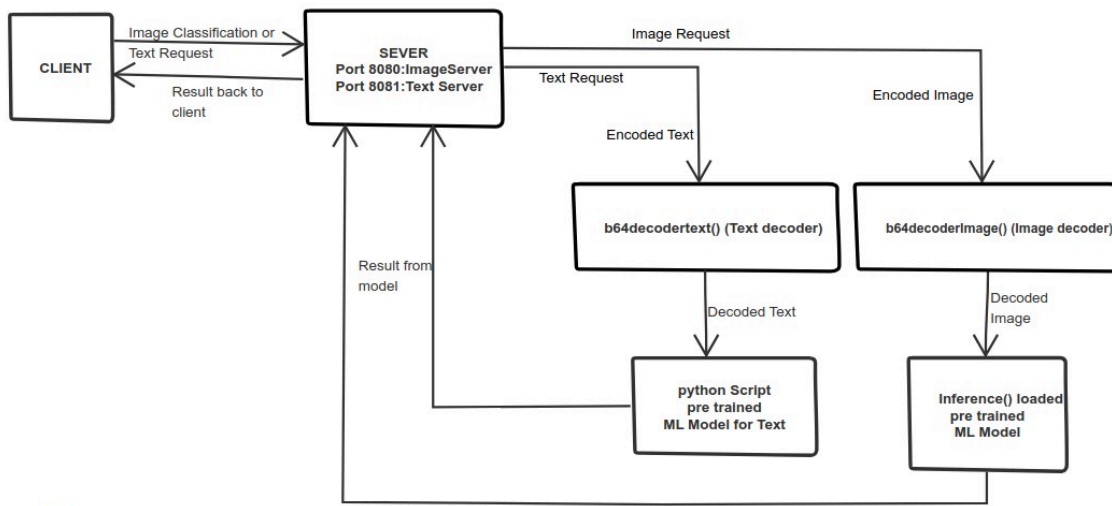


Project Overview:

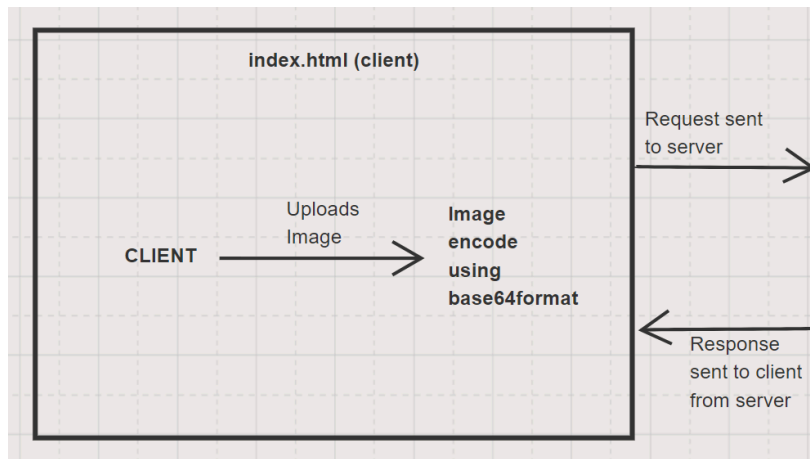
The project offers a toolkit featuring multiple pre-trained machine learning models, each trained for specific tasks. Users can perform tasks such as classification and text prediction easily with this toolkit.

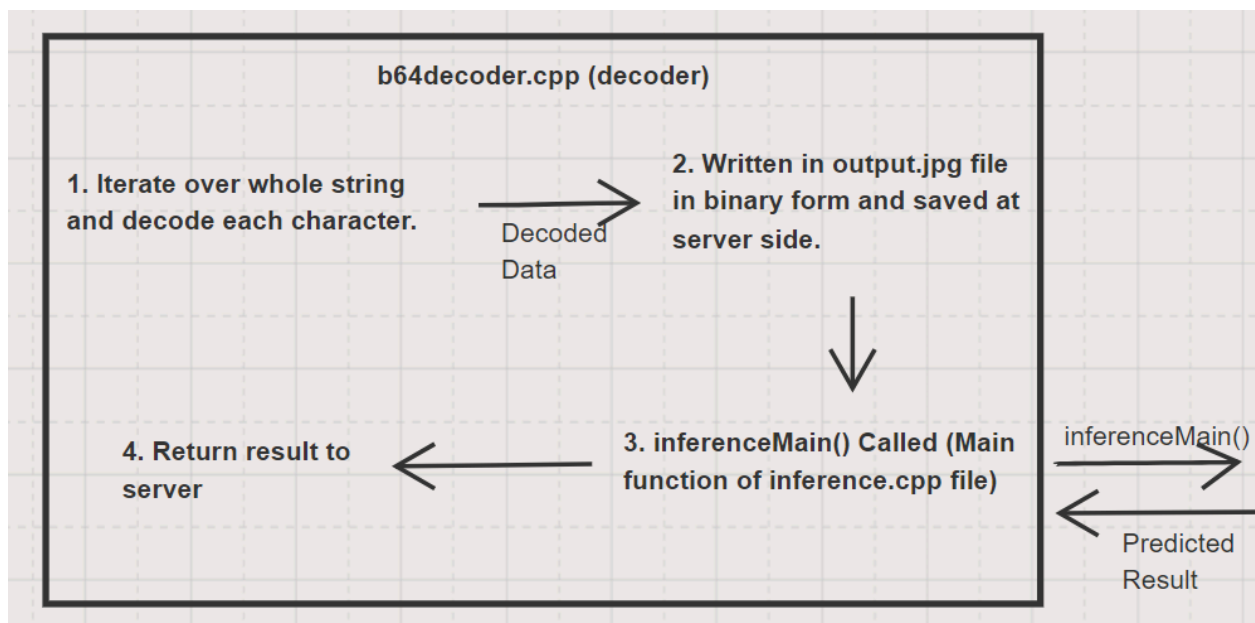
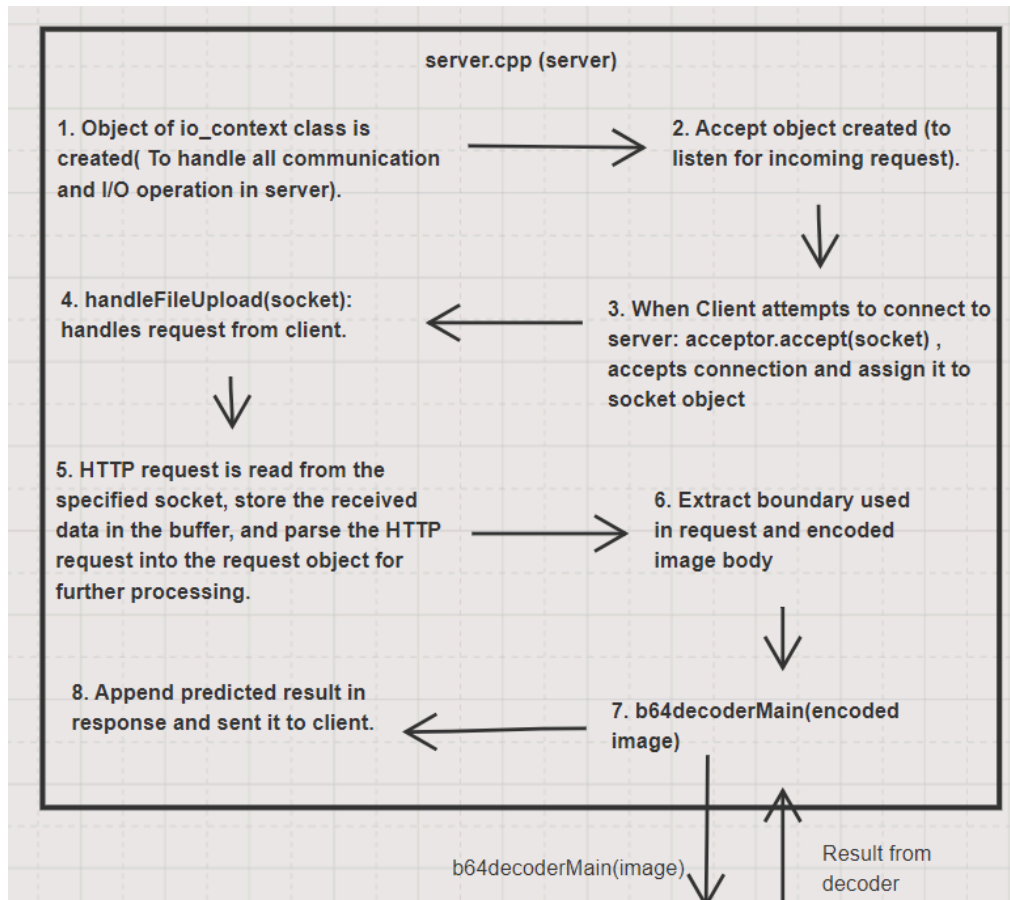
Design:

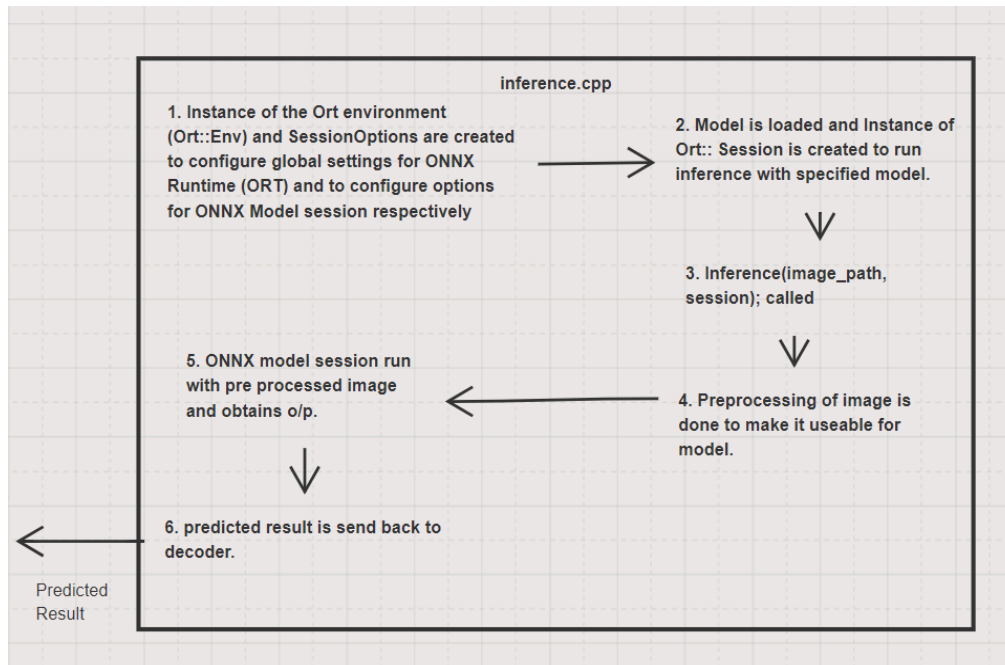
(breakup into processes/threads/classes/functions)



Modular Functionality:







For Image Classification

1. Client sends request to server through a HTML form.
Input: Image
Request is **encoded to base64 format** before sending to the server. Then send it to the server.
2. Server stores request and performs parsing of request to extract request body. In case of Image Classification, the extracted request body is sent to the decoder file, to decode the image encoded using base64.
Server is already listening to a port and the request is read and stored in a buffer. Next request body is extracted and sent to the decoder using **b64decoderMain()**.
Server is built using the **boost** library.
3. Image is decoded using **base64_decode()**. and the result is written in 'output.jpg' file in binary format. After decoding image **inferenceMain()** is called for further processing.
4. Inference module **openCV** is used to perform image processing on decoded images and **ONNX Runtime** is used to perform inferencing. This loads a pre-trained model and uses decoded images to perform prediction.

External Dependancies:

1. ONNXRuntime (version)
2. OpenCV
3. Boost

For Text Model

1. Data Collection:

Input text data is collected through an HTTP form, where users input text for sentiment analysis.

2. Data Encoding:

The collected text data is encoded into Base64 format before being transmitted to the server.

3. Server Processing:

Upon receiving the encoded text data, the server parses the request and decodes the text using a corresponding decoder, `b64decodertext.cpp`.

4. Text Preprocessing:

The decoded text undergoes preprocessing to normalize and clean it for further analysis.

Non-alphanumeric characters are removed, and the text is converted to lowercase.

5. Embedding Generation:

The preprocessed text is sent to OpenAI's API for generating text embeddings. OpenAI's text embedding model is utilized for generating embeddings from the input text.

6. ONNX Model Loading:

The system loads a pre-trained ONNX model (`text_model.onnx`) using ONNX Runtime.

This model is responsible for performing sentiment analysis on the text embeddings.

7. Inference Execution:

The text embeddings are fed into the loaded ONNX model for inference.

The model predicts the sentiment of the input text.

8. Sentiment Analysis:

The inference result is processed to determine the sentiment of the input text. A binary sentiment classification (Positive/Negative) is assigned based on the model's prediction.

Confidence scores are computed to indicate the strength of the sentiment prediction.

9. Result Output:

The sentiment classification along with confidence scores are printed to the console.

Dependencies:

1. **onnxruntime**: Python API for ONNX Runtime.
2. **numpy**: Library for numerical computing.
3. **openai**: Python client for OpenAI's API.
4. **Boost**: Server built on boost

ML Model Used:

1. Image Classification

Dataset: [Lumpy Skin Disease Detection Classification Dataset](#)

Size of Train and Test Data: 730 , 82 (404 positive , 408 negative)

Accuracy: train: 94.11, test : 98.78

2. Text

Dataset: [Sentiment Analysis Dataset | Kaggle](#)

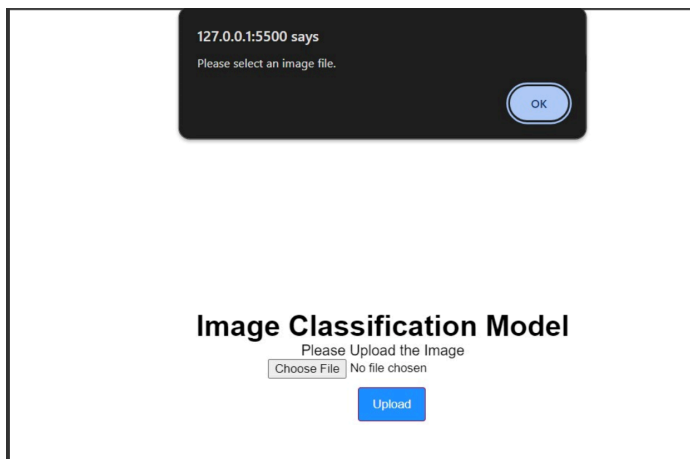
Size of Train and Test Data: 27482,3535

Accuracy: 94,91.5

Test Cases (Image Model)

- 1) Frontend

- 1.1) Empty Image



- 1.2) Wrong format (not jpg)

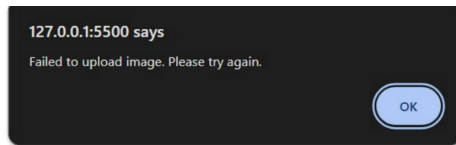
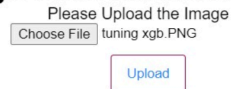


Image Classification Model



1.3) Image size Too large (Image size>1mb)

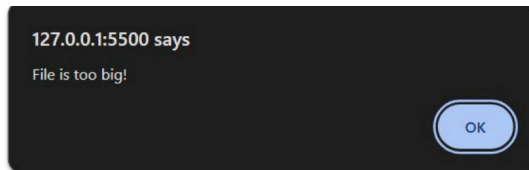
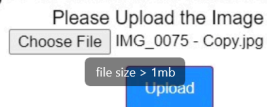


Image Classification Model



Result: Negative on all of the above

2) Server

2.1) Empty request -> Blocked by Javascript

2.2) positive class request

```
Received HTTP request:
Method: POST
Path: /
HTTP version: 11
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary4morwBoIMedE4jMg
Boundary: ----WebKitFormBoundary4morwBoIMedE4jMg
GOT REQUEST STRING
Starting inference...
Running inference...
Image Path: C:\Users\hp\source\repos\boost_test\boost_test\output.jpg
Original Image Size: [640 x 640]
Resized Image Size: [224 x 224]
Running session...
Session run time: 104 ms
Inference completed.
Predicted class: 0
```

2.3) Negative class request


```
Received HTTP request:
Method: POST
Path: /
HTTP version: 1.1
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryAkgIlea9XnYLrksS
Boundary: ----WebKitFormBoundaryAkgIlea9XnYLrksS
GOT REQUEST STRING
Starting inference...
Running inference...
Image Path: C:\Users\hp\source\repos\boost_test\boost_test\output.jpg
Original Image Size: [640 x 640]
Resized Image Size: [224 x 224]
Running session...
Session run time: 126 ms
Inference completed.
Predicted class: 1
```

Test Cases (Text Model)

1) Frontend

1.1) Empty Text Field

Text Sentiment Detection

  Please fill out this field.

1.2) Text length too long

Text Sentiment Detection

Result: Negative on all of the above

2) Server

2.1) Empty request -> Rejected at frontend

2.2) positive class request

```
Received text: This is positive test  
Inference Result: Positive 0.9879519939422607  
|
```

2.3) Negative class request

```
Servers listening on ports: 8080, 8081  
  
Received text: This is negative test  
Inference Result: Negative 0.9830383062362671  
|
```