Design Document

OVERVIEW:

This software/website has two features. The first feature will help you analyse the text and tell you how toxic, insulting, and threatening the text is. The second feature will allow you to upload an image and tell you how many faces are in that image. This text analysis feature can help mitigate toxicity and ensure healthy dialogue. Both features will be implemented using two different public APIs. The first feature will use an API called "Perspective" and the second feature will use an API called "Imagga".

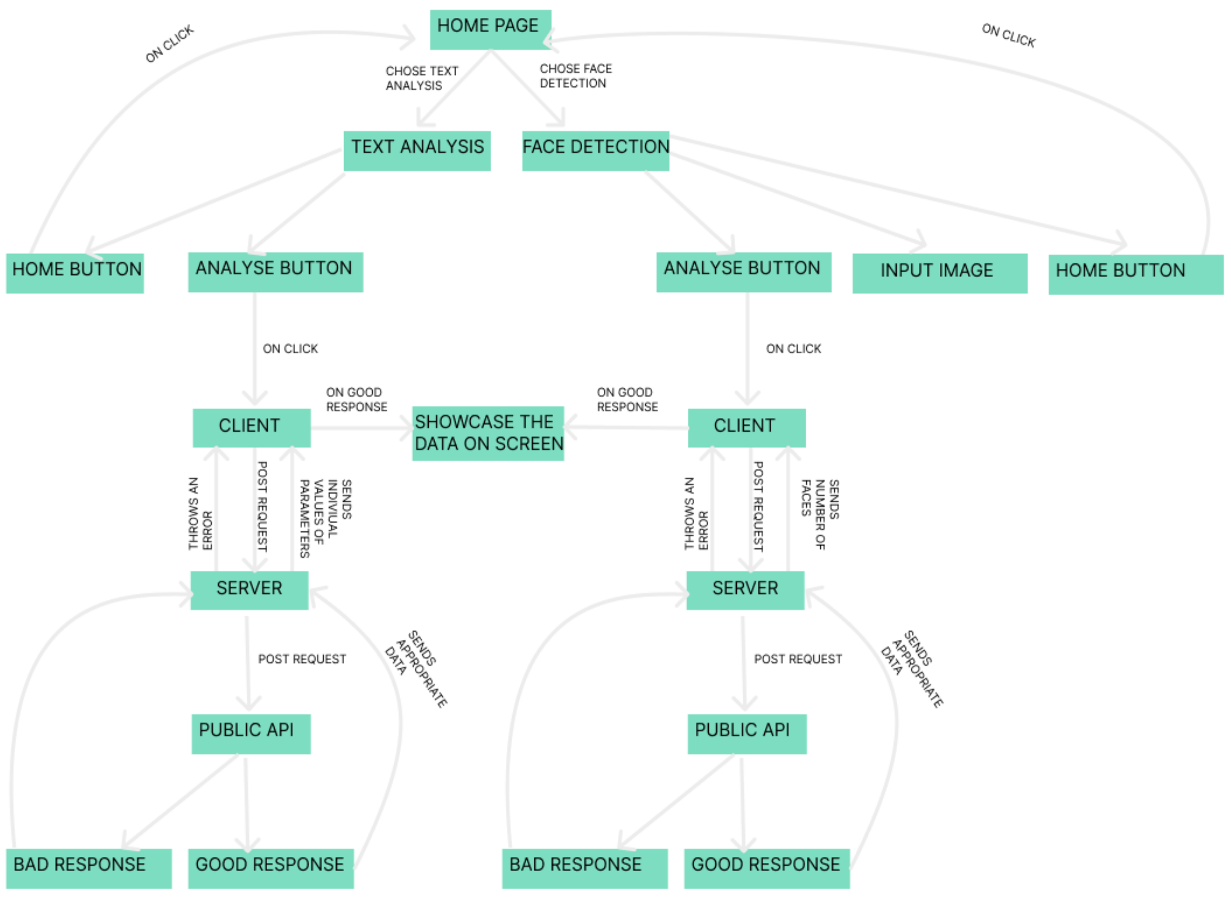
PROBLEM STATEMENT:

- Online hatred is commonplace today. The text analysis feature is meant to assist persons who speak English as a second language but are not very good. The purpose of this software is to assist such individuals in determining if the person speaking to them is speaking in good faith or is being toxic or disrespectful.
- The task of counting everyone in attendance at a conference or event is quite time-consuming. When determining how many people attended an event or gathering, the Face identification tool will be quite helpful.

HOW WILL IT WORK?

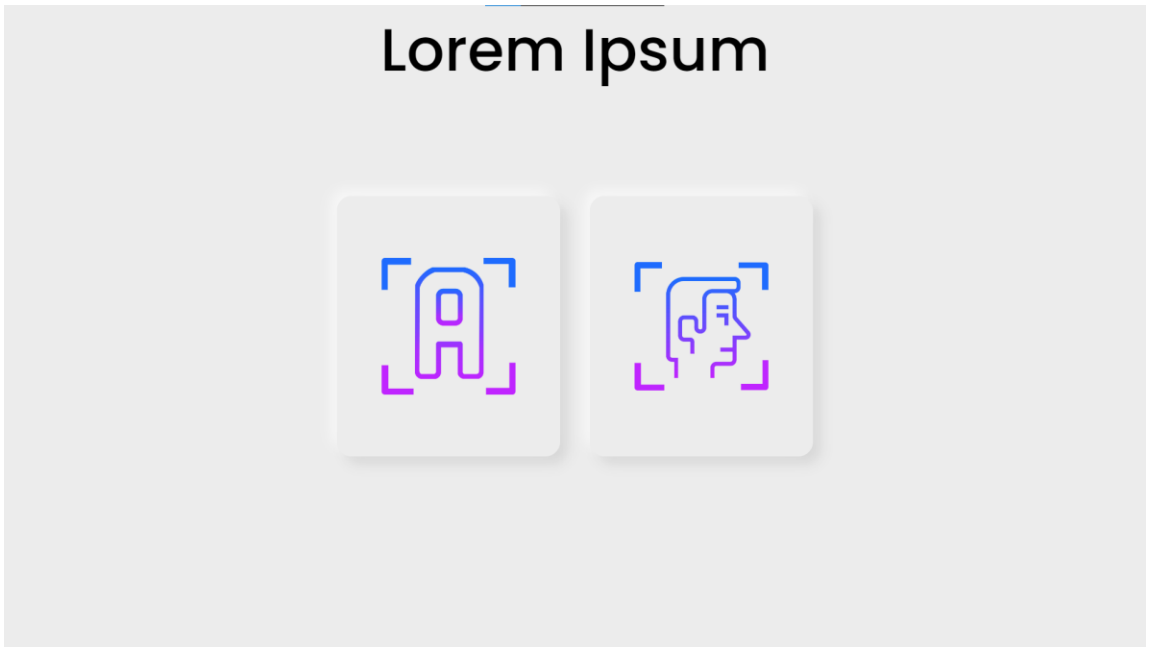
This software will show the outputs on frontend (React.js) and will call to the API of the backend (Node.js) which will act as a middleware where the actual API call to the public API will take place.

DESIGN AND WORKFLOWS

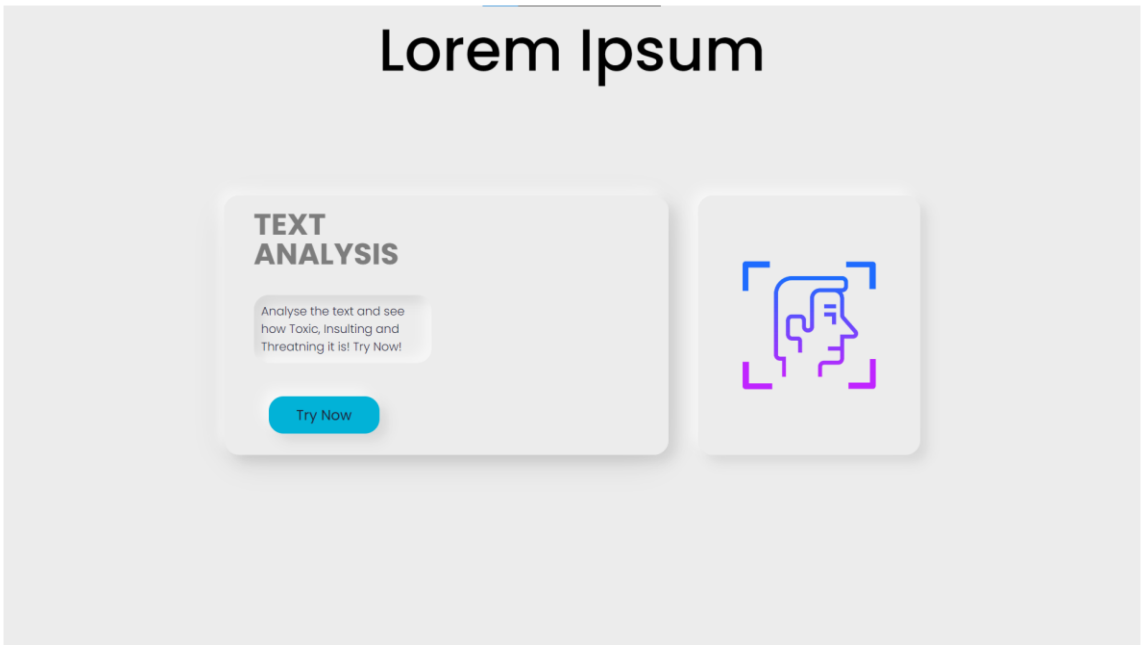


Frontend: UI

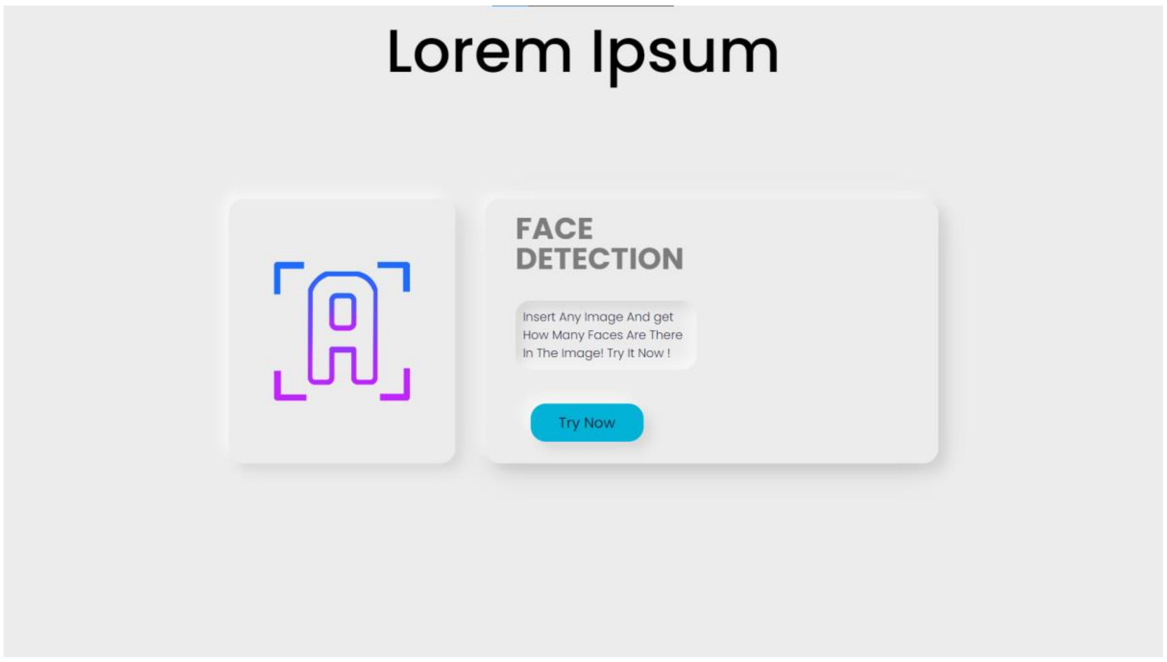
- First, the user will be presented with two options—text analysis or face detection—and a brief description of each will be supplied so that he or she may decide which function they wish to employ.



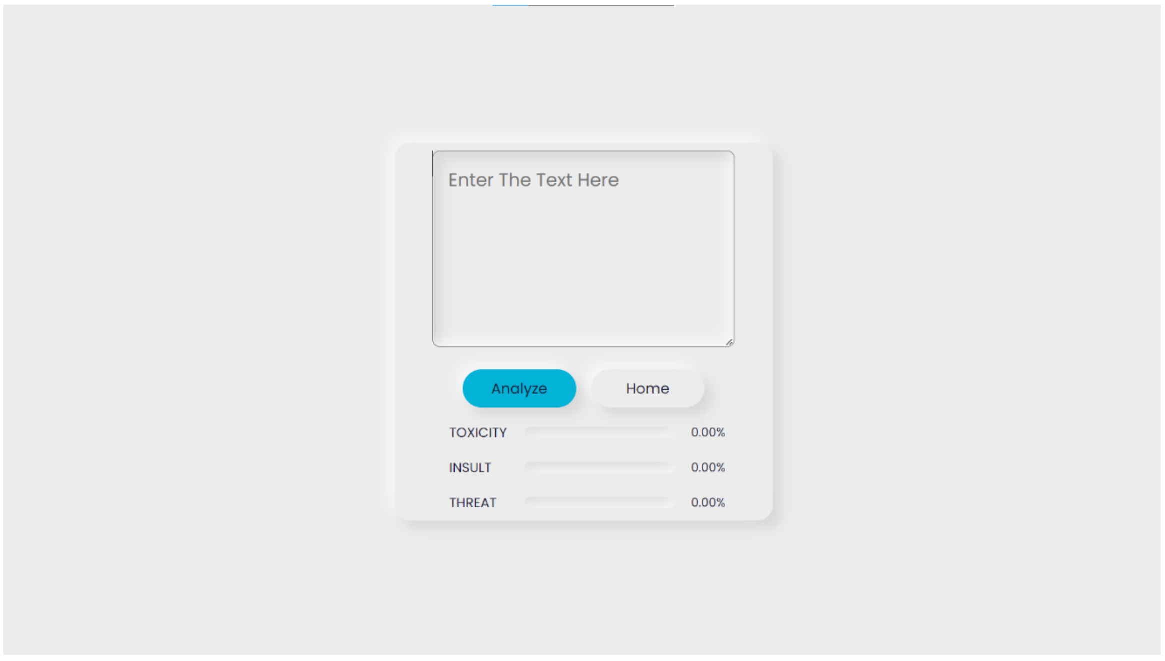
- When one of the features is clicked, the entire page switches to the user interface for that particular API functionality.

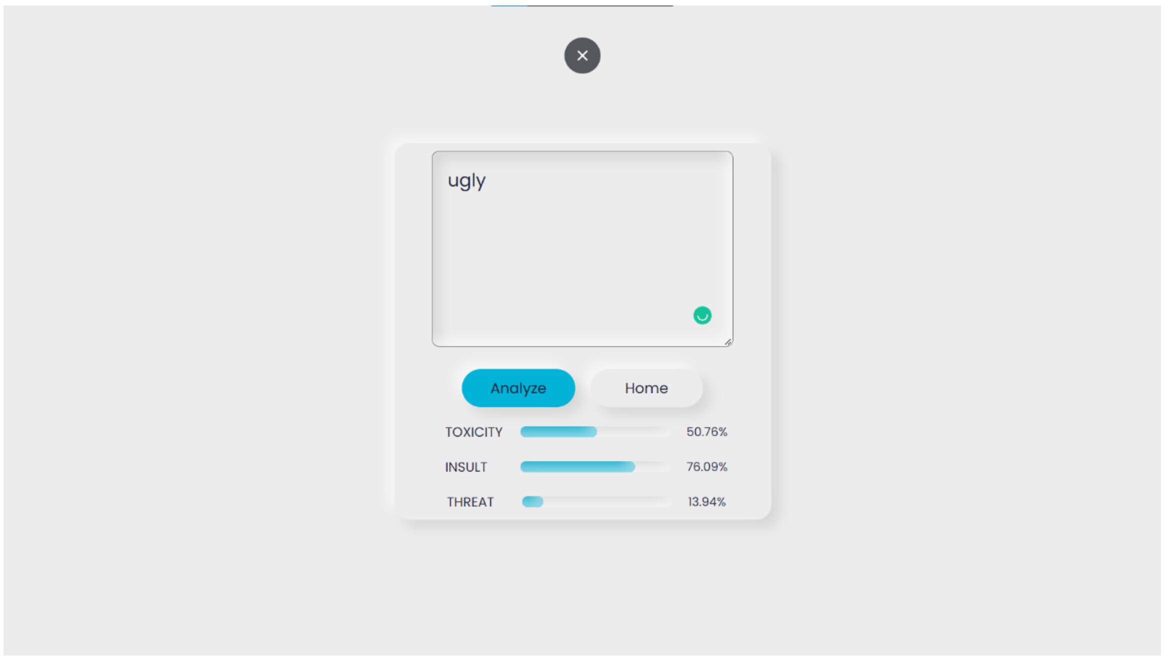


Analyse the text and see how Toxic, Insulting and Threatning it is! Try Now!


Lorem Ipsum
FACE DETECTION
Insert Any Image And get How Many Faces Are There In The Image! Try It Now !
Try Now

- If client select text analysis, the page will change to a card with three bars showing the Toxicity, Insult, and Treat metre based on the API answer and client input, as "Textarea" and Two buttons "Analyse" and "Home." You are returned to the original page after hitting the "Home" button. The "Analyse" button will initiate an API request to the server, and once the server responds, the bars will display the text's percentage-based parameters like (Toxicity, Insults and Threat).

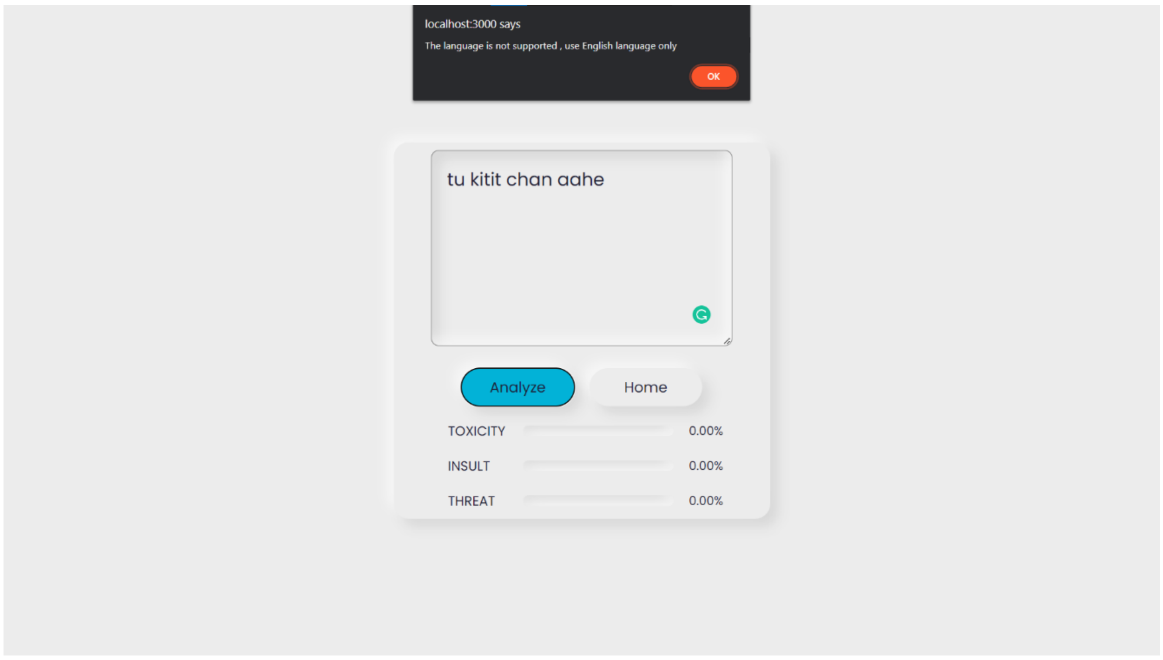




ugly

TOXICITY		50.76%
INSULT		76.09%
THREAT		13.94%

- If the user enters an unsupported language then a error message will pop on the client screen suggesting to use English language only.



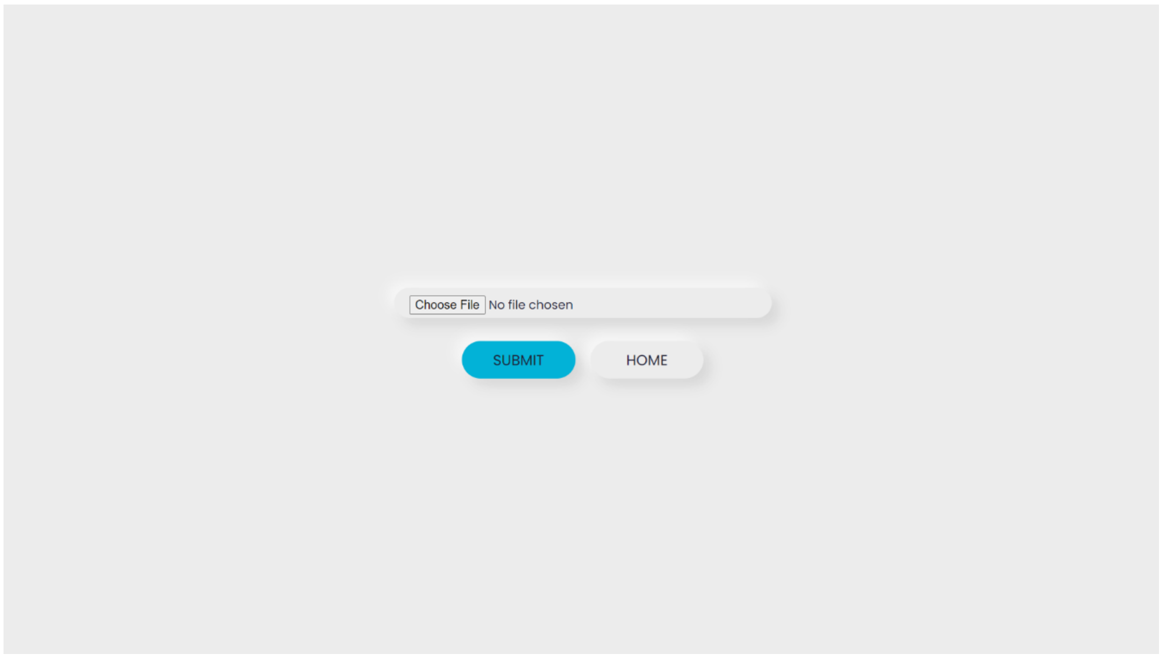
localhost:3000 says

The language is not supported , use English language only

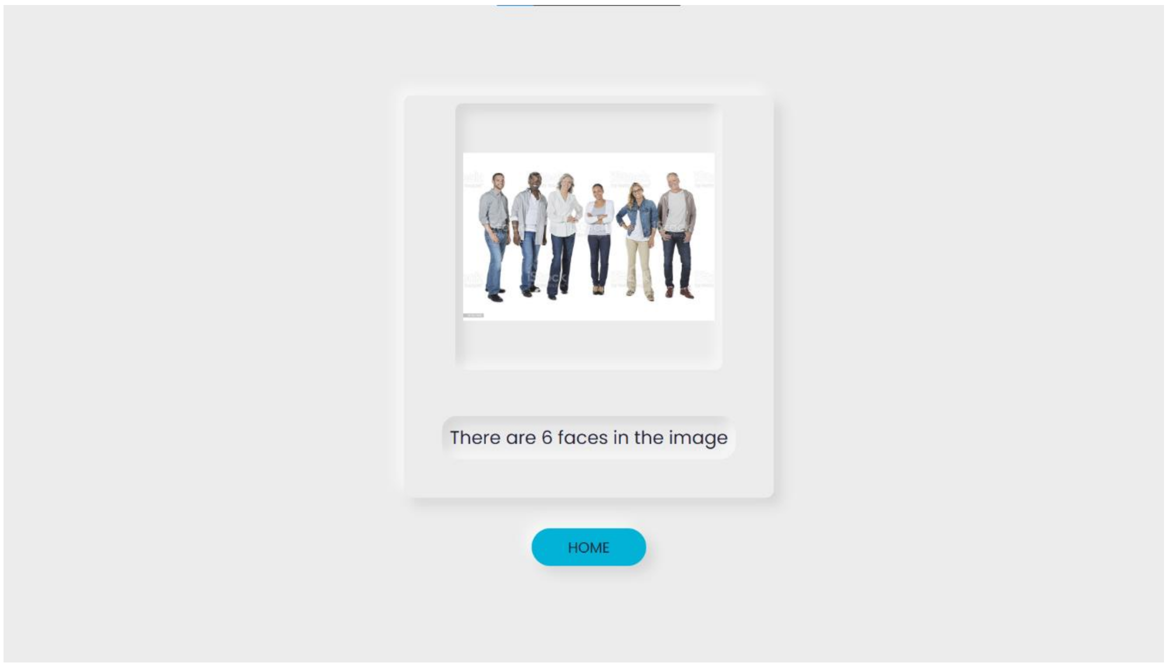
tu kitit chan aahe

TOXICITY		0.00%
INSULT		0.00%
THREAT		0.00%

- If the client selects the face detection, the page will change and it will show a button to upload the file from the client’s local machine. This page will also have two more buttons: "Submit", when clicked, sends the image to the server by making the proper API call; and "Home," when clicked, will take the client to the home page.



- After getting a proper response from the server, the page will again change and showcase the input image with how many faces are there in that image, along with a "Home" button to take the client to the home page.


There are 6 faces in the image

Backend + Frontend (API calls)

- When the text input is given to the frontend, the frontend will make a POST request to the server to a corresponding API endpoint as “…/text_analysis”. The frontend will send the JSON object to the server in the following format

```
{
      statement: `${sentence}`,
}
```

If the format is wrong or the text is empty or unsupported language is given the server will throw an error and send a JSON object as follows

```
{
    error: "error",
}
```

If the format of the JSON object is correct then the server will make a POST request to the "Perspective" public API sending the client's input text in form of JSON object as follows

```
{
    text: `${statement}`, // Client's Input text
 },
 requestedAttributes: { // requesting for certain parameters
      TOXICITY: {},
      INSULT: {},
      THREAT: {},
 },
```

After receiving a proper response from the Perspective API, the server will send the response to the frontend with following JSON object

```
{
        TOXICITY: `${tox}`, // tox ~ 0-1
        INSULT: `${ins}`, // ins ~ 0-1
        THREAT: `${trt}`, // trt ~ 0-1
        error: "",
 }
```

Here the "tox", "ins" and "trt" are the variable which will store the corresponding probability values of toxicity, insult and threat which will be in the response from the public API. error:"" shows that the input from the frontend is valid.

- When client clicks on the button to upload an image the image will be taken input in base64 format. Then the frontend will to POST API call to the server to an corresponding API endpoint "…/img_analysis". The frontend will send the JSON object in the following format

```
{
  img_bs64: `${up_img}`, //base64 image data
}
```

Here the "up_img" is variable which will have Base64 data of the input image.

Then the server will make the POST request to the "imagga" public API endpoint in the format of form-data. After getting a appropriate response from the public API the server will send the response to the frontend as JSON object as follows

```
{
      Faces: data.result.faces.length, // Number of Faces
}
```

After getting the response from the server the frontend will showcase the image and the number of faces on the screen. The image is not being saved anywhere locally due to privacy reasons.

<u>WHY REACT.JS AND NODE.JS?</u>

- React.js and Node.js are both JavaScript based so decided to use them to save time of development.
- Node.js is fast because of the non-blocking I/O model, it also has a fast pass through.

- As the public API is used, the Node.js is light weight as compared to the full setup of java and all
- React.js is very convenient when we have to show conditional things on web page without creating separate HTML pages for each one, useState hook in React.js comes very handy as it re-renders the whole page on conditions.
- Express.js is used to reduce the coding time by half and help build the efficient applications.
- Express.js is a layer built on the top of the Node.js that helps manage servers and routes.

TIMESTAMP:

- Start Date: **26 October 2022**
- **26 October 2022**: Found public API for the project and made the full backend in Node.js with all the API calls.
- **27 October 2022**: Started working on Frontend in React.js and made a rough design.
- **28 October 2022**: Implemented the Text Recognition functionality in to the software.
- **29 October 2022**: Implemented the Face Detection Functionality into the software where user can add image link to detect face. Added code in frontend and backend so when unsupported language is input it show the error on screen.
- **30 October 2022**: Fully changed the code for input of an image in Face Detection functionality and used file reader. Now client can directly upload local images for Face Detection.
- **31 October 2022**: Started working on aesthetics of the software. Used black themed Neumorphism design and added some Images as thumbnail for Face Detection and Text Recognition on Home Page.
- **1 November 2022**: discarded the Black themed Neumorphism design and changed it to White Neumorphism design and also changed the thumbnail images to match with the look. Also changed the backend code to take care of code 400 response error.

FURTURE WORK:

- Option to request more parameters like “IDENTITY_ATTACK”, “SEVERE_TOXICITY”, “PROFANITY” etc.
- Option to show the probability of how accurate the face analysis is!
- Can use more API endpoints from the public API’s to showcase more things!

AUTHOR:

Prathamesh Jondhale

102001012

Civil Engineering

Indian Institute of Technology Palakkad