 | [Course Title] | [Date]

# Table Of Contents:

|  |  |
| --- | --- |
| Objective | 2 |
| Techn0logy Stack | 2 |
| Dataset Description | 3 |
| Model Used | 3 |
| Frontend Design | 6 |
| Flask Framework | 7 |
| Deployment | 8 |
| AWS Sage maker | 11 |
| Screen Shots | 12 |
| References | 15 |

1

Objective

Developing an automated system for chest X-ray image analysis, generating clinically relevant and accurate medical text reports.

The system aims to streamline the diagnostic workflow, provide timely insights for healthcare professionals, and enhance overall efficiency in medical reporting.

Technology Stack

|  |  |
| --- | --- |
|  |  |
| Frontend | Html, CSS, React |
| Backend | MongoDB Atlas |
| Authentication | MongoDB |
| Model | Chex Net  DistilGPT2 |
| Backend API | Flask |
| Deployment | Amazon Web Services  NGROK |

Dataset Description

Chest X Ray Dataset link: Chest X-Ray Indiana University

“ <https://www.kaggle.com/datasets/raddar/chest-xrays-indiana-university> “

|  |  |
| --- | --- |
| Total Dataset | 7430 |
| Training Dataset | 6930 |
| Testing Dataset | 500 |

Model Used

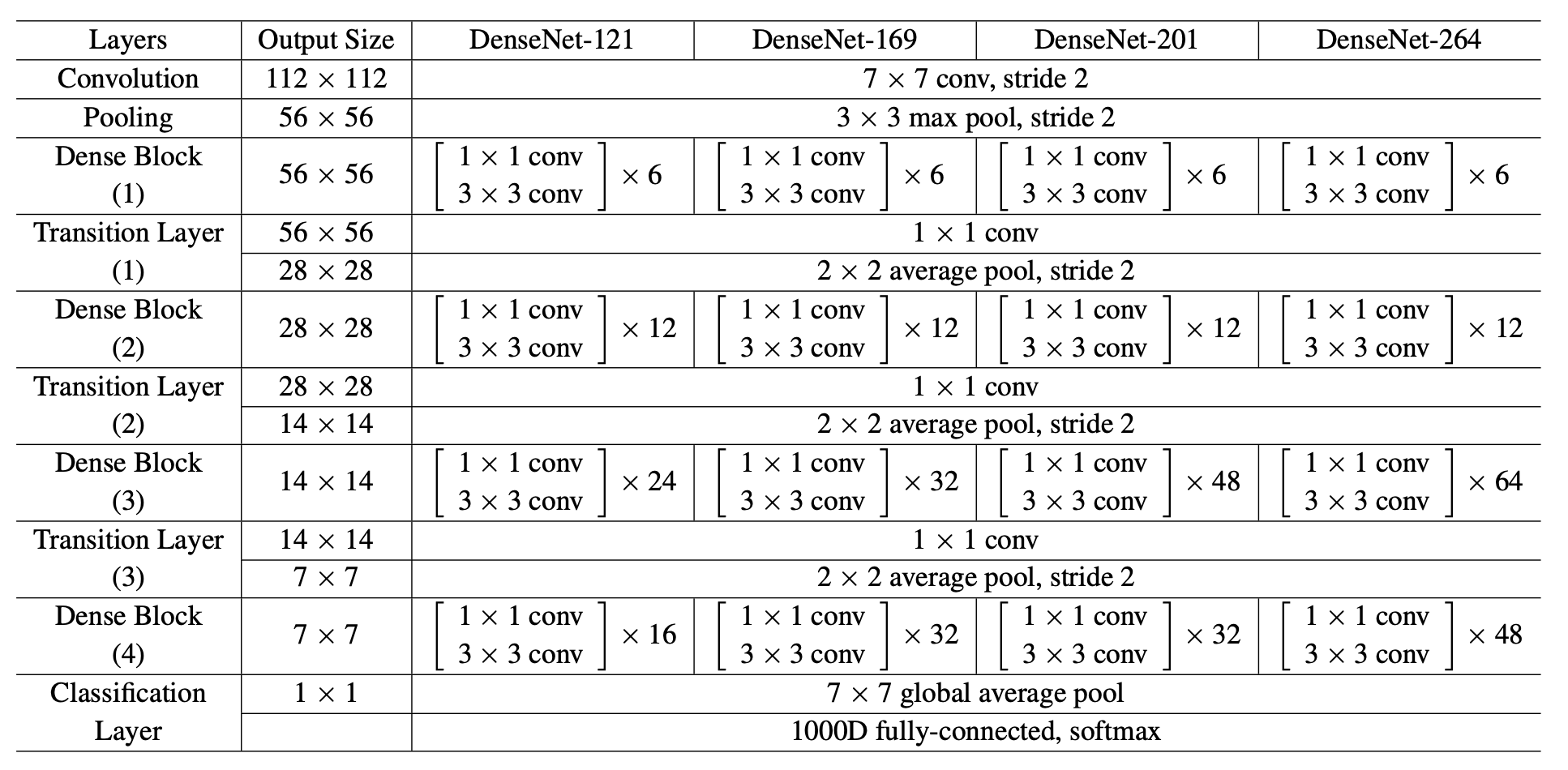
|  |  |
| --- | --- |
| Encoder – Visual Feature Extraction | DenseNet 121 |
| Decoder - Large Language Model | conditioned distil generative pre-trained trans-  former 2 |

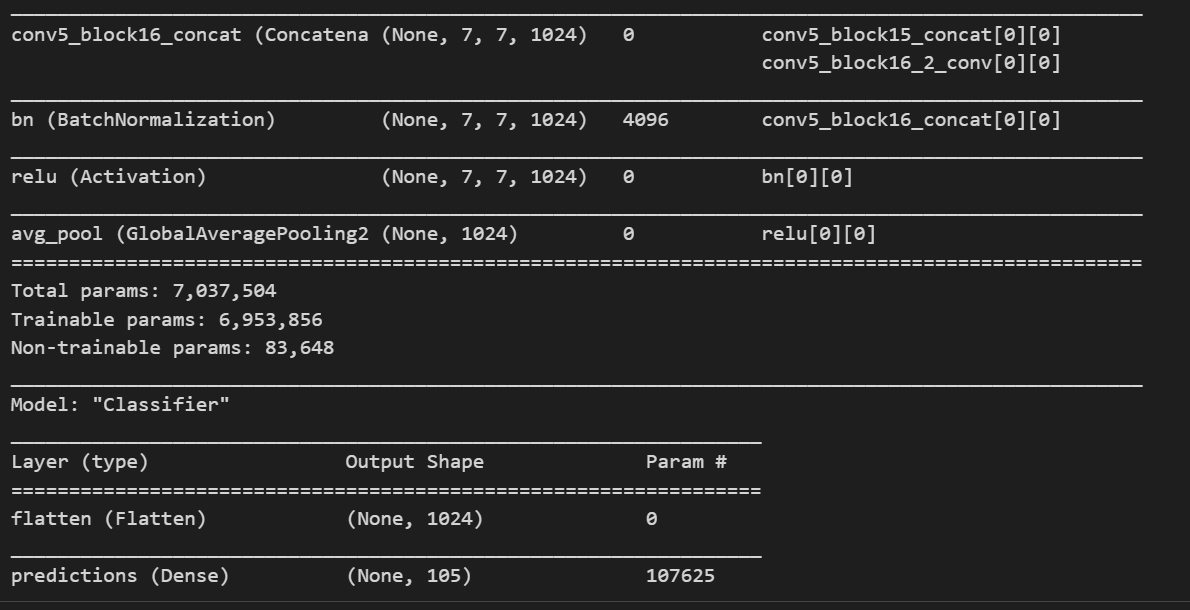
* **DenseNet 121:**

DenseNet-121 is a convolutional neural network architecture used for image classification tasks.

It is part of the DenseNet family, known for its densely connected layers. In DenseNet-121, each layer receives input from all preceding layers, promoting feature reuse and reducing the number of parameters.

This architecture enhances model efficiency and training performance, making it popular in computer vision applications.

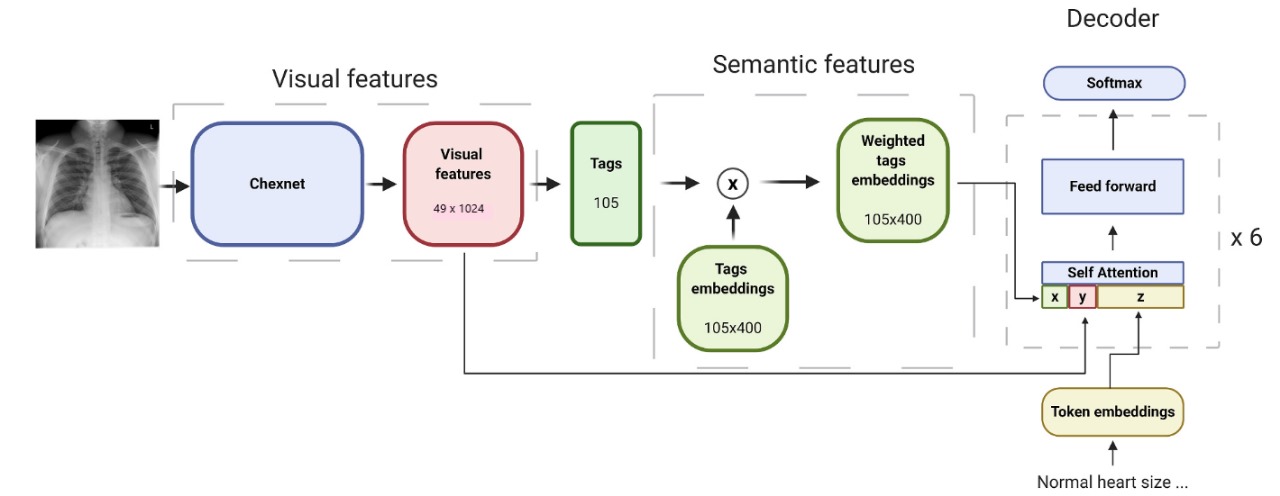




* **Conditioned Distil Generative Pretrained Transformer:**

A pre-trained distilGPT2 was used as the decoder. distilGPT2 is a compressed version of GPT2 that consists of 6 layers, a hidden layer size of 768, 12 heads, and 82 million overall parameters. distilGPT2 is twice as fast as a normal GPT2 model. The model was pre-trained on OpenWebTextCorpus,4 a reproduction of OpenAI’s Web Text dataset which was used to train the original GPT2 model.

The output layer consists of 50257 nodes representing the English language’s byte-pair encoding. The same output layer was kept as it can generate all the medical terms.



Frontend Design



* The React front-end of our MERN stack application is a critical component that delivers a responsive, modular, and user-friendly interface.
* By leveraging React's powerful features and adhering to best practices, we've created a robust and scalable front-end that integrates seamlessly with the server-side components, providing a cohesive and engaging user experience.
* The React application features user-friendly forms for user registration and login. These forms collect user credentials and communicate with the server-side API to initiate the authentication process.

**MongodB:**

Our project leverages MongoDB as the authentication backend, providing a robust and scalable solution for user management and authentication. MongoDB, a NoSQL database, is employed to store user credentials securely and manage user-related data seamlessly.

Authentication Process:

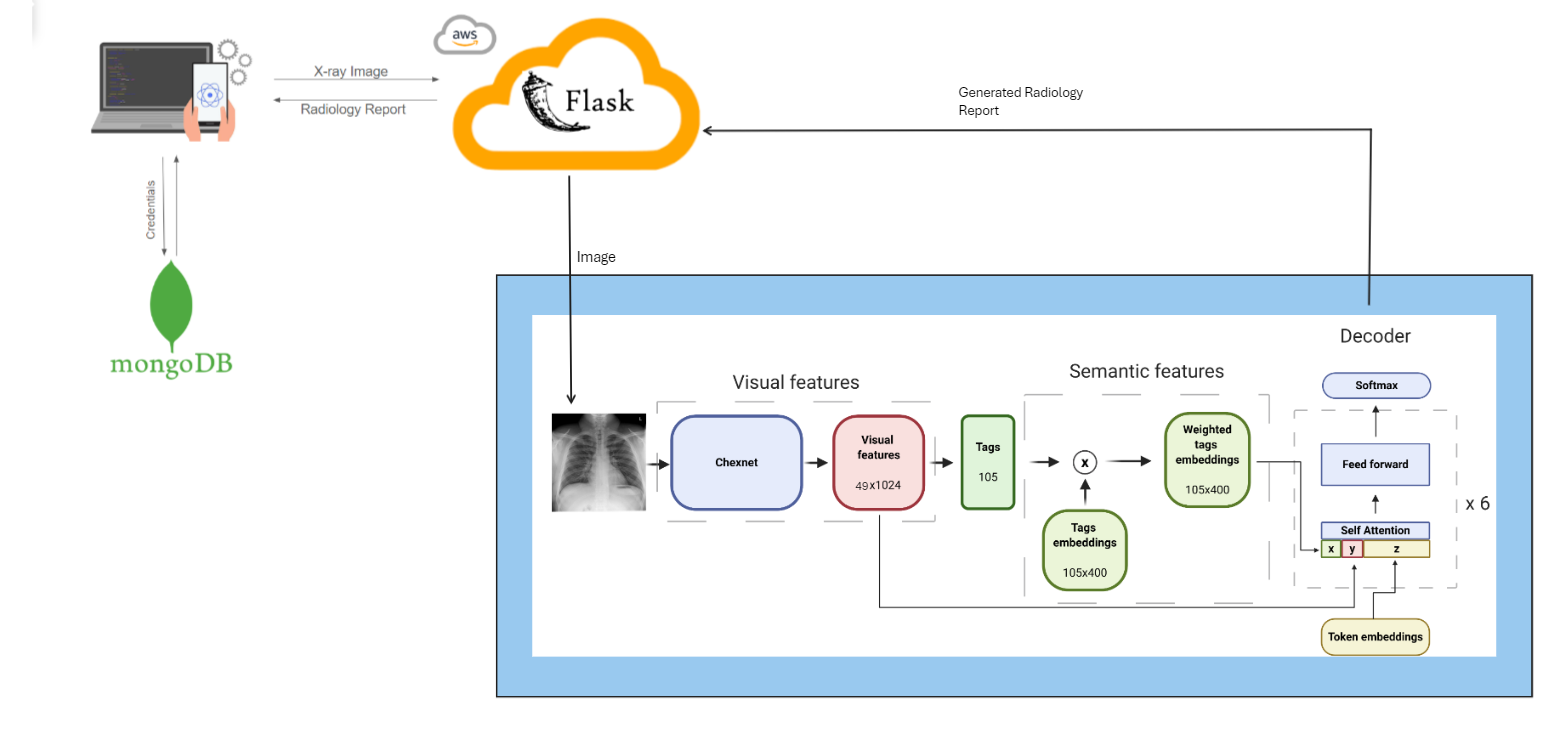
1. User Registration:

When a user registers on our platform, their authentication credentials (typically username and hashed password) are securely stored in the MongoDB database. Additional user information, such as profile details, can also be stored in MongoDB documents.

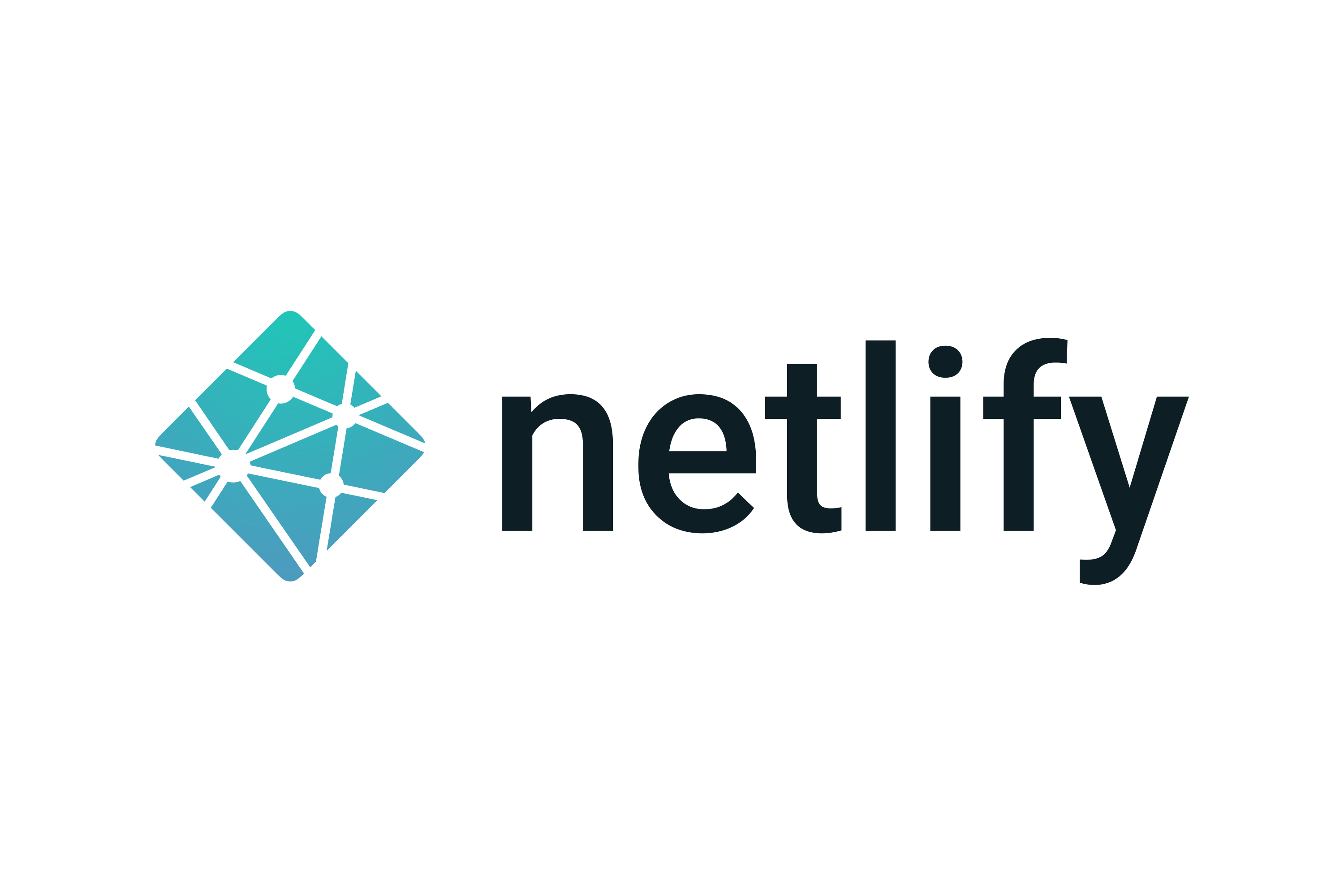
2. User Authentication:

During the login process, the user-provided credentials are verified against the stored data in MongoDB. If the provided credentials match an existing user, access is granted; otherwise, authentication fails.

Flask Framework



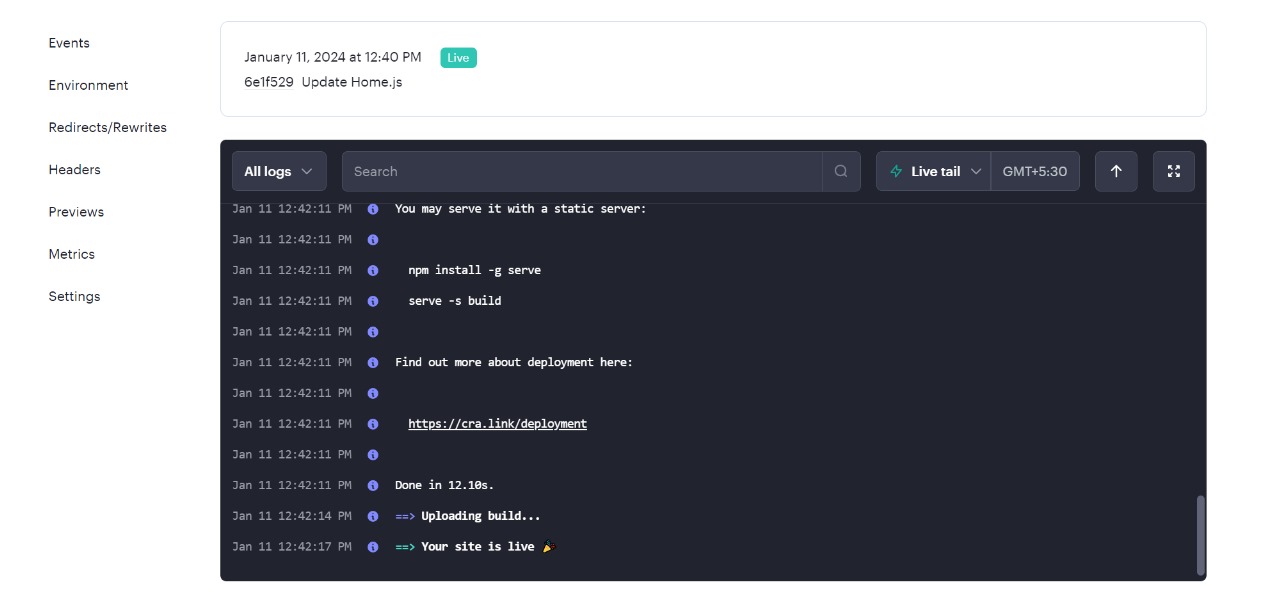
Deployment

|  |  |
| --- | --- |
| Frontend | Render/Netlify |
| Backend | Render |
| Model | AWS |

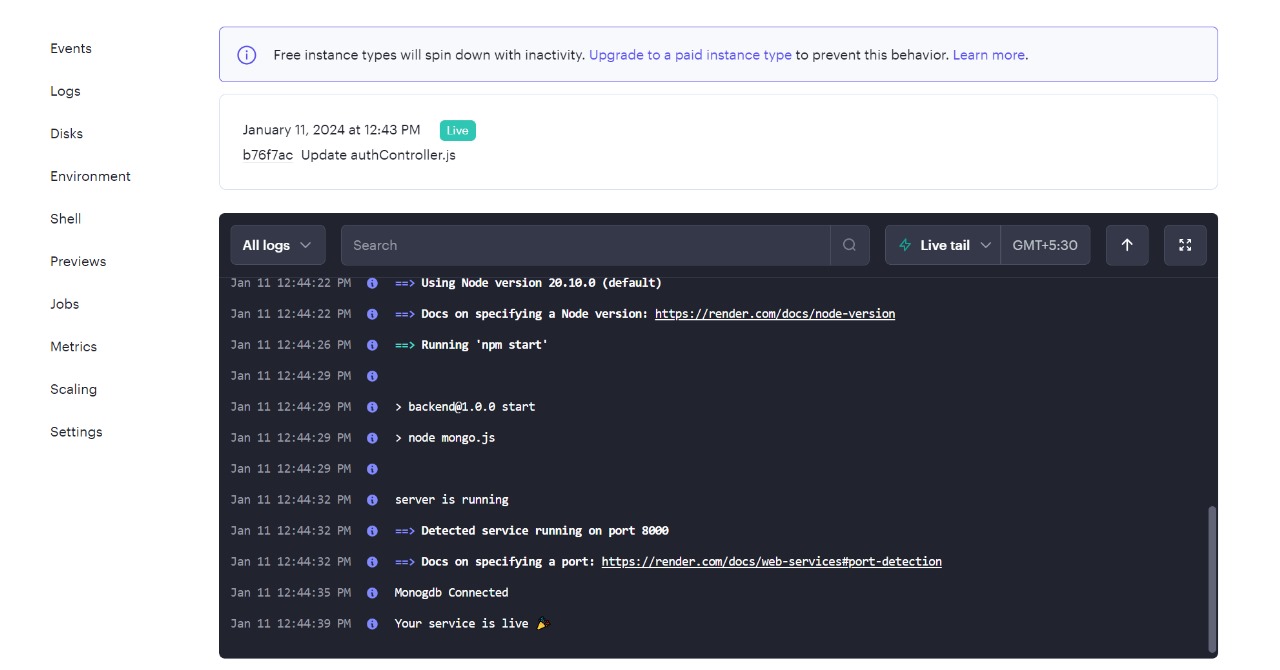
Front End Deployment:

The front end of our application is hosted on Render's web service. By leveraging Render's automatic deployment from our Git repository, updates to the front end are seamlessly applied upon each code push. The front end is accessible at <https://frontend-c1ep.onrender.com>



Back End Deployment:

Our back end services are deployed as Render services, ensuring reliability and scalability. Render automatically manages the containerization of our backend, allowing us to scale effortlessly based on demand. The back end API is accessible at <https://backend-ou2k.onrender.com>



Model Deployment:

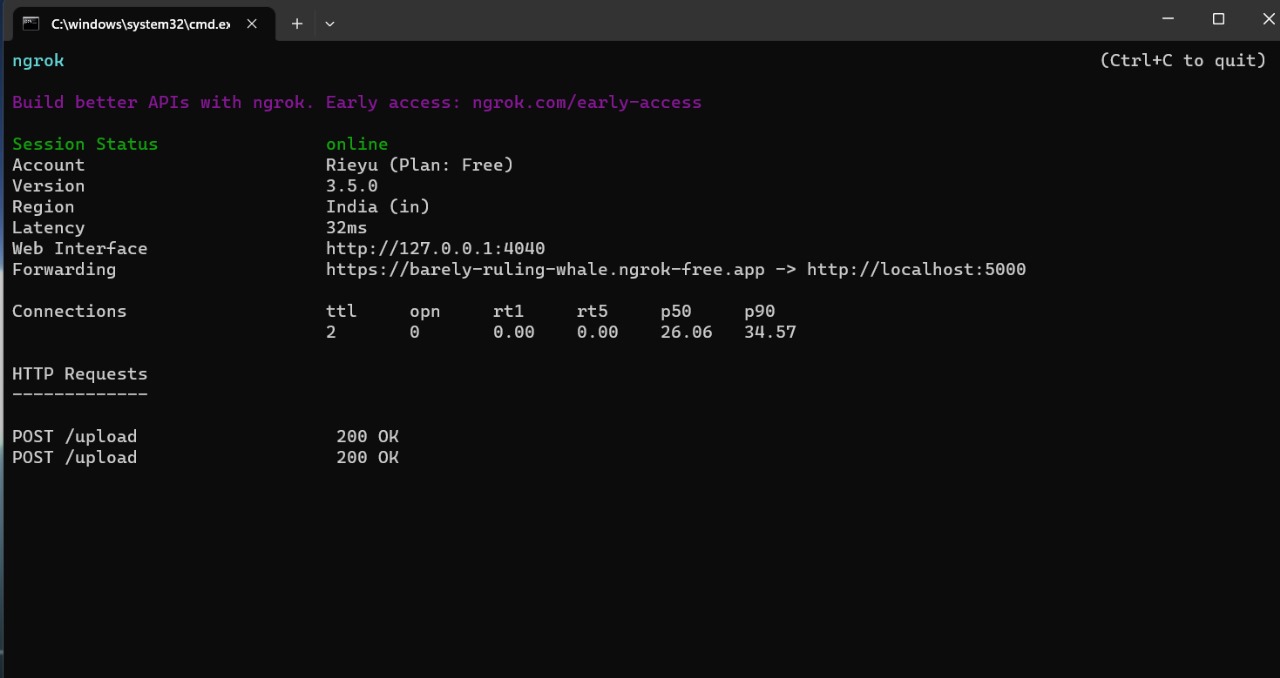
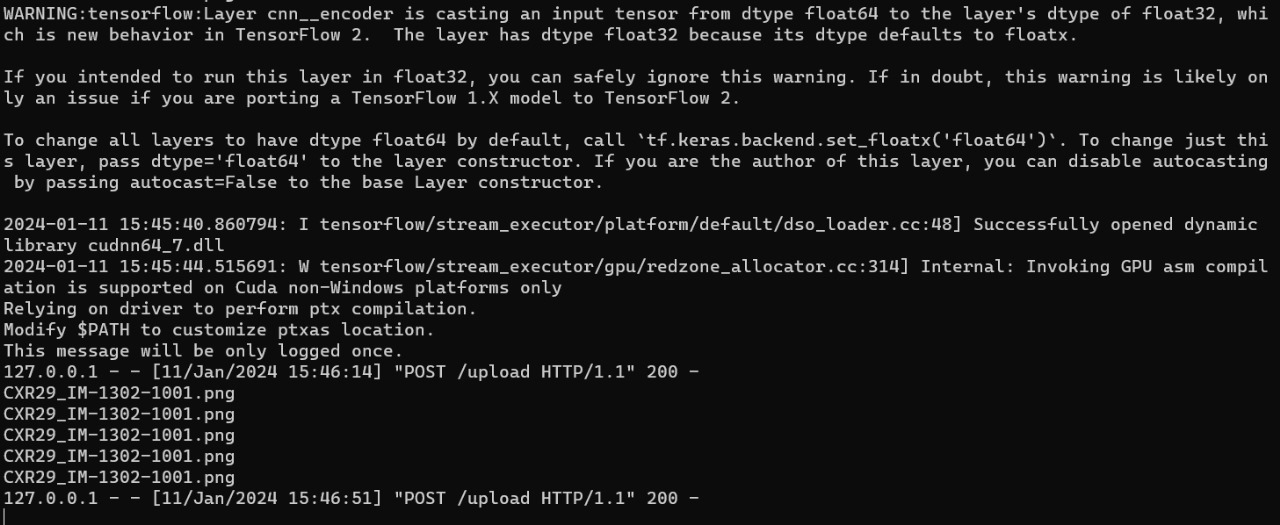
Amazon EC2 Instance:

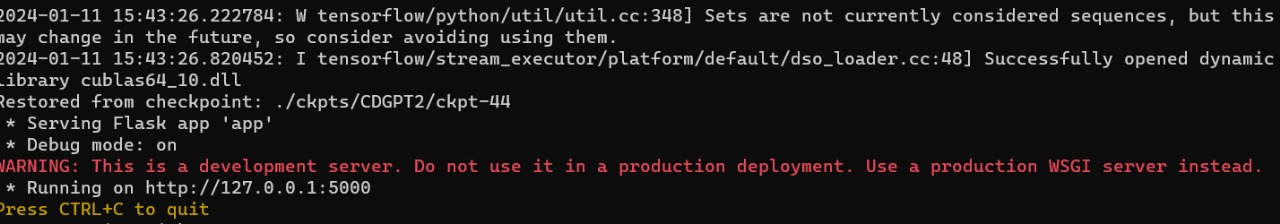
Our machine learning model is hosted on an Amazon EC2 instance, offering scalable compute capacity in the cloud. The EC2 instance provides a reliable and configurable environment for running our model.

Ngrok for Tunneling:

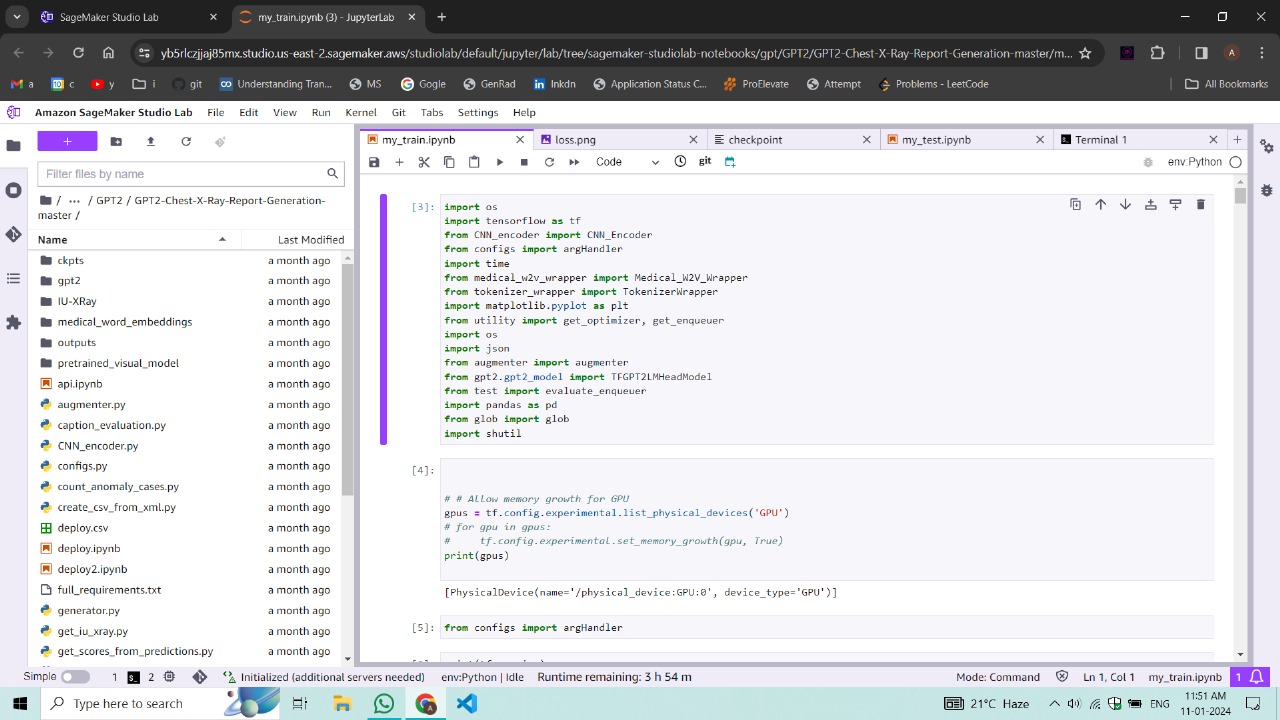
Ngrok is employed to create a secure tunnel to our EC2 instance. This allows external users and systems to interact with our machine learning model through a public URL without exposing the internal workings of our EC2 instance.

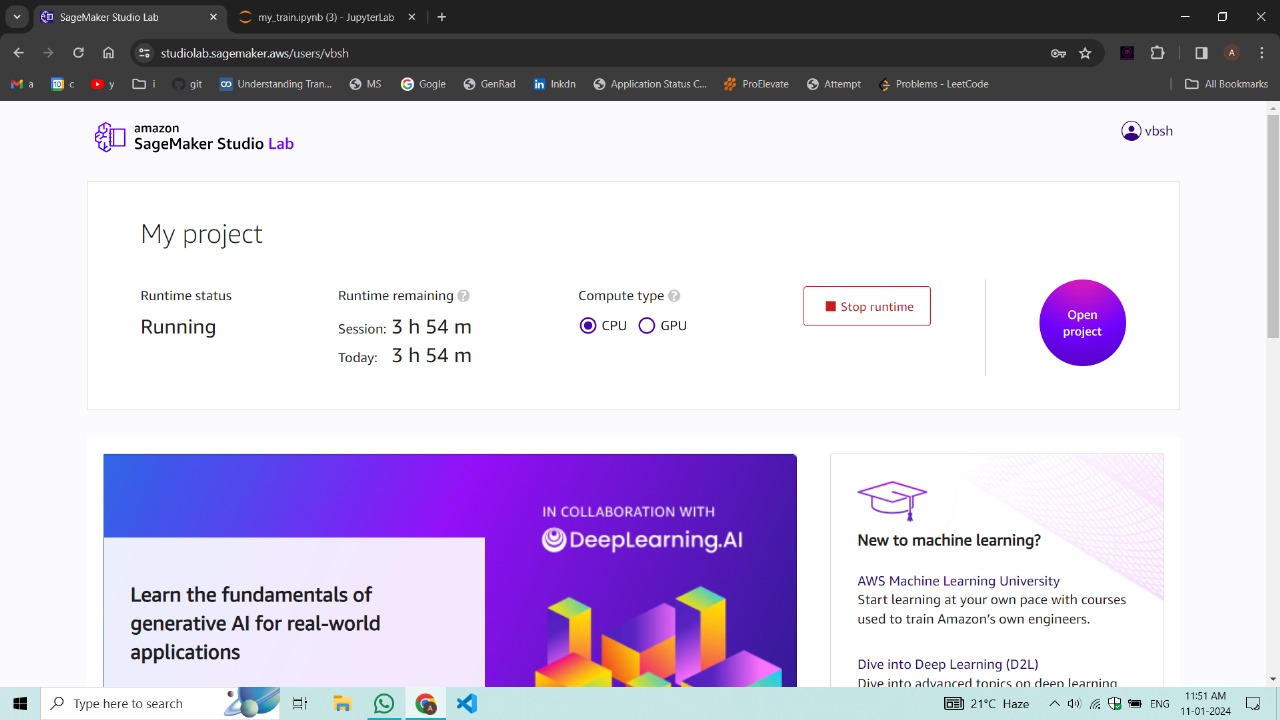
The machine learning model is accessible through the ngrok-generated URL, providing a secure and temporary public endpoint. This URL is dynamic and automatically updated by ngrok whenever the EC2 instance is restarted.

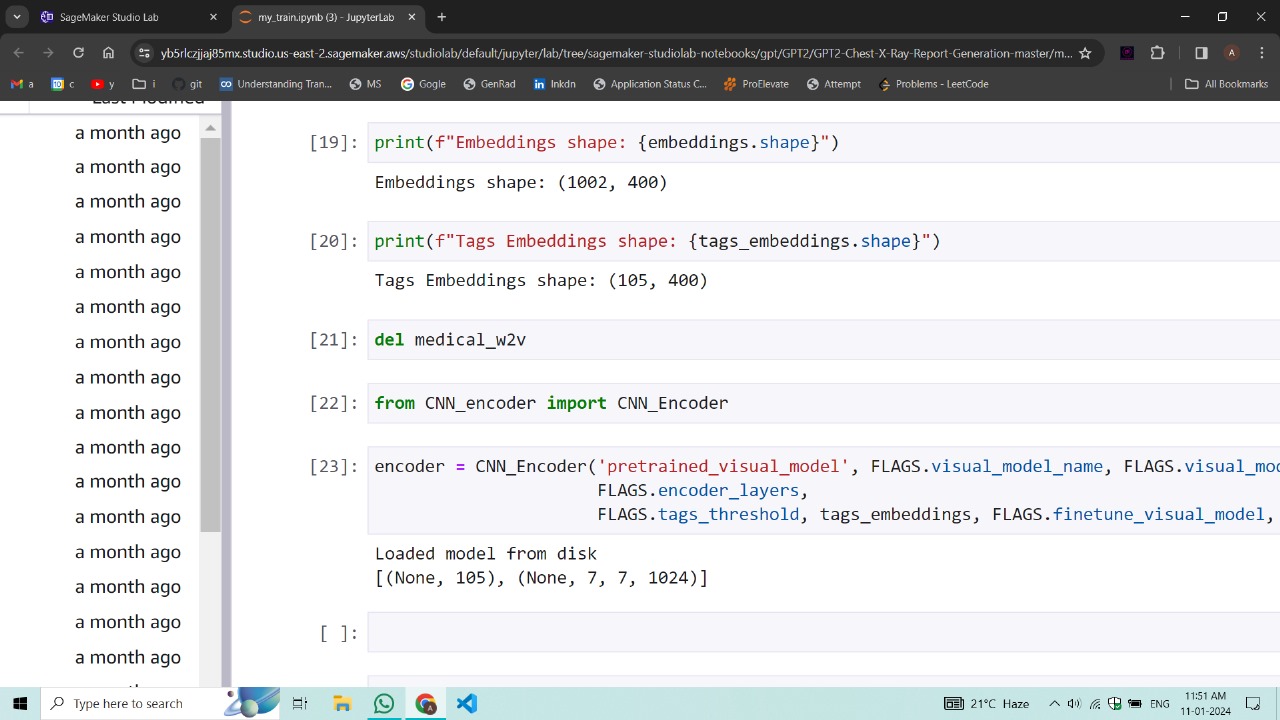
  
  




AWS Sage maker Training

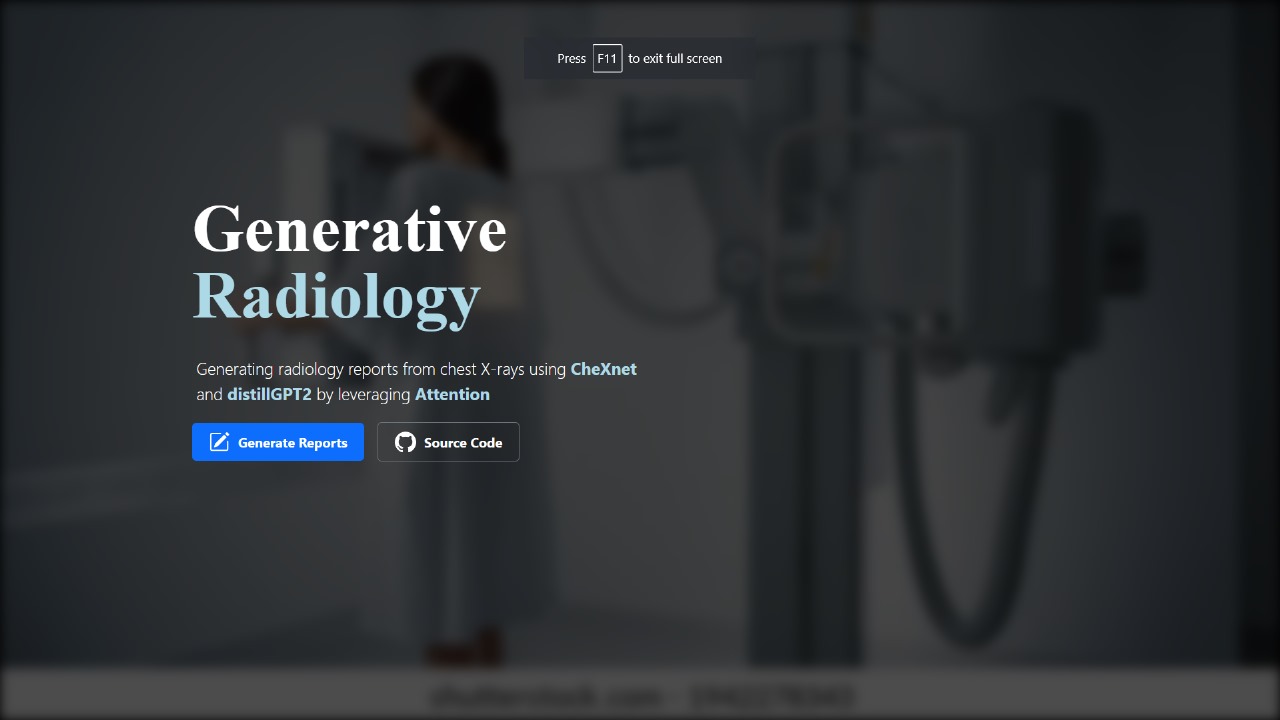




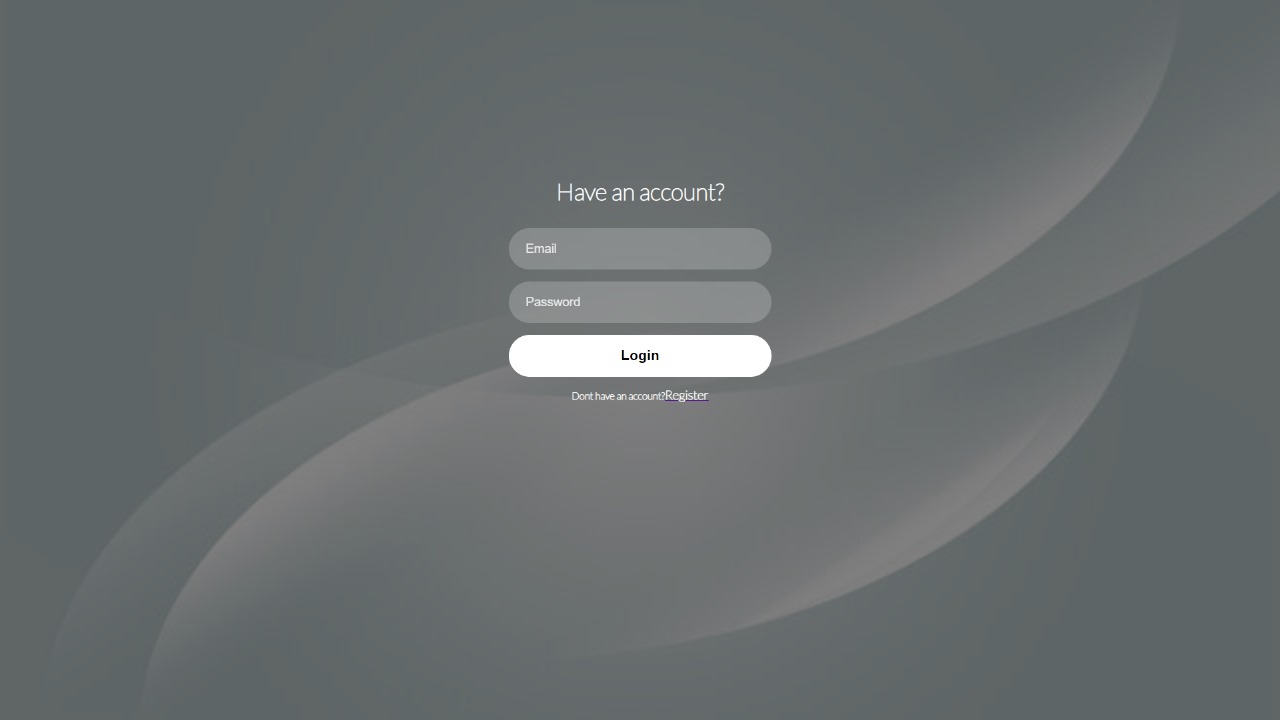


Screenshots

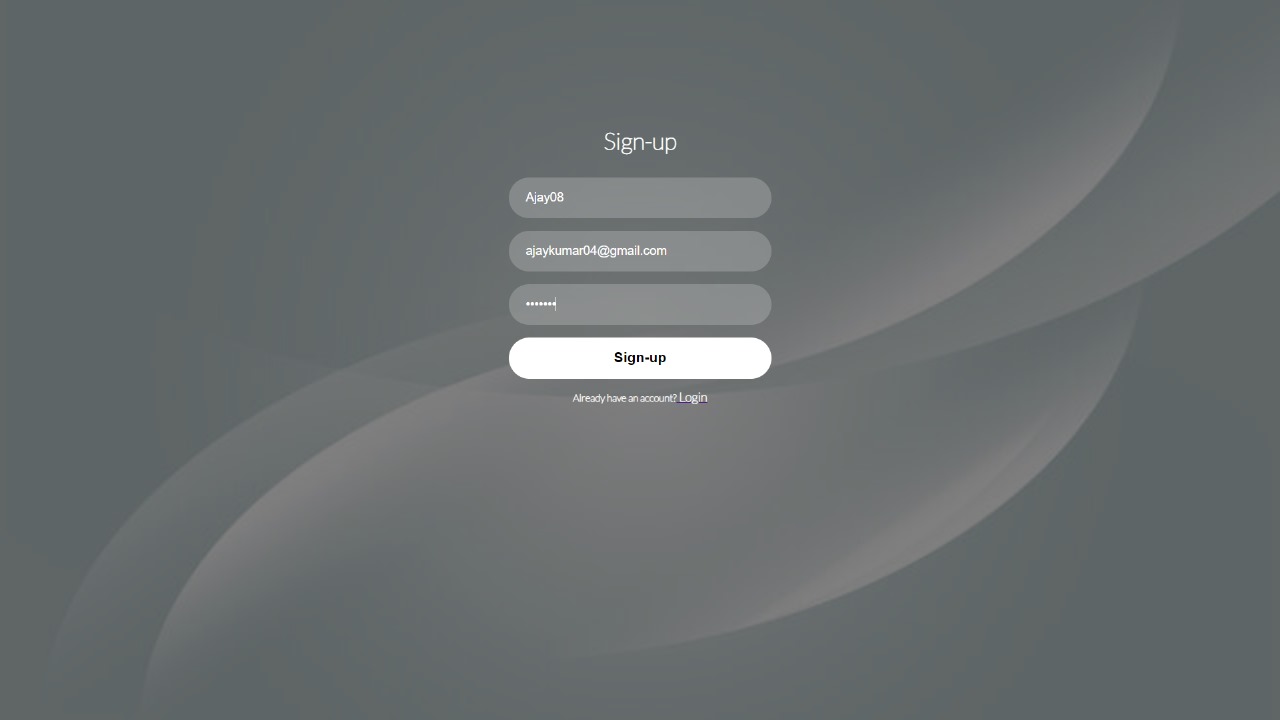
Walkthorugh of Website

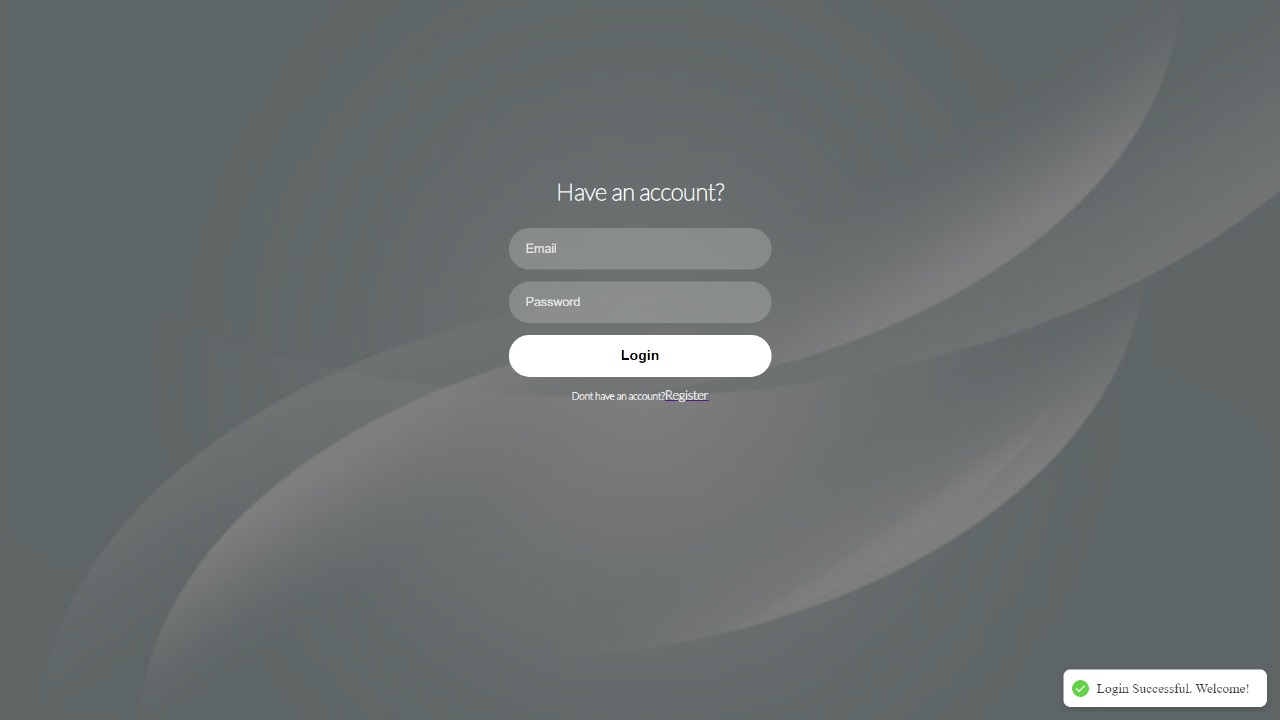


**Login**

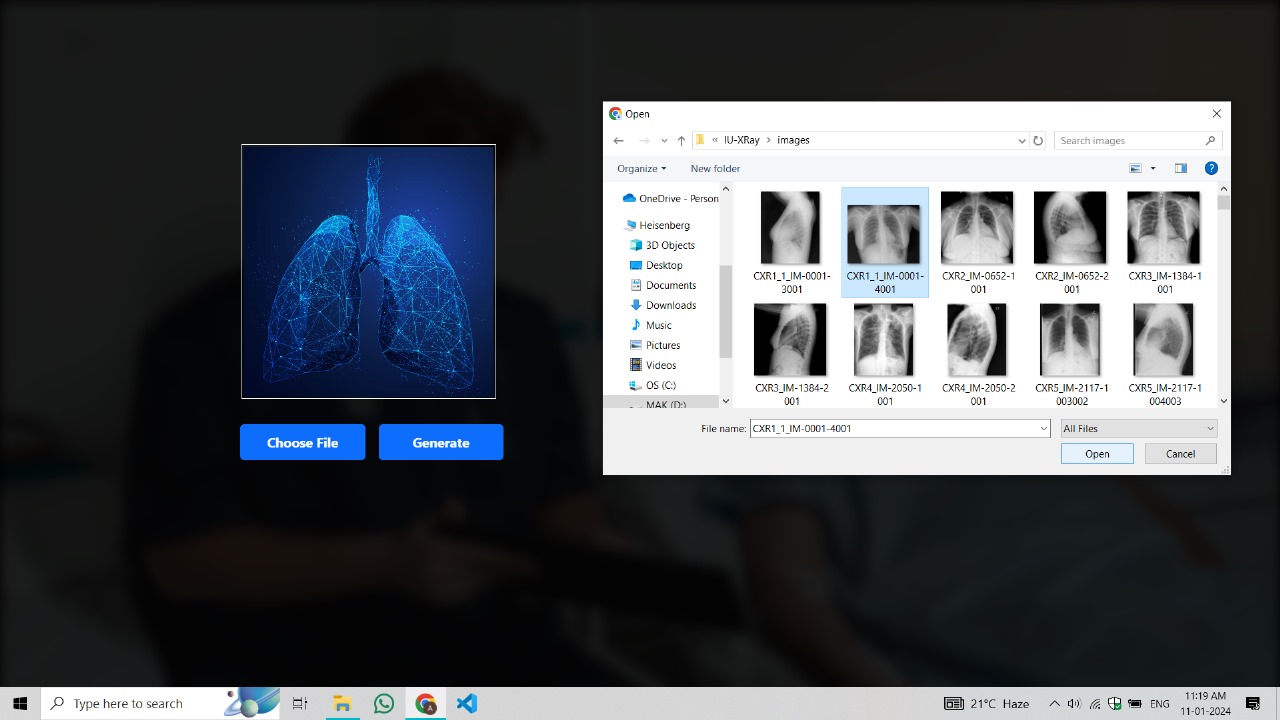


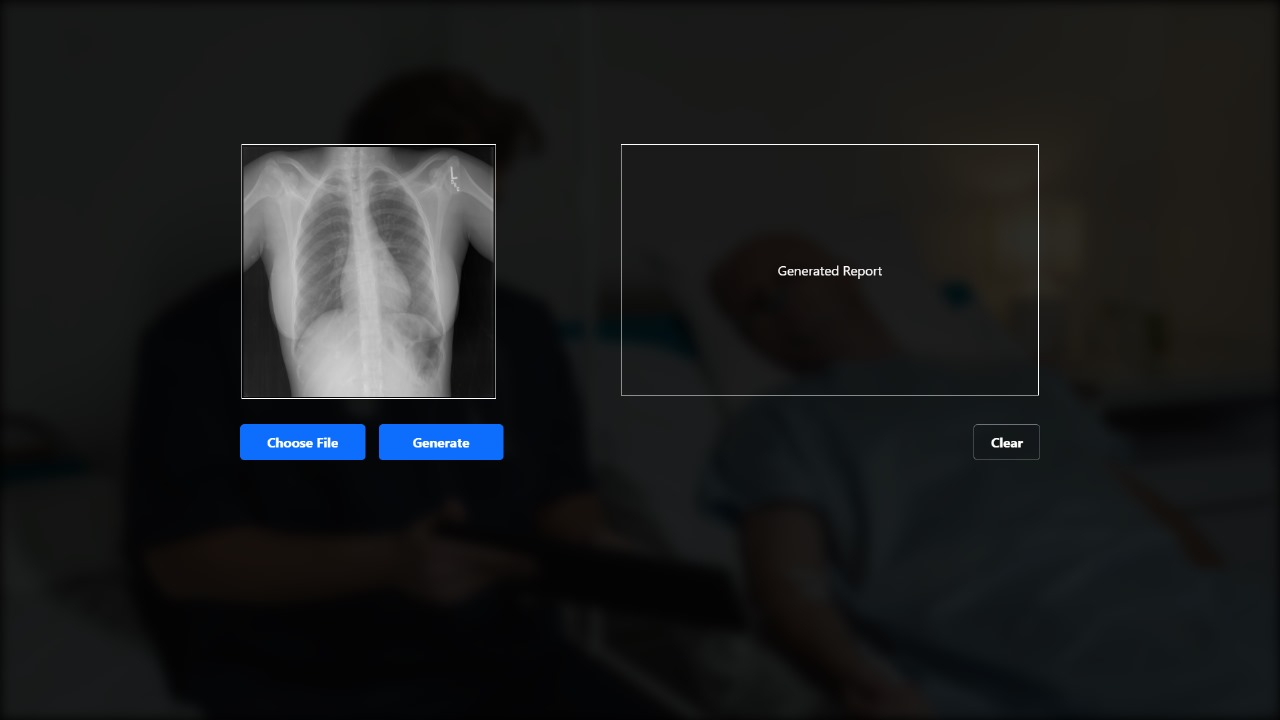
**Signup**



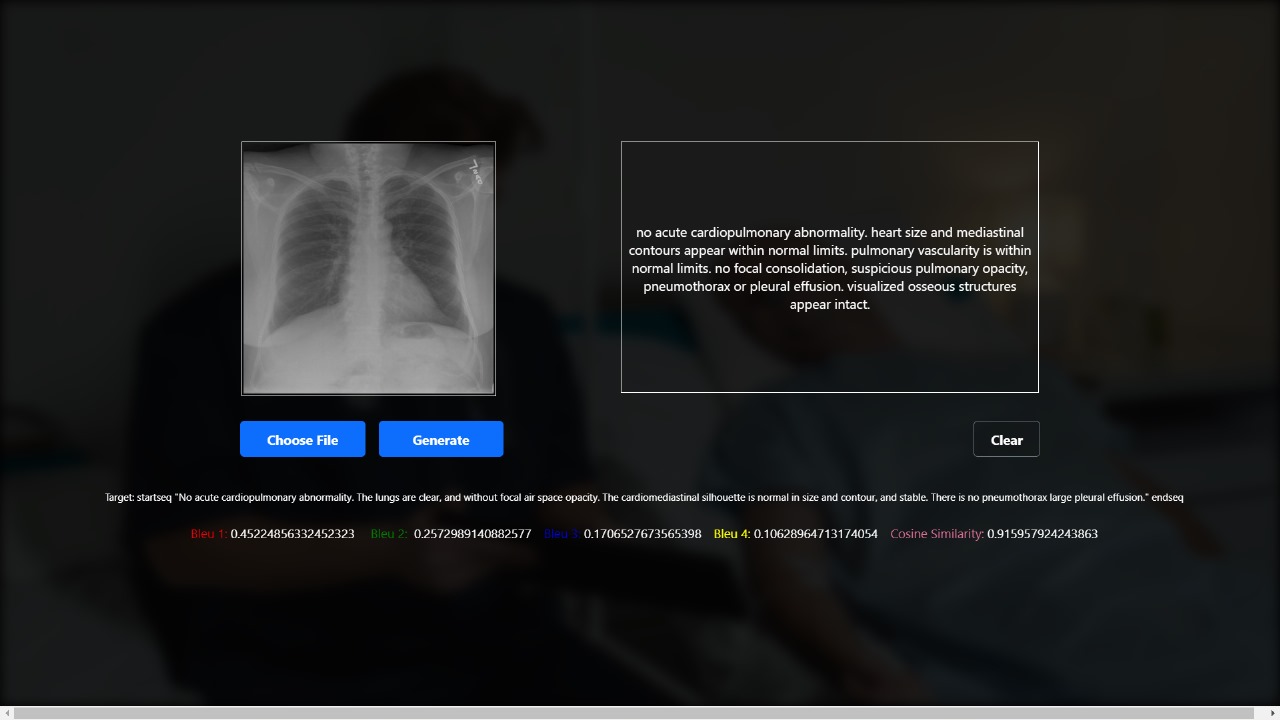


**Selecting an Image**





**OUTPUT:**



References

I. Source Paper:

<https://www.sciencedirect.com/science/article/pii/S2352914821000472>

II. Chexnet Based on Densenet121:

<https://arxiv.org/abs/1711.05225>

III. Understanding Transformer Article

<https://medium.com/@fareedkhandev/understanding-transformers-a-step-by-step-math-example-part-1-a7809015150a>

IV. Medical Words from PubMed

<https://pubmed.ncbi.nlm.nih.gov/>

V. Indiana University Dataset for training and testing

<https://www.kaggle.com/datasets/raddar/chest-xrays-indiana-university>

VI. Video References for Embeddings, Transformers

<https://www.youtube.com/watch?v=sZGuyTLjsco&pp=ygUad29yZCBlbWJlZGRpbmcgdHJhbnNmb3JtZXI%3D>

1. <https://www.youtube.com/watch?v=bQ5BoolX9Ag&t=216s>

VII. Source Code [https://Github.com/omar-mohamed/GPT2-Chest-X-](https://Github.com/omar-mohamed/GPT2-Chest-X-Ray-Report-Generation)

[Ray-Report-Generation.](https://Github.com/omar-mohamed/GPT2-Chest-X-Ray-Report-Generation)