

**KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY**

**PROJECT SCHOOL CERTIFICATE**

**Title:**  Medical Image Captioning (LLM)

**Mentor:** Dr. Devika Rubi

**Duration:** 2nd April 2024 - 23rd July 2024

**Name:**

**Class:**

**Roll Number:**

**Year:** II-II

Signature of Student Signature of Mentor

**TABLE OF CONTENTS**

* Objective (Deep Learning)……… 3
* Objective (LLM)……………………… 8
* Technical Stack………………………… 3
* Model and Data Description…… 3 - 11
* MERN……………………………………… 12 - 22
* Project Screenshots………………… 22 - 23
* Platforms Used………………………… 24
* References………………………………… 24
* Summary…………………………………… 24

**PART-1: DEEP LEARNING**

**Objective:**Develop a CheXNet model, a type of Convolutional Neural Network (CNN), based on the DenseNet-121 architecture to accurately classify chest X-ray images as 'Effusion' or 'Normal'. By utilizing DenseNet-121's efficient parameter usage and ability to capture intricate features, the model aims to enhance classification precision. This tool is intended to assist radiologists in the early detection of pleural effusion, improving patient outcomes through timely medical intervention.

**Technical Stack:**

|  |  |
| --- | --- |
| **Frontend** | React JS |
| **Backend** | MongoDB Atlas |
| **Authentication** | Twilio |
| **Model** | CNN |
| **Backend API** | Flask |

**Deep Learning Description:**

**Data Set**

# Name: NIH Chest X-ray Dataset

**Download Link:** [**https://www.kaggle.com/datasets/nih-chest-xrays/data**](https://www.kaggle.com/datasets/nih-chest-xrays/data)

This NIH Chest X-ray Dataset is comprised of 112,120 X-ray images with disease labels from 30,805 unique patients. The dataset was split into 12,292 images for training, 1,792 for validation, and 3,600 for testing. Data augmentation techniques, including vertical and horizontal flips, were applied to enhance model robustness.

|  |  |  |
| --- | --- | --- |
|  | Effusion | Normal |
| Train | 6146 | 6146 |
| Test | 1800 | 1800 |
| Validation | 896 | 896 |

TensorFlow is an open-source machine learning library developed by Google that provides a comprehensive platform for building and deploying deep learning models. keras, on the other hand, is a high-level neural networks API that runs on top of TensorFlow.

**Model**

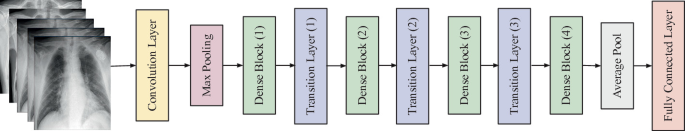
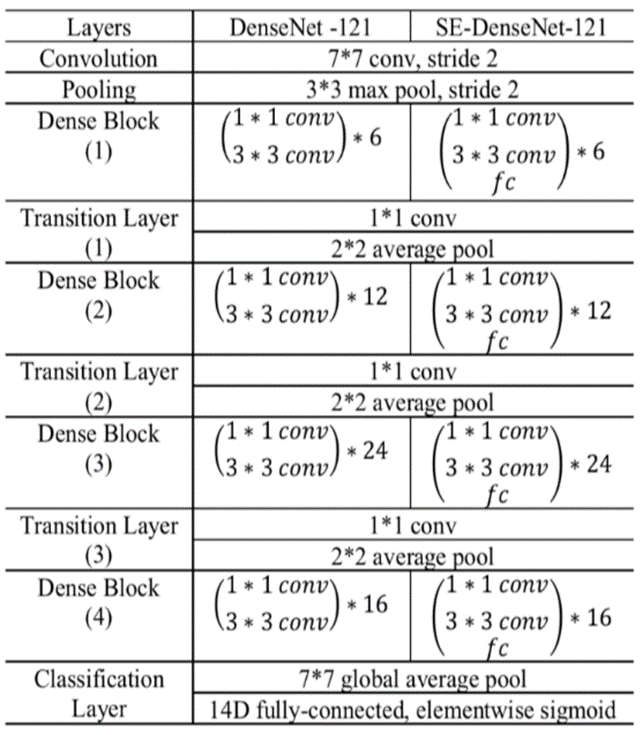
**CheXNet:** CheXNet is a deep learning model specifically designed for the detection of 14 thoracic diseases in chest X-ray images. It was developed by researchers at Stanford University and is based on the DenseNet-121 architecture. CheXNet can classify chest X-rays into multiple disease categories, significantly aiding radiologists by providing a second opinion and potentially improving diagnostic accuracy and speed.

**DenseNet-121:** DenseNet-121 is a variant of the DenseNet (Densely Connected Convolutional Networks) family, which is known for its dense connectivity pattern. In DenseNet, each layer receives inputs from all preceding layers, enhancing feature reuse and improving gradient flow during training. DenseNet-121 consists of 121 layers and is particularly efficient in terms of parameter usage while achieving high accuracy. This architecture is well-suited for tasks requiring detailed image analysis, such as medical image classification, due to its ability to capture intricate features through dense connections.

C:\Users\Admin\AppData\Local\Packages\5319275A.51895FA4EA97F_cv1g1gvanyjgm\TempState\5A0948634378D18B07E1615C0F953076\WhatsApp Image 2024-07-21 at 17.56.43_87a1495f.jpg

****

**Architecture:**

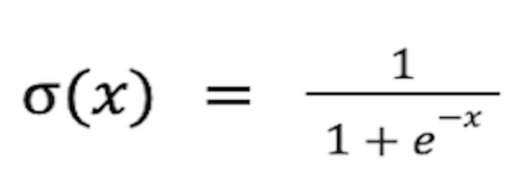
**Global Average Pooling:**

Global Average Pooling (GAP) is a layer used in convolutional neural networks that reduces each feature map to a single value by computing the average of all values in that feature map, often used before the final classification layer to minimize overfitting and model size.

**Dense Layer:**

Dense layers are fully connected layers in a neural network where each neuron in a layer is connected to every neuron in the preceding layer or the input layer in the case of the first dense layer.

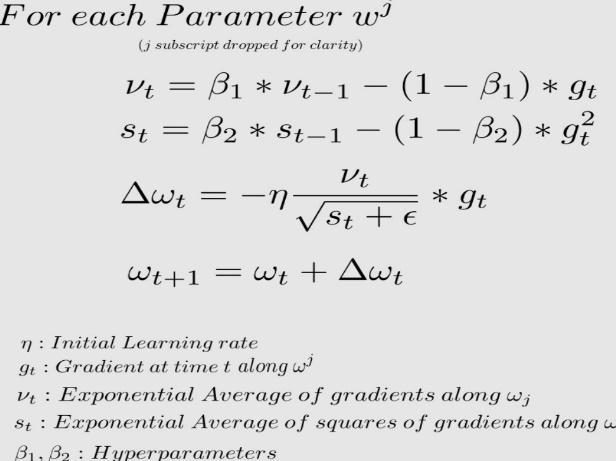
**Sigmoid Activation Function:**



The sigmoid activation function maps input values to a range between 0 and 1. It is commonly used for binary classification tasks, outputting probabilities that indicate the likelihood of a particular class.

**Adam Optimiser**

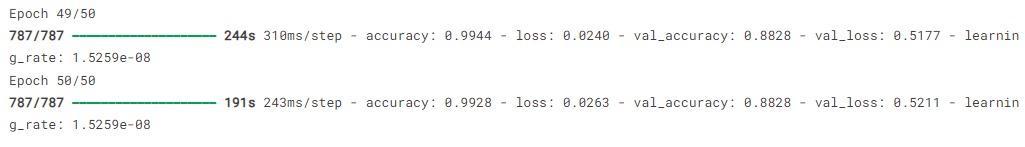
**C:\Users\Admin\AppData\Local\Packages\5319275A.51895FA4EA97F_cv1g1gvanyjgm\TempState\E7FED988DD5F2A44BB605BFFD0F6C19D\WhatsApp Image 2024-07-21 at 18.10.47_f15ab83b.jpg**



Adaptive Moment Estimation is an algorithm for optimization technique for gradient descent. It requires less memory and is efficient. Intuitively, it is a combination of the gradient descent with momentum algorithm and the RMSP algorithm. Its adaptive nature adjusts learning rates for each parameter individually, making it robust to varying scales and sparsity.

**Accuracy:**

**Train Accuracy:**



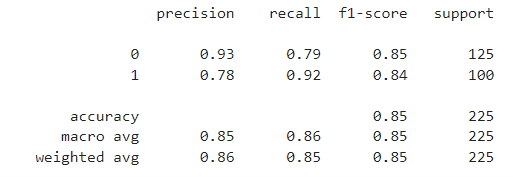
**Test Accuracy:**

****

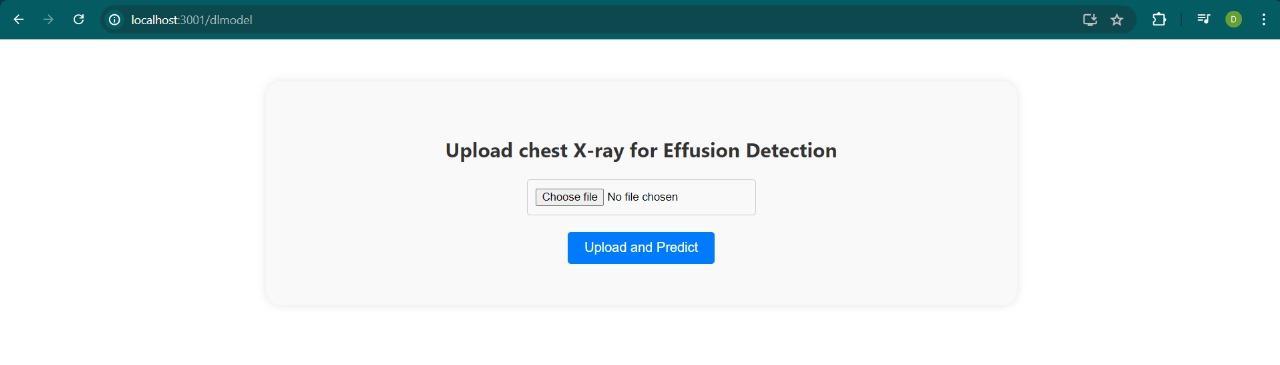
**Validation Accuracy:**

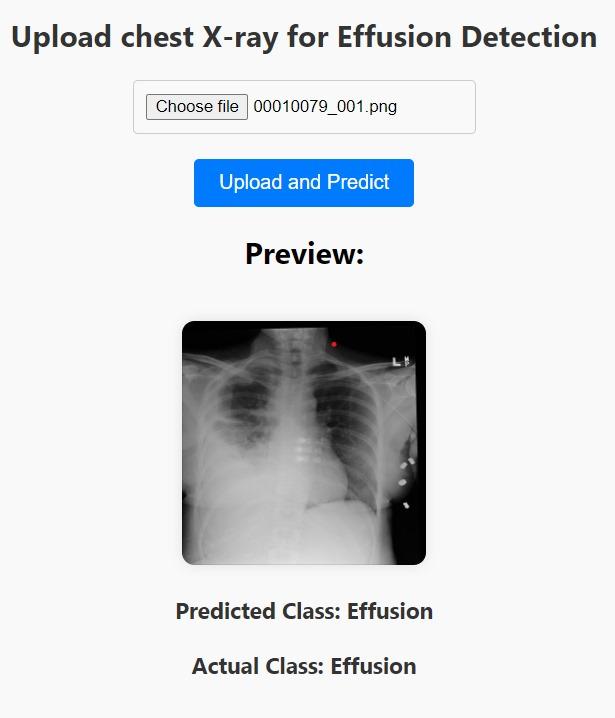
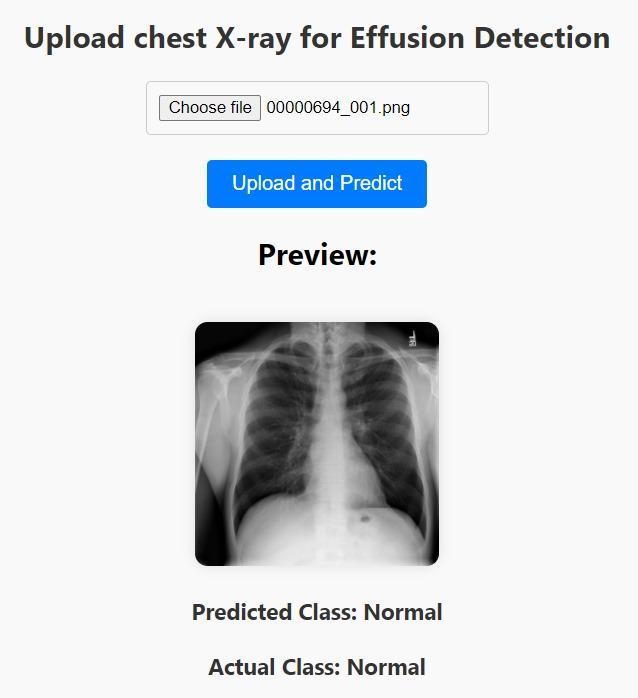
****

**Classification Report**



**FRONTEND DESIGN**

****

** **

**PART-2: LLM**

**Objective:** Develop a report generation system using Long Short-Term Memory (LSTM) networks, leveraging feature vectors extracted from chest X-ray images alongside word embeddings to generate detailed textual reports. This system will utilize the Inception V3 architecture for image feature extraction and LSTM layers for natural language processing to provide comprehensive medical documentation. The goal is to improve the efficiency and accuracy of radiological reporting by integrating image-based features with natural language understanding.

**LLM Description:**

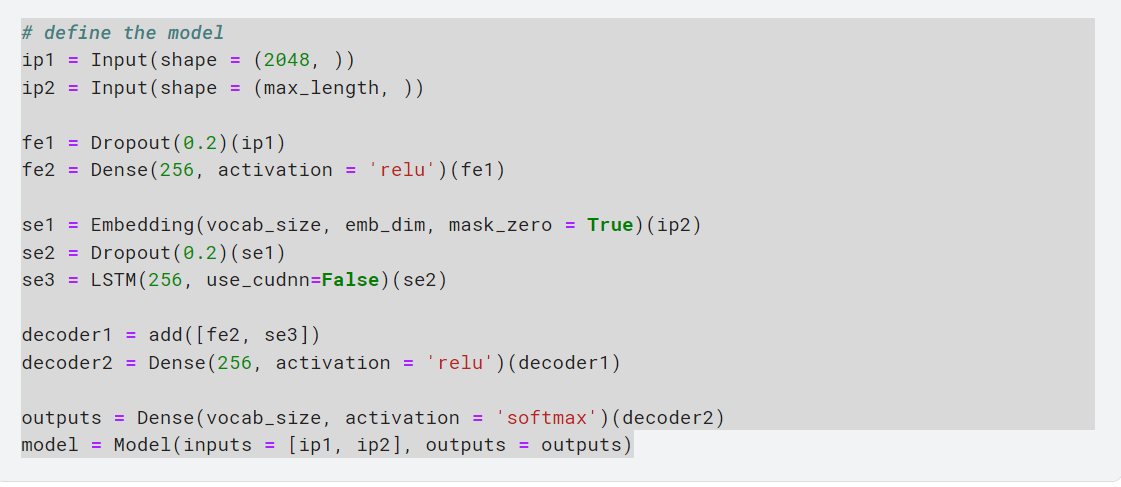
**DATASET :- Chest X-rays (Indiana University)**

Download Link : [**https://www.kaggle.com/datasets/raddar/chest-xrays-indiana-university**](https://www.kaggle.com/datasets/raddar/chest-xrays-indiana-university)

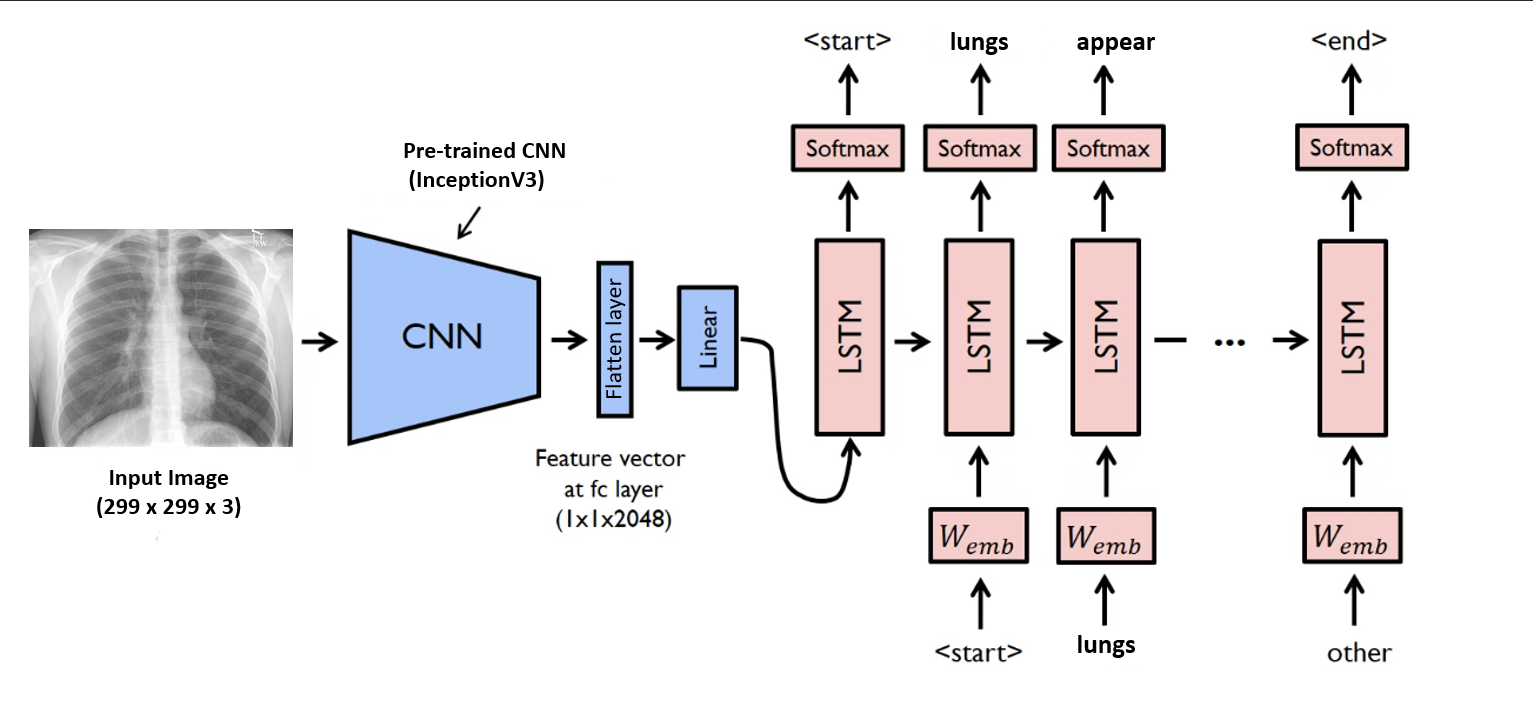
The Chest X-rays dataset from Indiana University, available on Kaggle, contains a collection of chest X-ray images paired with diagnostic reports. This dataset includes 7,470 images along with 3,955 corresponding medical reports. It is widely used for tasks such as image captioning and radiology report generation.

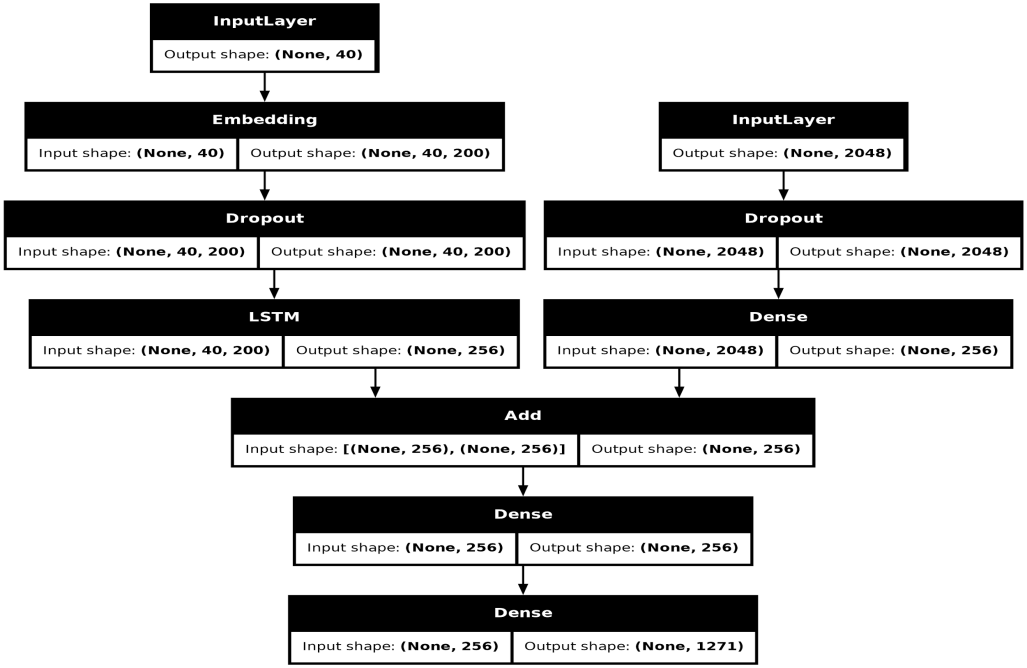
Key features of this dataset include:

1. Variety of Diagnoses: The dataset includes images with various diagnoses, ranging from normal findings to different types of pathologies.
2. Paired Reports: Each X-ray image is paired with a detailed radiology report, which can be used for natural language processing tasks.
3. Medical Research Utility: This dataset is particularly useful for developing and testing machine learning models in the medical field, such as for automated diagnosis or report generation.



**Architecture:**





**Dense**

Dense layers are fully connected layers in a neural network where each neuron in a layer is connected to every neuron in the preceding layer or the input layer in the case of the first dense layer.

**LSTM (Long Short-Term Memory)**

An LSTM (Long Short-Term Memory) layer is a type of recurrent neural network (RNN) layer designed to overcome the limitations of traditional RNNs, particularly in handling long-term dependencies and mitigating the vanishing gradient problem. LSTMs are effective in processing and predicting sequences, making them suitable for tasks such as time series forecasting, natural language processing, and speech recognition.

**ReLU (Rectified Linear Unit)**

The ReLU (Rectified Linear Unit) activation function is widely used in neural networks, particularly in deep learning models, due to its simplicity and effectiveness. It introduces non-linearity to the model, allowing it to learn complex patterns.

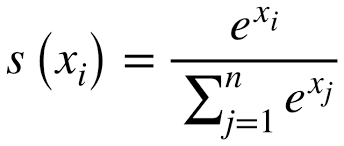
Formula and Operation:

The ReLU function is defined as: ReLU(x)=max(0,x)

**Softmax activation function**

The Softmax activation function is commonly used in the output layer of neural networks for multi-class classification problems. It transforms the raw output scores (logits) of the network into probabilities, which sum to 1, making it suitable for probabilistic interpretation.

**Formula**



**CNN MODEL :-**

InceptionV3 is a deep learning model for image classification, part of the Inception family of neural networks developed by Google. It was introduced in 2015 and builds upon the success of its predecessors, InceptionV1 and InceptionV2, by incorporating several enhancements to improve efficiency and accuracy.

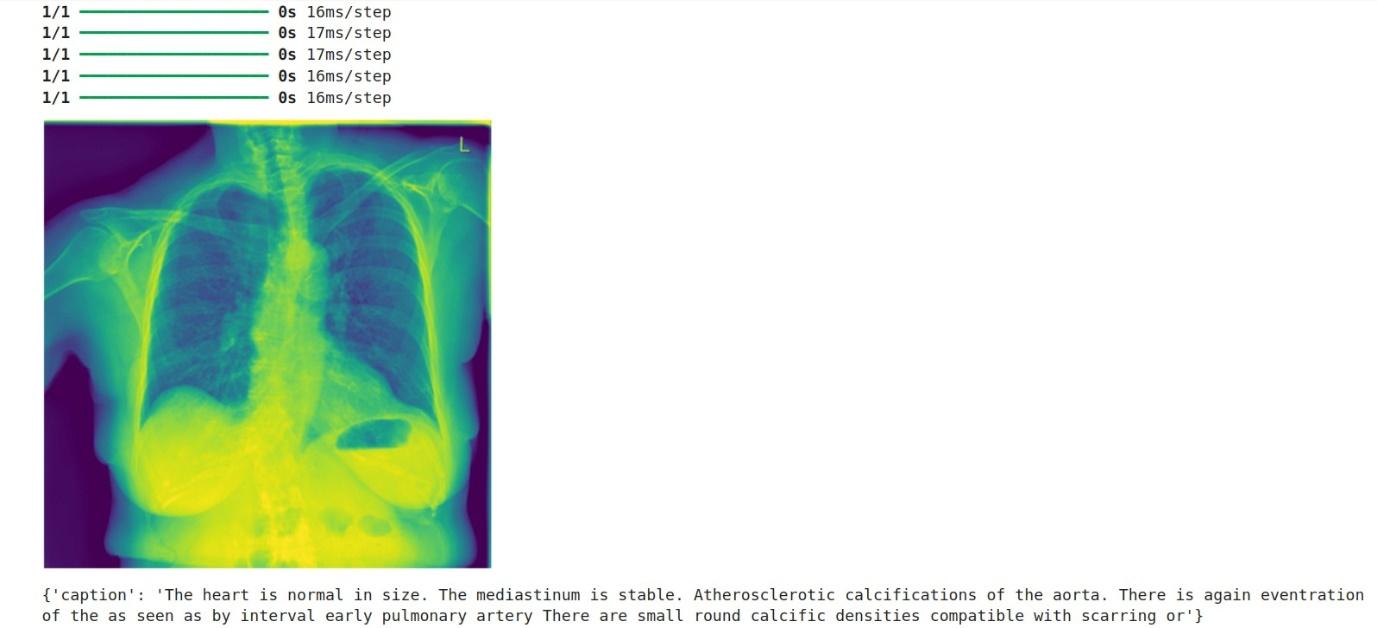
The key features of InceptionV3 include:

1. **Inception Modules**: These are the core building blocks of the network, designed to capture various levels of feature abstractions by using multiple filters of different sizes in parallel.
2. **Factorized Convolutions**: This technique splits convolutions into smaller, more manageable operations, reducing computational cost and improving network performance.

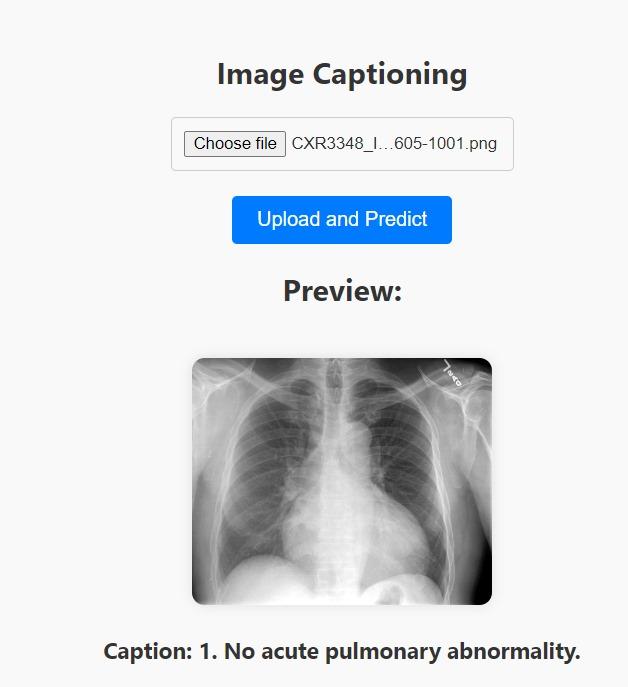
Word2Vec is a popular word embedding technique introduced by Google researchers in 2013, specifically for the task of transforming words into continuous vector representations. It is implemented in the Gensim library, which is widely used for natural language processing tasks. Word2Vec uses shallow, two-layer neural networks to learn the vector representations of words from a large corpus of text. The primary objective is to place similar words closer together in the vector space. There are two main approaches within Word2Vec:

1. **Continuous Bag of Words (CBOW)**: In this approach, the context (surrounding words) is used to predict the target word. It averages the vectors of context words to predict the center word, which makes it faster and more efficient with large datasets.
2. **Skip-Gram**: This method does the opposite; it uses the current word to predict the surrounding context words. It is particularly effective for smaller datasets and capturing rare words or phrases.

**Model Prediction :**

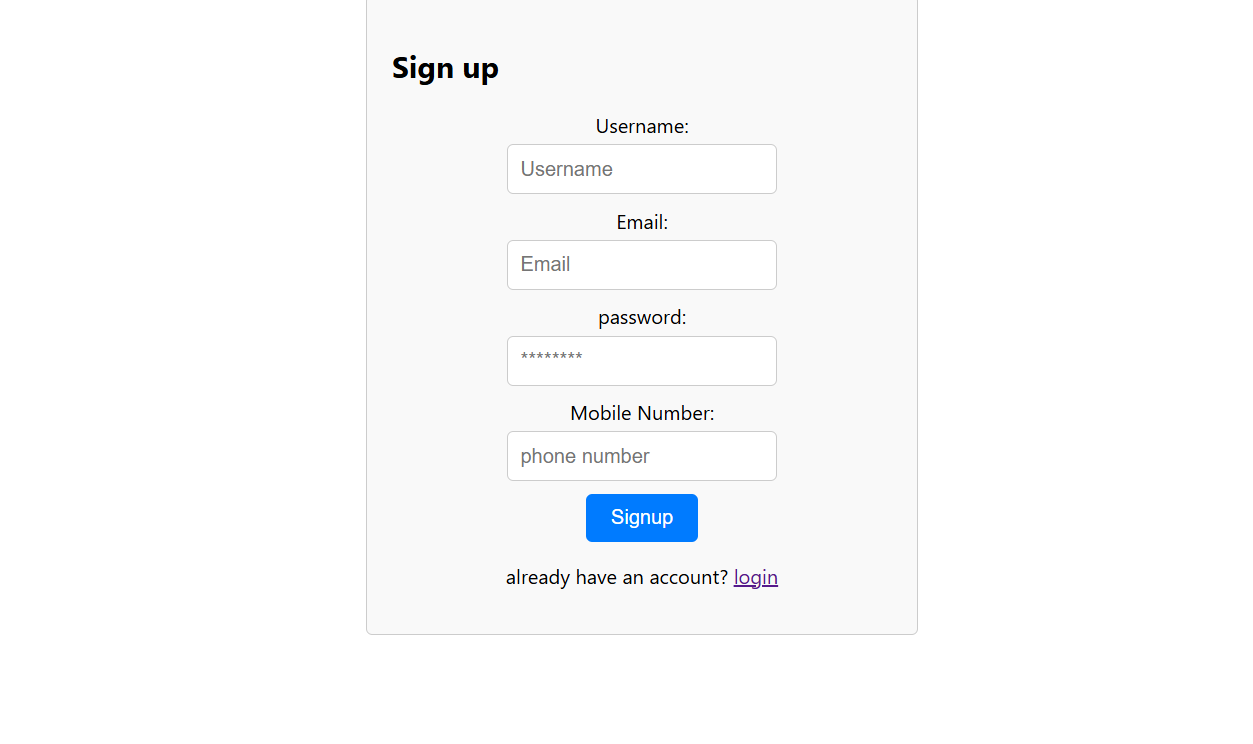
****

**FRONTEND DESIGN**



# MERN:

Register Page: Register using name, email, phone Number and password.



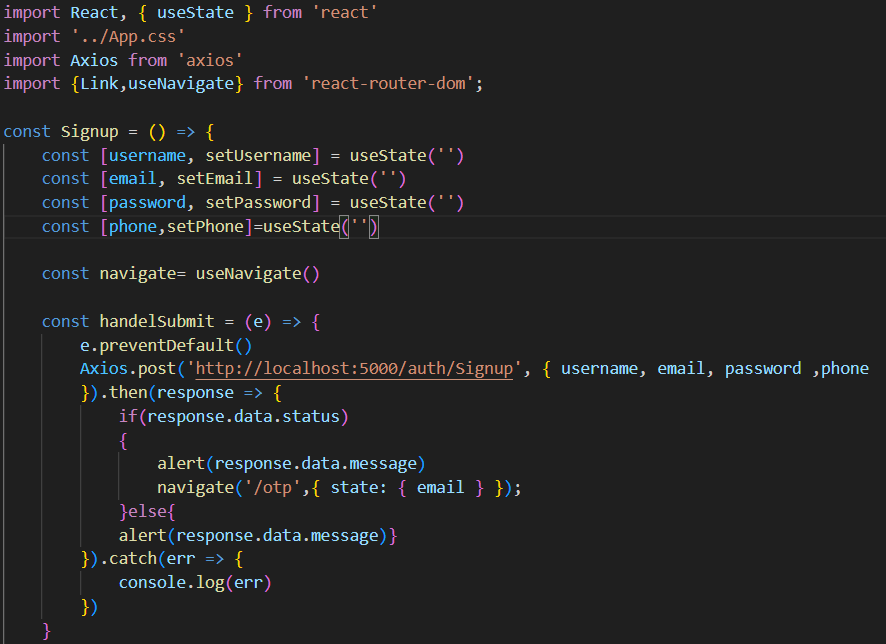
Frontend Code:

React: Main library for creating the component.

useState: React hook for managing state within the component.

Axios: HTTP client for making requests to the backend.

Link, useNavigate: React Router components for navigation and linking.

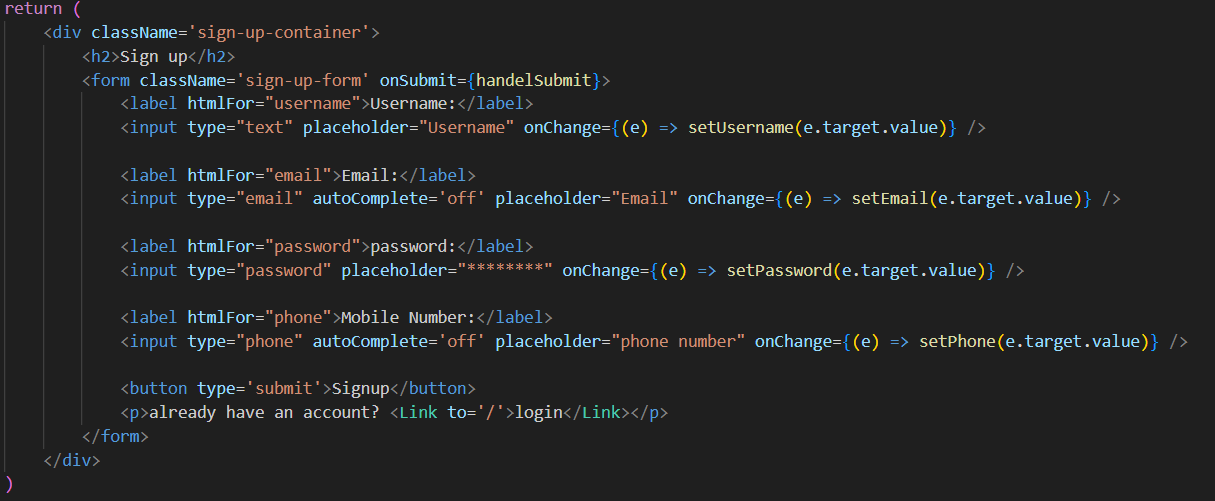


State Variables: Stores the user's input.

Provides navigation functionality to redirect users after successful signup

Sends a POST request to the signup endpoint with user details.

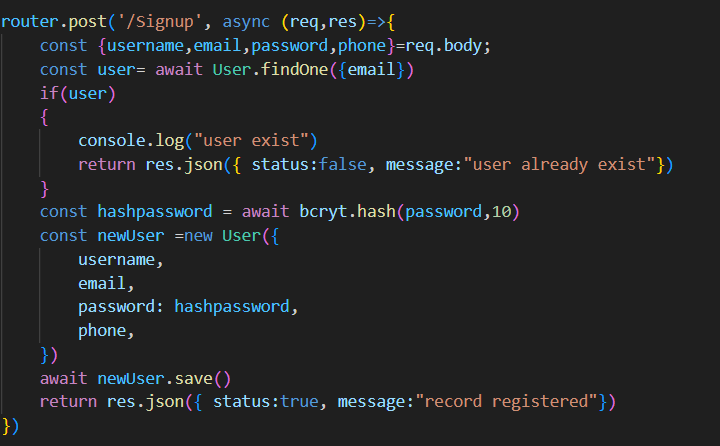
HTML code:



Form Structure:

Input: Captures the user's details.

Backend Code:



Handles user registration.

Hashes the password using bcrypt.

User Schema for Register:



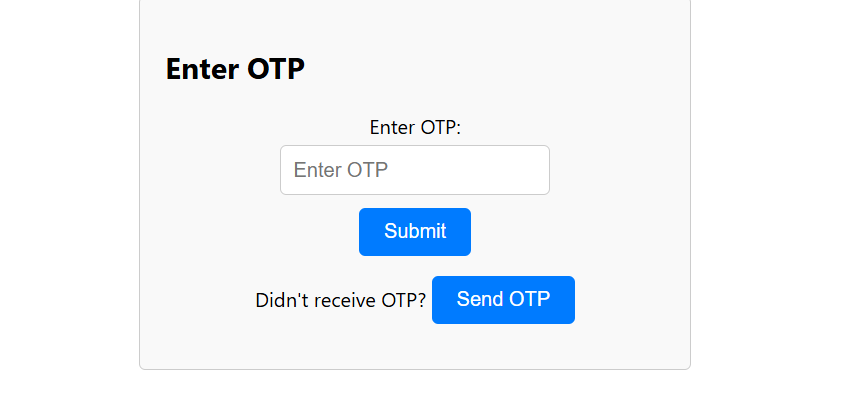
userSchema: This defines the structure of the documents in the User collection.

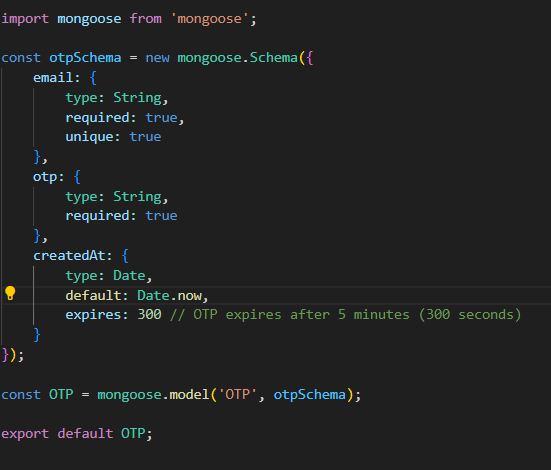
Model Creation: Creates a model for interacting with the User collection.

Export: Makes the User model available for import in other files.

OTP page:

After registering, you will be redirected to the otp page. Enter the received otp on your registered email/Phone Number.





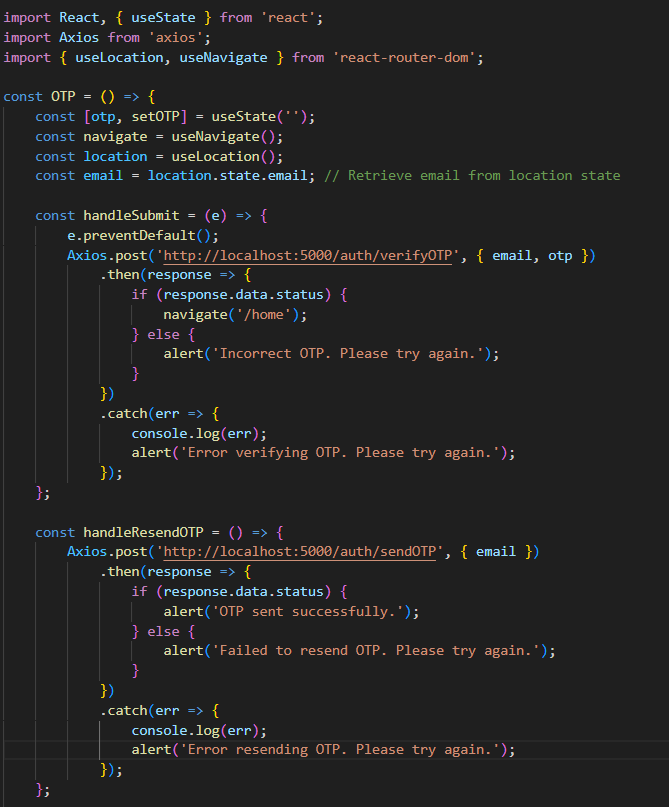
imports the mongoose library, which is used for interacting with MongoDB and managing data models.

otpSchema: This defines the structure of the documents in the OTP collection.

Export: Makes the OTP model available for import in other parts of your application.

Export: Makes the OTP model available for import in other parts of your application.

Frontend:



useState: Initializes state for otp to store the user's input.

useNavigate: Hook to programmatically navigate to different routes.

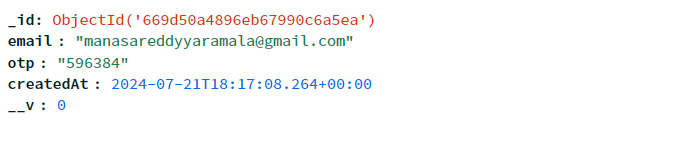
useLocation: Hook to access location state, used here to get the email passed from the previous page.

Sends the email and otp to the server for verification.

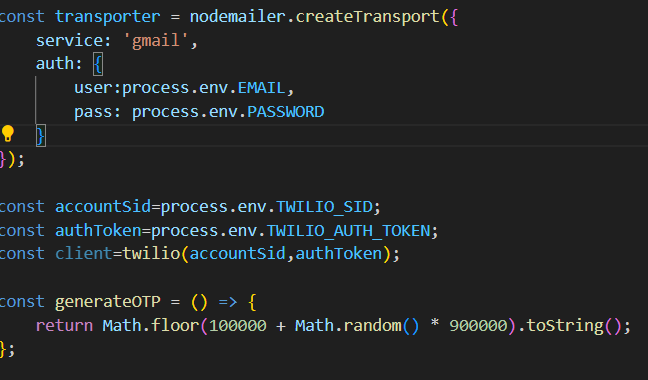
Sends a request to resend OTP to the server.



Email along with otp is also saved in mongo Database during Sign up.

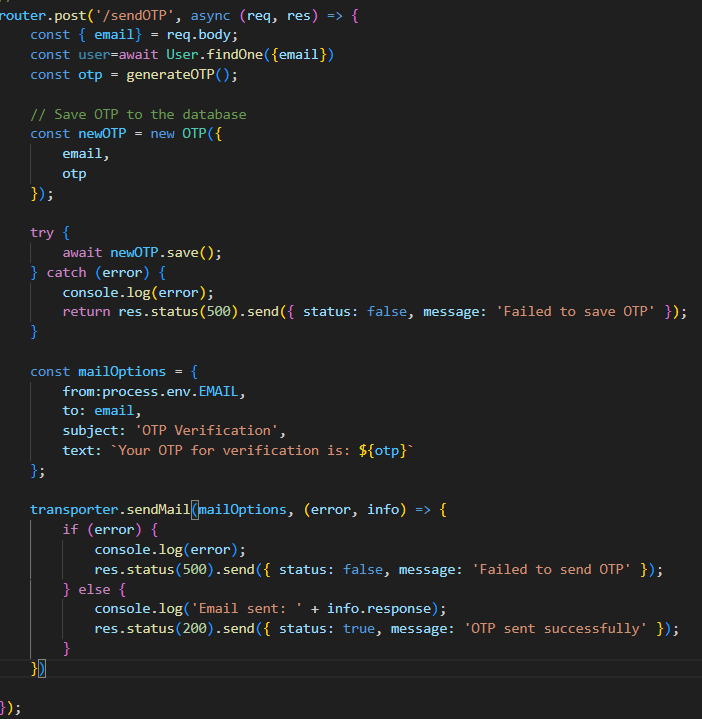


Backend code

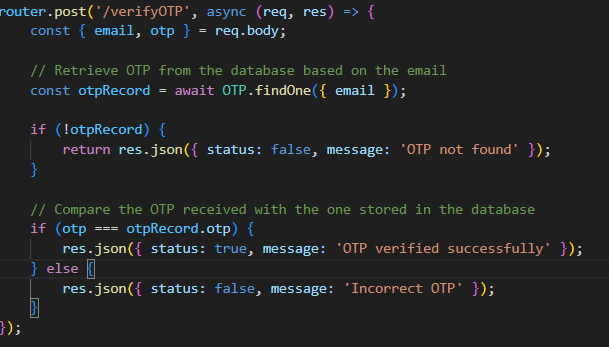


Configures the transporter for sending emails via Gmail with authentication credentials.

function to generate a 6-digit OTP.

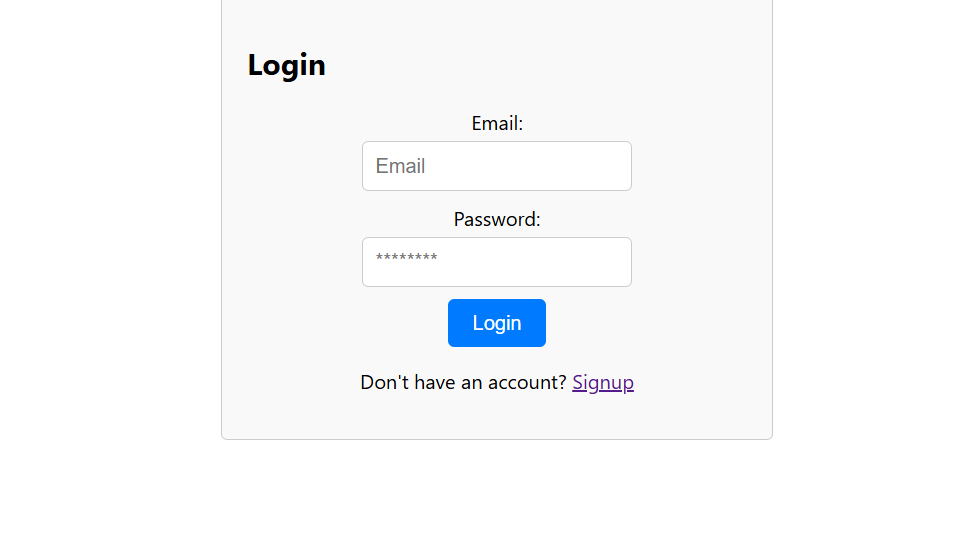


/sendOTP Route: Handles sending OTP via email and SMS.

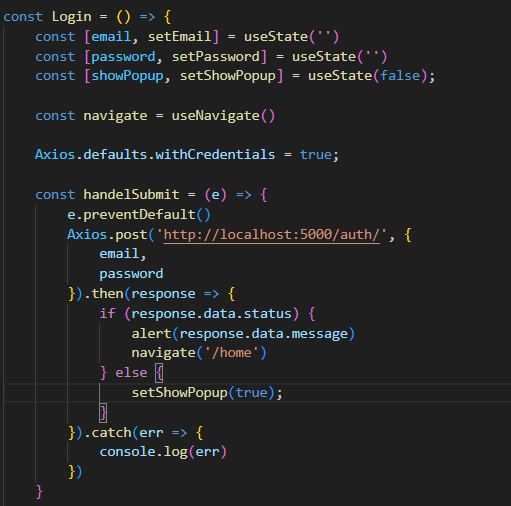


/verifyOTP Route: Handles OTP verification.

Login page



Frontend



State Variables:Stores the user's input.

Provides navigation functionality to redirect users after a successful login.

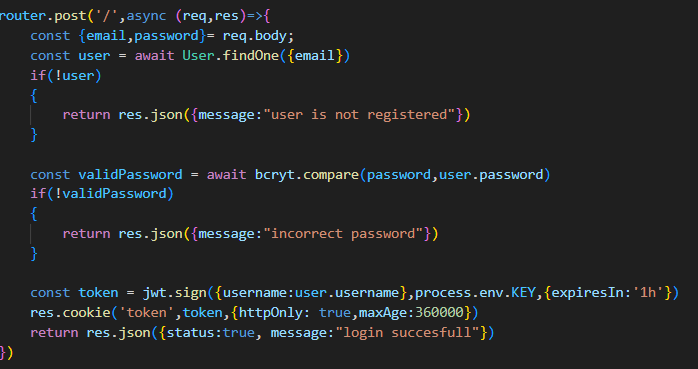
Sends a POST request to the login endpoint with the email and password.

Login HTML

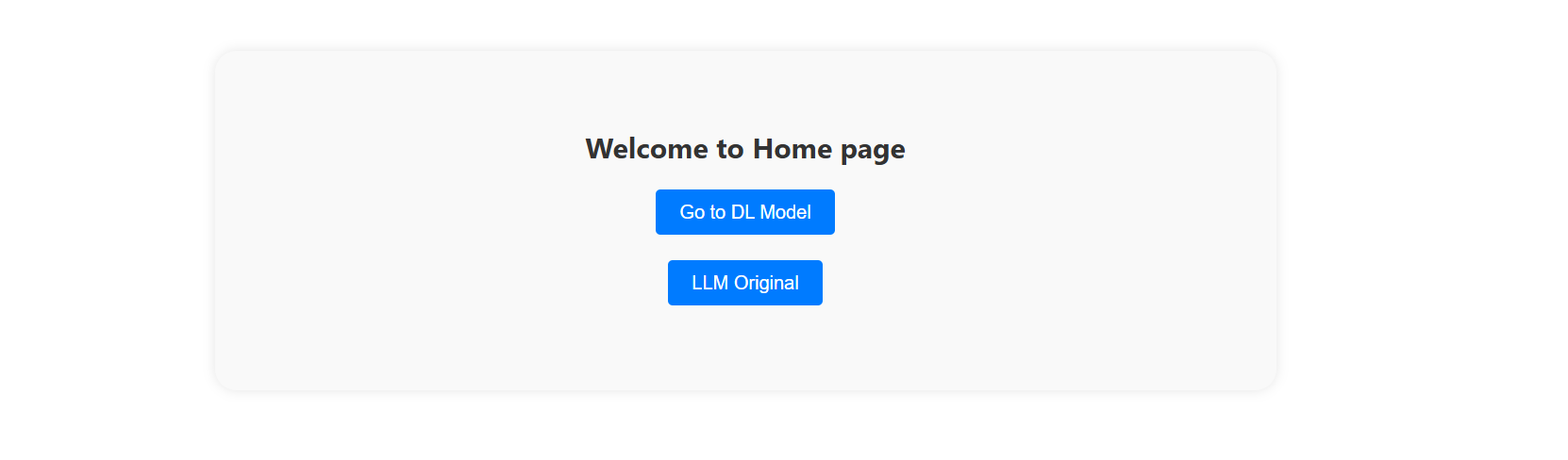


Backend

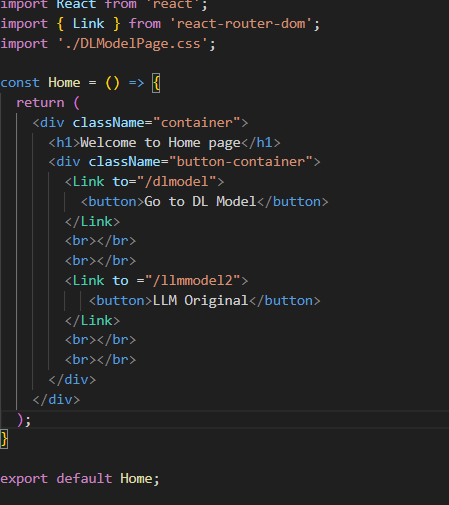
Route: Handles user login.



Home page



Frontend



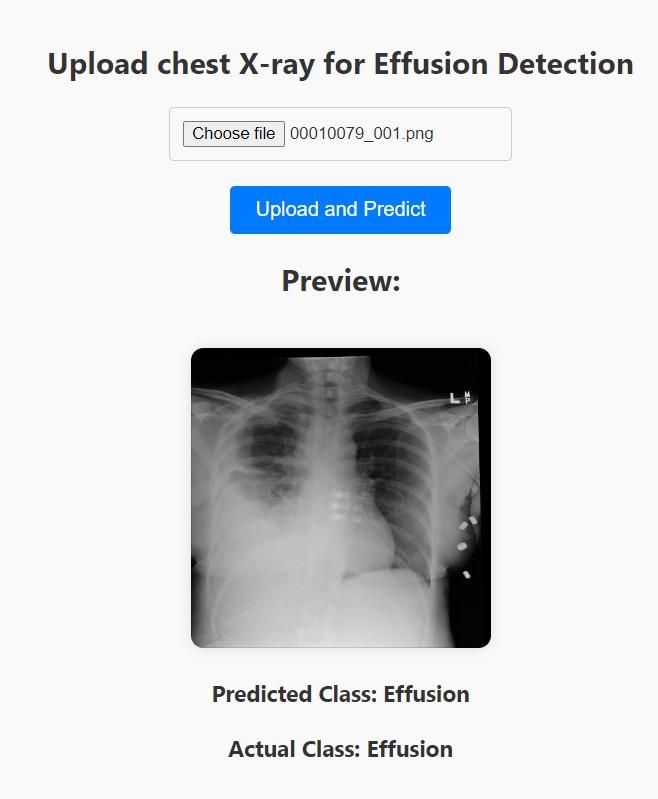
Clickable elements inside each Link for navigation.

Link Components: Navigation links that redirect to different routes.

Exports the component so it can be imported and used in other parts of the application.

In Home page, you can select the model you want to use for prediction i.e Deep Learning, LLM

Deep Learning



Frontend

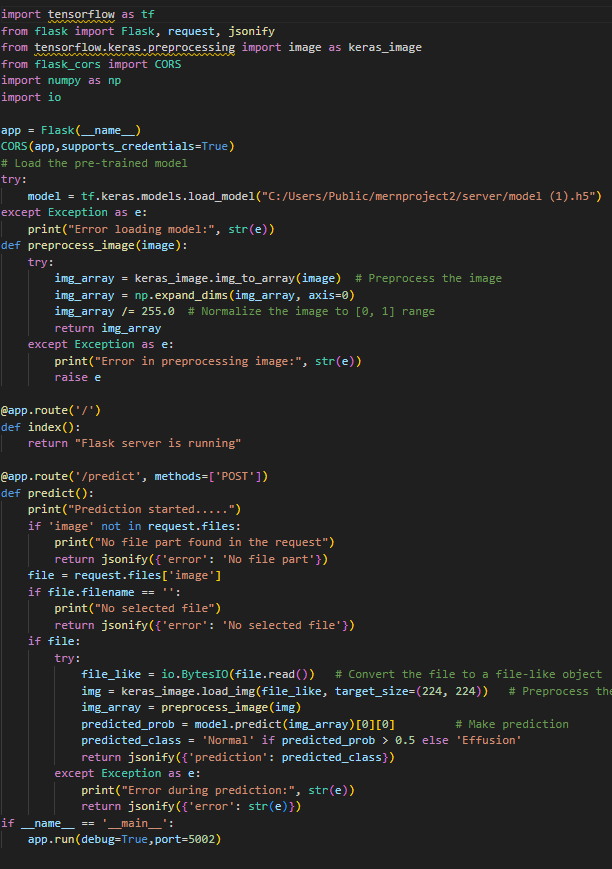


DLModelPage: A functional component that manages file uploads and predictions.

State Variables: Stores the user's input.

Sends a POST request to the server with the image file.

Backend



Ensures the app runs only if the script is executed directly.

Ensures the app runs only if the script is executed directly.

Necessary libraries for TensorFlow, Flask, and image handling.

Initializes the Flask application.

Enables CORS on the Flask app, allowing it to handle requests from different origins.

@app.route('/'): Defines a route for the root URL.

Defines a route for handling POST requests to /predict.

LLM

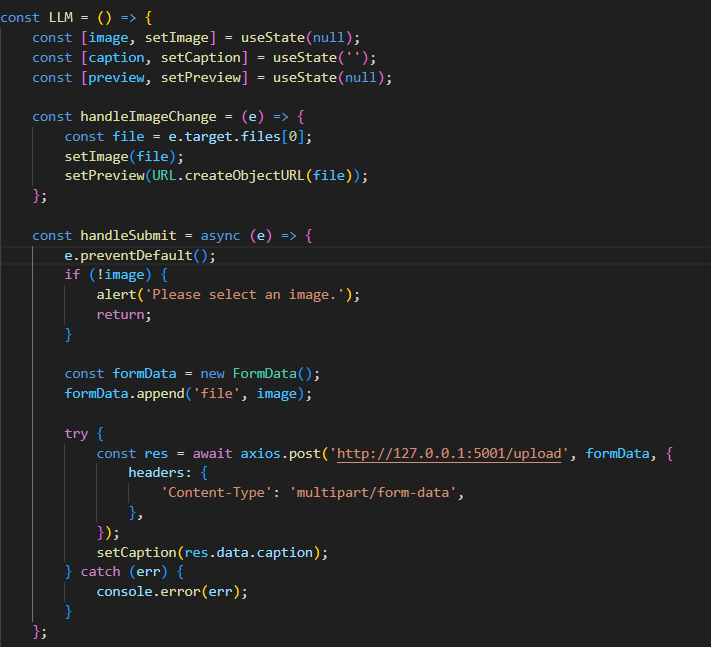
State Variables:

image: Stores the selected image file.

caption: Stores the caption generated by the model.

preview: Stores the URL for the image preview.

Frontend

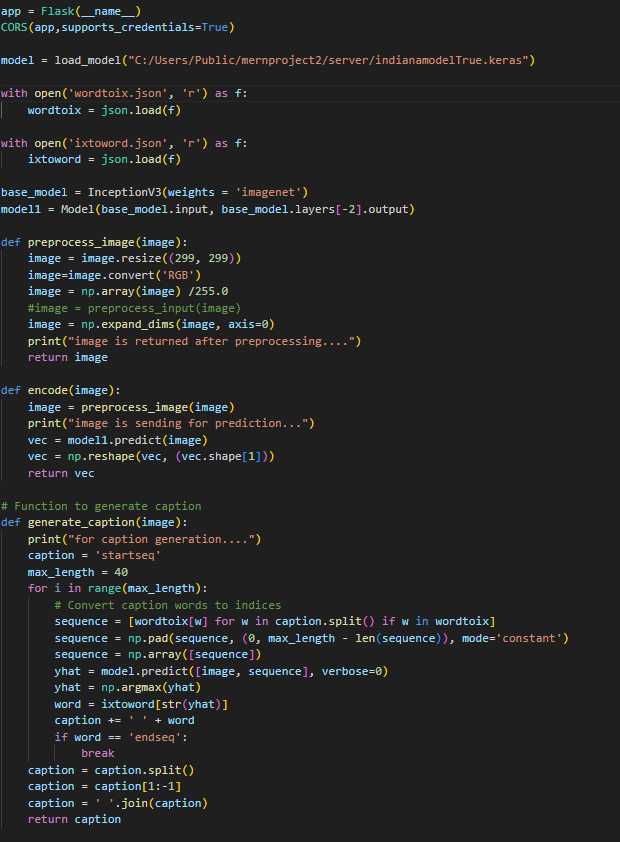


Updates the image state with the selected file and sets a preview URL for the image.

Sends a POST request to the server with the image file.

Backend

Necessary libraries for TensorFlow, Flask, and image handling



Enables CORS on the Flask app, allowing it to handle requests from different origins.

Image Preprocessing Function

Encode Image Function

Generate Caption Function

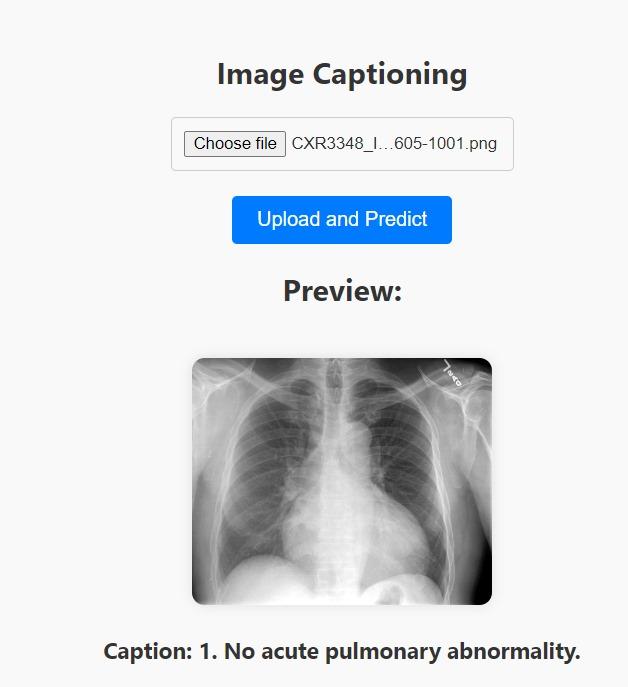
LLM Flask



Defines a route for handling POST requests to /upload.

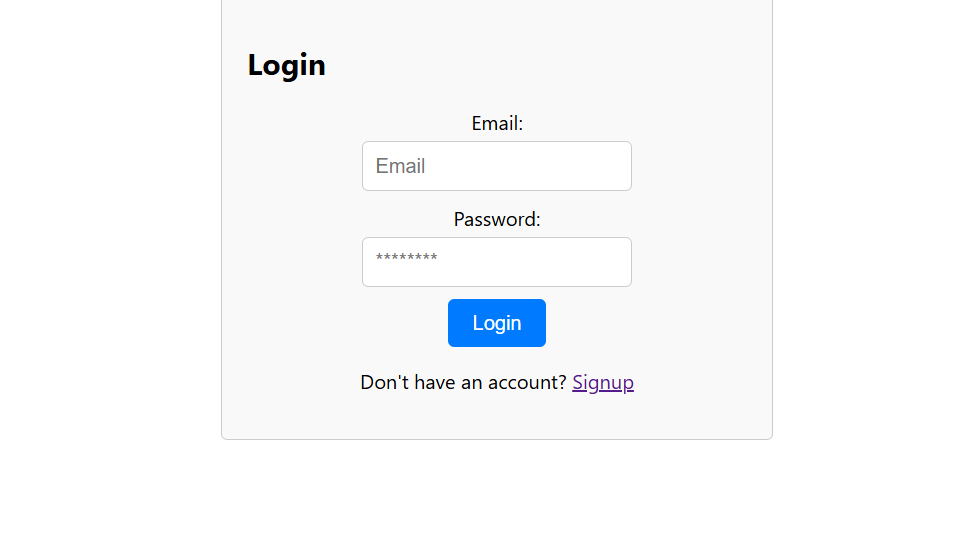
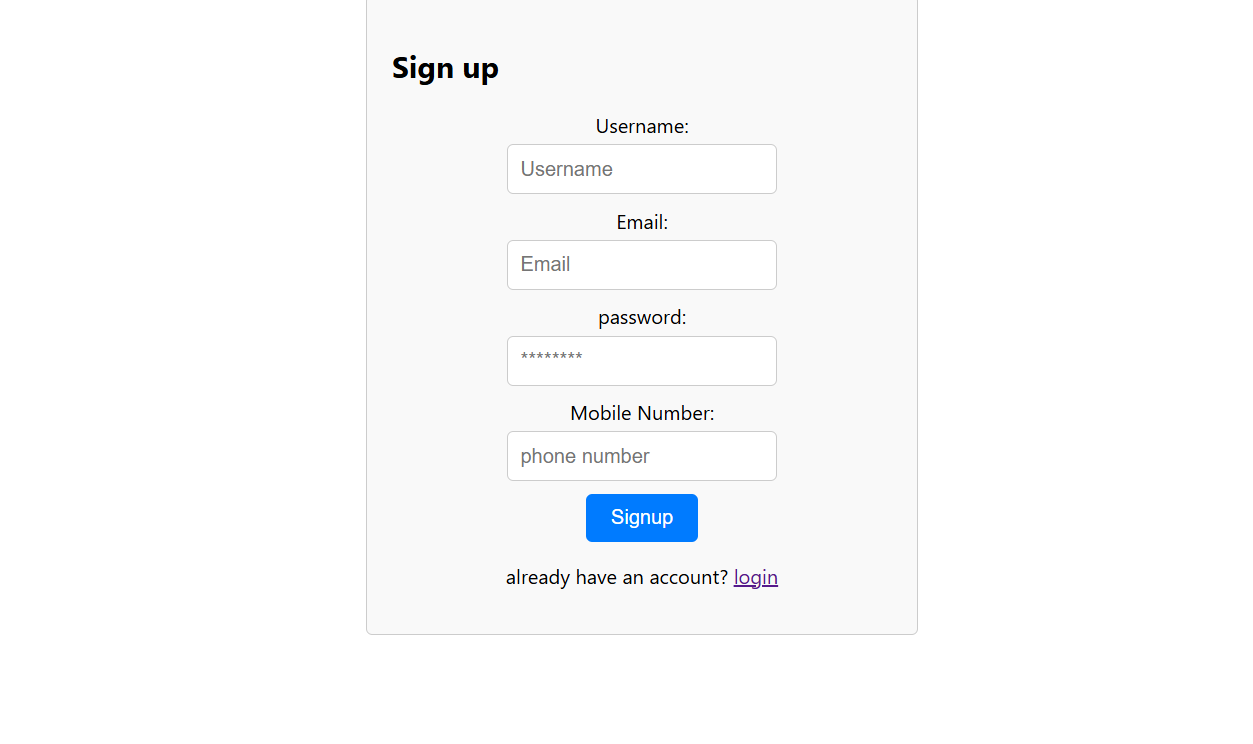
Ensures the app runs only if the script is executed directly.

Starts the Flask development server with debug mode enabled.

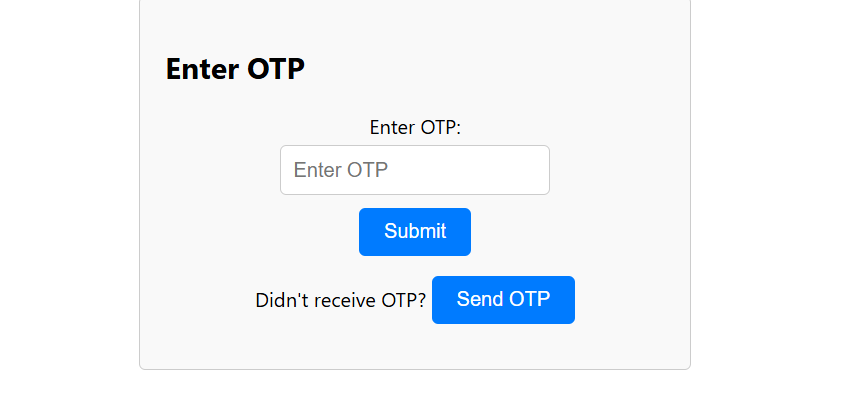


**PROJECT SCREENSHOTS**

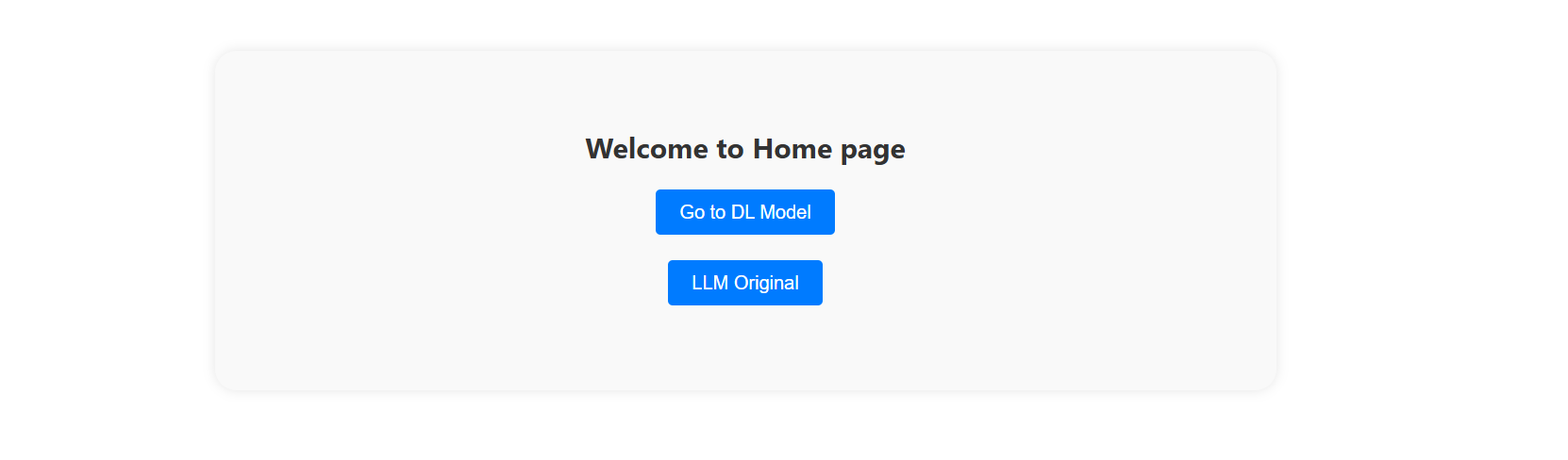
**Login page Sign-up page**

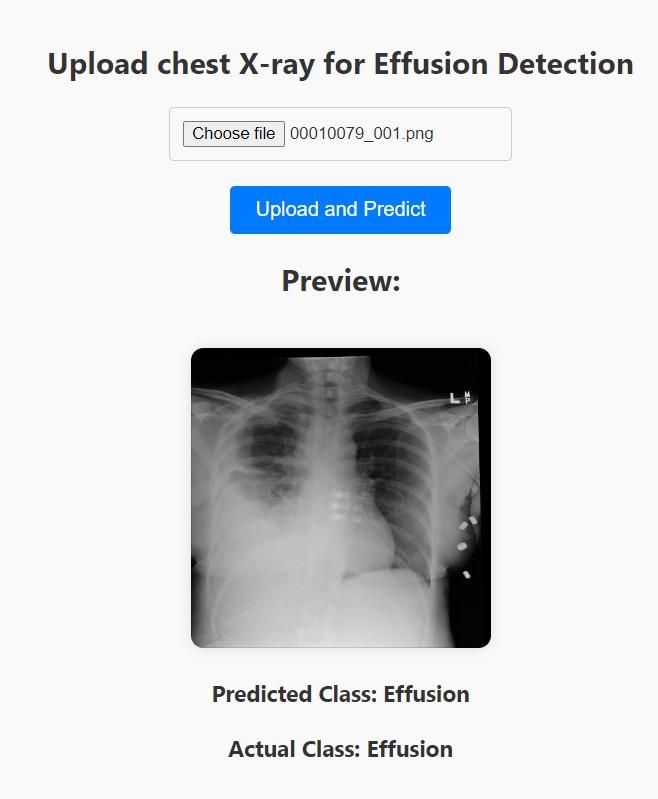
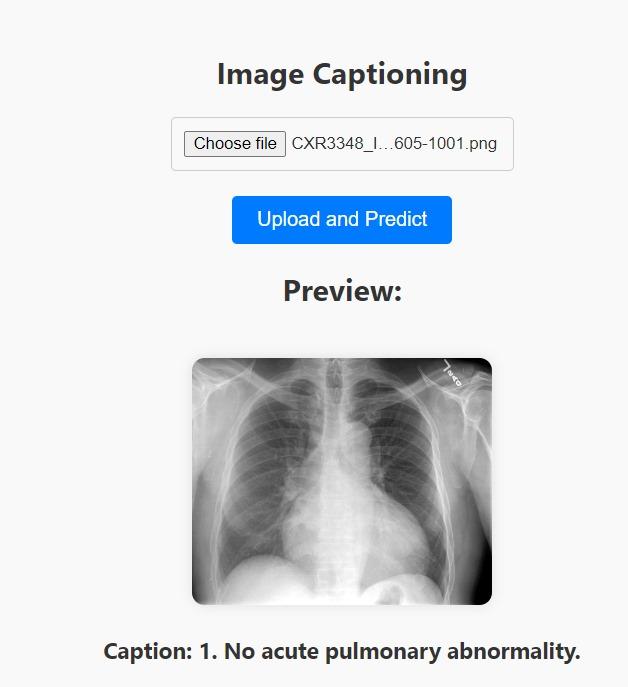
**OTP page**



**Home page**



**Deep learning page LLM page**

**PLATFORMS USED**

* Kaggle for datasets and training using GPU.
* Jupyter notebook and VScode for implementing code.
* Twilio for authentication.
* MongoDB Atlas for database to store login credentials.

**REFERENCES**

* OTP login using MERN Stack

[<https://youtu.be/O9hEU_2S0fg?si=wTZlsHr2RXGJPbTe>]

* Image Caption generation using Deep Learning

[<https://www.geeksforgeeks.org/image-caption-generator-using-deep-learning-on-flickr8k-dataset/>]

* National Institutes of Health Chest X-Ray Dataset [<https://www.kaggle.com/datasets/nih-chest-xrays/data>]
* Chest X-Ray Report Generation from Chest-X Ray Images [[https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1224/reports/custom\_117157386.pdf]](https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1224/reports/custom_117157386.pdf)
* Medical Image Captioning on Chest X-Rays [<https://towardsdatascience.com/medical-image-captioning-on-chest-x-rays-a43561a6871d>]

**Summary**

The project on medical image captioning focuses on generating descriptive captions from chest X-ray images. The system utilizes deep learning models to analyse these images and produce relevant textual descriptions that encapsulate key findings and details depicted in the X-rays. Additionally, there is an integrated deep learning model designed to classify the images into two categories: normal and effusion. This dual-functionality enhances the utility of the system by not only providing descriptive insights but also assisting in preliminary diagnosis through image classification.