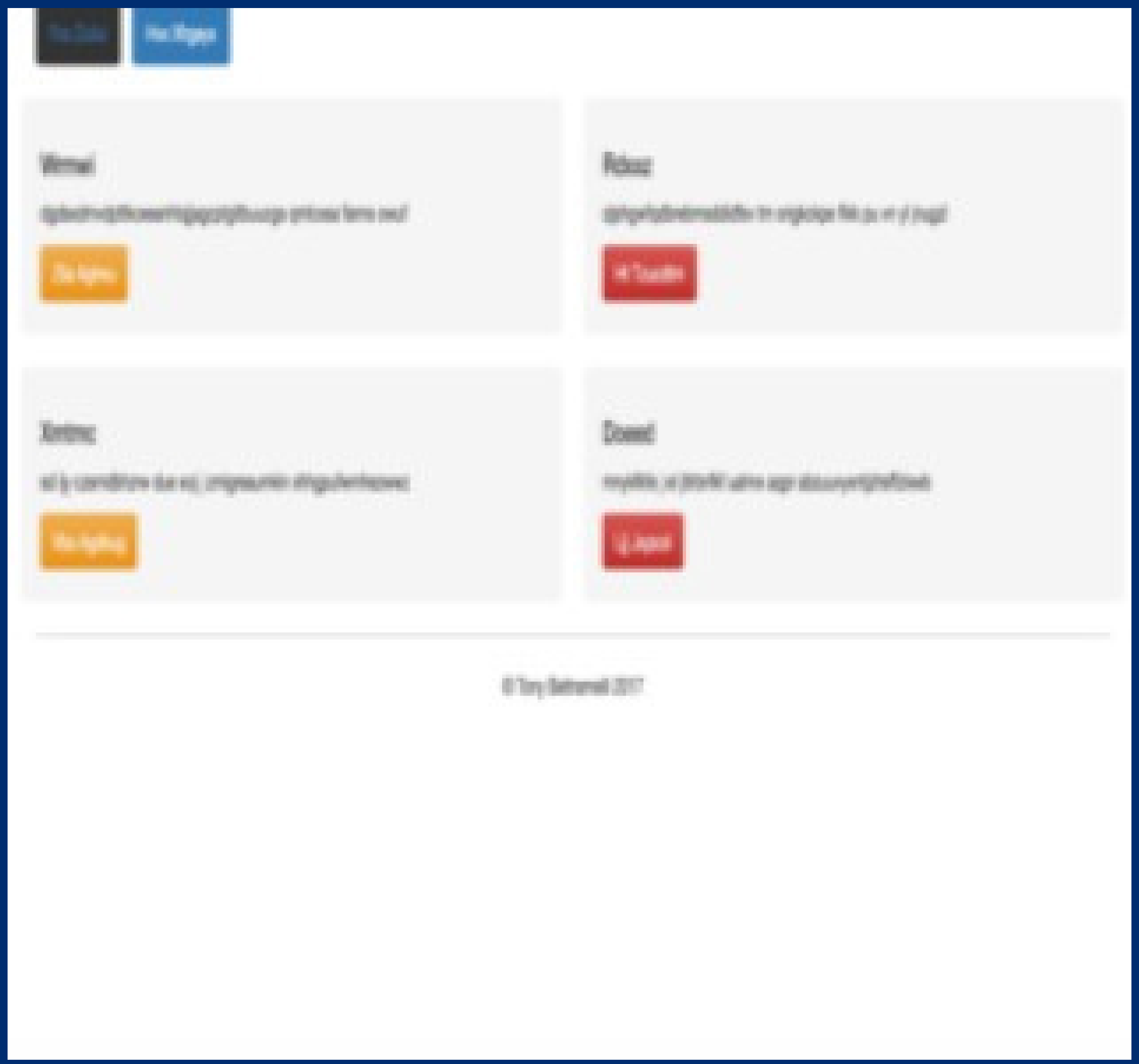
## Quick Recap

- The main aim is to automate the front-end generation process for websites using deep learning techniques.

- We will use a CNN and RNN encoder-decoder model to address challenges such as recognizing dependencies and creating sequential text.

- The objective is to automate front-end development in a secure manner, reducing the cost of hiring frontend engineers.

# In short



INPUT

```
header{
btn-inactive,btn-active
}
row{
double{
small-title,text,btn-green
}
double{
small-title,text,btn-green
}
}
row{
double{
small-title,text,btn-green
}
double{
small-title,text,btn-green
}
}
```

OUTPUT

## Feedback from C2

At the end of discussion, sir advised us to look into automating the front-end generation for **Web3** based models and figure out how we can extend our project to do so.

We found some research papers, blogs and articles on the same and studied them thoroughly.

The outcomes will be discussed in the following slides.

## Automating front-end generation for Web3

Web3 technology allows for the integration of decentralized features, such as interacting with smart contracts and accessing blockchain data, into web applications.

To automate the front-end generation process in Web3, we will need to:

- Implement code generation for more advanced front-end libraries like CSS, JS.
- Configure the model to take more than just the UI image as input.
- The input files will include smart contract details, specifications about the blockchain network to be used, event handlers etc.
- Implement highly advanced Web3 based front-end libraries like Web3.js, Ethers.js

## Automating front-end generation for Web3

In addition to the technologies used in our model, we might need to add additional frameworks to handle more complex problems like Web3. Some of the machine learning models we can incorporate are:

- **NLP** to analyze and process natural language input from users, such as requirements or descriptions of the Web3 Features.
- Transformer models, such as the popular **BERT** can be employed for tasks like code completion or code suggestion.
- **Generative Adversarial Networks**: GANs can be utilized to generate realistic and visually appealing UI designs for new pages.

## Challenges in the process

When automating code generation for Web3-based frontend projects, you may encounter several challenges that require careful consideration and mitigation:

- Complexity of Web3 Frameworks
- Flexibility and Customization
- Evolving Standards and Technologies
- Security Considerations
- Handling Design Complexity

## Dataset: Pix2code

**Dataset Link:** https://www.kaggle.com/code/himasha0421/pix2code/input

The Pix2code dataset is a collection of image-HTML pairs that are used for training and evaluating ML models that aim to automatically generate code from GUI designs.

The dataset contains **15,000** images of GUI designs, each of which is paired with the corresponding HTML code that describes the layout and functionality of the design.

The images are taken from a wide variety of sources.

The dataset contained three portions: Android, IOS and web, we will currently be working on the web segment.

The dataset is split into three parts: a training set (**6000**) , a validation set (**500**), and a test set (**500**).

## Methodology

01 Data  Collection

02 Pre-processing the Data

03 Building the CNN Model

04 Building the RNN Model

05 Training the model

06 Evaluating the model

07 Fine-tuning the model

08 Deploying the model

01

## Data  Collection

Collect a dataset of UI design images and their corresponding HTML code.

02

## Pre-processing the Data

Pre-process the images and HTML code to make them suitable for training.

03

## Building the CNN Model

CNN model to extract features from the **input image**. The output will be a **feature vector** that will be input to RNN Model.

04

## Building the RNN Model

RNN model that generates **HTML Code** for the **input feature vector**s.

05

## Training the model

Train the CNN-RNN model after pre processing the dataset. Pre processing involves cleaning the HTML code using **one-hot-encoding.**

06

## Evaluating the model

Evaluate the model's performance on a separate validation dataset .

07

## Fine-tuning the model

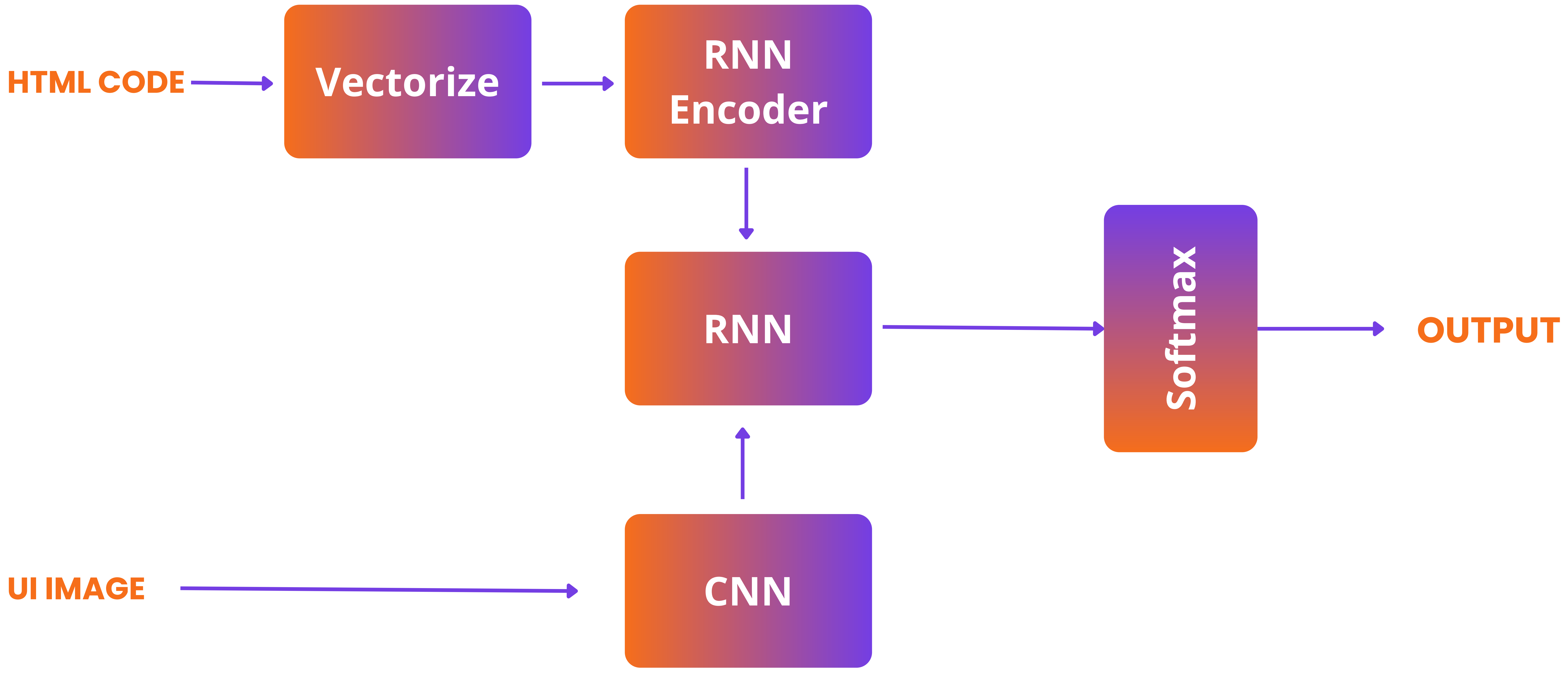
You may need to fine-tune the model by adjusting the hyperparameters or adding more layers to the CNN or RNN model.

08

## Deploying the model

The model can be deployed to a production environment where it can be used to convert UI designs to HTML code. We can also create a web application or API that facilitates the same.

# Flow of the model



# Performance Metrics

**Cross-entropy loss**, also known as log loss, is a widely used loss function in machine learning and deep learning, particularly in classification tasks.

The formula for cross-entropy loss is:

$$L(y, \hat{y}) = -\Sigma\ y\_i \log(\hat{y}\_i)$$

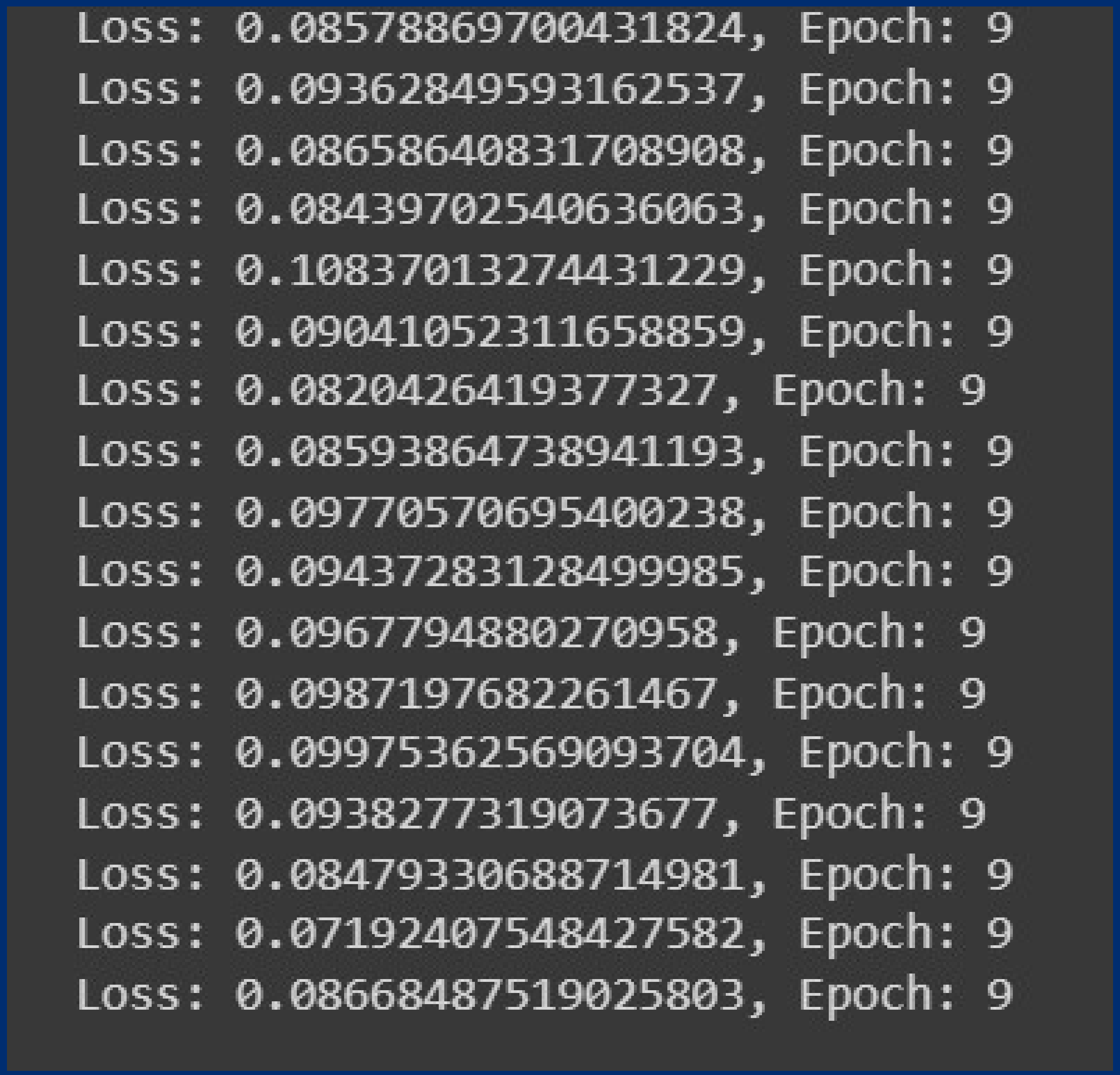
where:

- L: the cross-entropy loss
- y: the ground truth label (in this case, the ground truth HTML code)
- $\hat{y}$: the predicted label (in this case, the predicted HTML code)
- y_i: the i-th element of the ground truth label (either 0 or 1, indicating the absence or presence of a specific HTML element or attribute)

$\hat{y}$_i: the i-th element of the predicted label (a value between 0 and 1, indicating the model's confidence that the corresponding HTML element or attribute is present in the predicted code)

## Results

After complete epochs, our Cross-entropy loss is **0.086**



```
Loss: 0.08578869700431824, Epoch: 9
Loss: 0.09362849593162537, Epoch: 9
Loss: 0.08658640831708908, Epoch: 9
Loss: 0.08439702540636063, Epoch: 9
Loss: 0.10837013274431229, Epoch: 9
Loss: 0.09041052311658859, Epoch: 9
Loss: 0.0820426419377327, Epoch: 9
Loss: 0.08593864738941193, Epoch: 9
Loss: 0.09770570695400238, Epoch: 9
Loss: 0.09437283128499985, Epoch: 9
Loss: 0.0967794880270958, Epoch: 9
Loss: 0.0987197682261467, Epoch: 9
Loss: 0.09975362569093704, Epoch: 9
Loss: 0.0938277319073677, Epoch: 9
Loss: 0.08479330688714981, Epoch: 9
Loss: 0.07192407548427582, Epoch: 9
Loss: 0.08668487519025803, Epoch: 9
```

# Future Scope

**01**

We can extend this project and enable it to generate new sections or pages of a website based on the previously fed input. without giving any new UI input.

**02**

We can also apply the dimensionality analysis in this project, i.e. predicting the size of the object in the given image.

**03**

Various security features along with the code generation can be added in order to make the code generation secure

**04**

We can research on automating the backend generation and club it together to make a complete automatic website generation platform.

**05**

We can extend the project further to generate code for more advanced front-end libraries like CSS, JS to implement front-end which contains interactive web pages.

**06**

We can also modify and extend the model further to include Web3 functionalities and include code generation for libraries like Web3.js and Ethers.js

# Timeline

## C1

Researching on the topic and document review

Learning the related concepts

## C2

Dataset collection and preprocessing

Building the CNN Model

Building the RNN Model

## C3

Final compilation of model

Final Report

## References

- B. Aşıroğlu et al., "Automatic HTML Code Generation from Mock-Up Images Using Machine Learning Techniques," 2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT), Istanbul, Turkey, 2019, pp. 1-4, doi: 10.1109/EBBT.2019.8741736.
- D. Yashaswini, Sneha and N. Kumar, "HTML Code Generation from Website Images and Sketches using Deep Learning-Based Encoder-Decoder Model," 2022 IEEE 4th International Conference on Cybernetics, Cognition and Machine Learning Applications (ICCCMLA), Goa, India, 2022, pp. 133-138, doi: 10.1109/ICCCMLA56841.2022.9989298.
- B. Aşiroğlu et al., "A Deep Learning Based Object Detection System for User Interface Code Generation," 2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey, 2022, pp. 1-5, doi: 10.1109/HORA55278.2022.9799941.
- Generating Smart Contract Front-ends Using Machine Learning Techniques"by X. Li, X. Ma, and C. Zhang
- Automated Smart Contract Front-end Generation by S. Schuster, S. Schneckenburger, and K. Meißner

# Thank You

| | |
|---|---|
| SHREYA SHROTRIYA | IIT2020501 |
| HIMANSHU BHAWNANI | IIB2020035 |
| MANTRARAJ GOTECHA | IIT2020269 |
| SHREYANS JAIN | IIT2020238 |
| ISHAAN KAUSTAV | IIT2020261 |