**BATCH-16**

**SMART MEDICAL DIAGNOSIS ASSISTANT**

**Abstract:**

This project explores the development of an AI-based smart medical diagnosis system using TensorFlow, aimed at supporting healthcare professionals and patients with preliminary diagnostic insights. The system utilizes machine learning algorithms, specifically neural networks, trained on publicly available medical datasets to predict possible diseases or health conditions based on input symptoms or diagnostic images.

**Objective:**

To develop an intelligent system that can diagnose diseases based on symptoms, medical images, or health data using deep learning models in TensorFlow**.**

**Introduction:**

Healthcare is one of the most data-intensive and critical sectors. With the rise in global population and disease complexity, quick and reliable diagnosis is essential. AI-driven tools can significantly reduce the workload of medical practitioners and help in early detection. This project proposes a system using TensorFlow to perform medical diagnosis tasks either through symptom analysis or medical image classification.

**Problem Statement:**

Many people face delays or mistakes in getting medical diagnoses due to lack of doctors or time. This project aims to build an AI-based tool using TensorFlow that helps predict diseases by analyzing symptoms or medical images, making early diagnosis easier and faster for everyone.

**Key Features:**

Symptom-based disease prediction (text input)

- Medical image diagnosis (X-ray, CT, etc.)

- Chatbot-based interaction for diagnosis

- Real-time diagnosis suggestions

- Recommendations for specialist consultation

**Technologies used:**

- TensorFlow & Keras

- Python

- Flask/Streamlit for Web UI

- Google Colab/Jupyter for model training

**Future Scope:**

Integration with EHR (Electronic Health Records).

- Multi-language support.

- Real-time diagnosis via wearable devices.

### UML Diagram Architecture:

### @startuml

### skinparam componentStyle rectangle

### node "User Device" {

### [Web UI]

### }node "Web Server" {

### [Flask Backend]

### [TensorFlow Model]

### [Explainable AI Module]

### }node "Data Storage" {

### [Medical Image Datasets]

### }[Web UI] --> [Flask Backend]

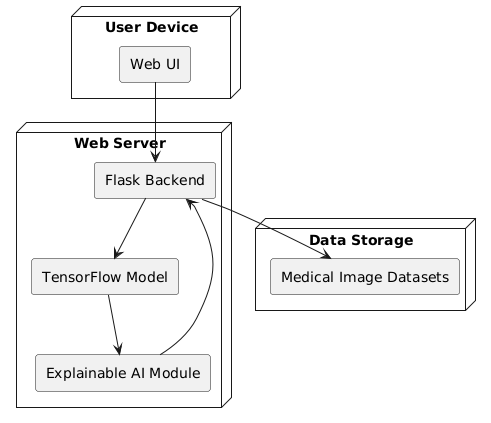
### [Flask Backend] --> [TensorFlow Model]

### [TensorFlow Model] --> [Explainable AI Module]

### [Explainable AI Module] --> [Flask Backend]

### [Flask Backend] --> [Medical Image Datasets]

### @enduml



**UML Diagram Types:**

### Use Case Diagram

### @startuml

### actor Doctor

### actor Patient

### rectangle "Smart Medical Diagnosis System" {

### Doctor --> (Upload Medical Image)

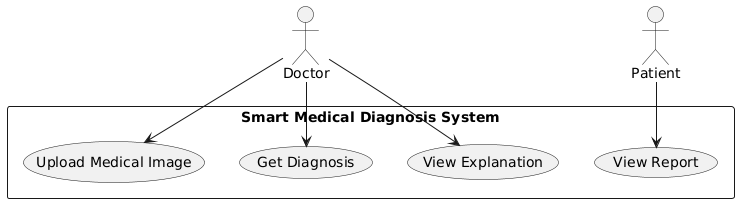
### Doctor --> (Get Diagnosis)

### Doctor --> (View Explanation)

### Patient --> (View Report)

### }

### @enduml



### Activity Diagram:

### @startuml

### start

### :Upload Image;

### :Preprocess Image;

### :Run AI Model;

### :Predict Disease;

### :Generate Grad-CAM Heatmap;

### :Show Results and Explanation;

### stop

### @enduml

### C:\Users\ANIL\Desktop\3333333333.png

### Class Diagram

@startuml

class MedicalImage {

- image\_id : string

- image\_path : string

- image\_type : string

+ preprocess()

+ display()

}

class DiagnosisModel {

- model\_name : string

- confidence\_score : float

+ load\_model()

+ predict()

}

class GradCAMExplainer {

- heatmap : array

+ generate\_map()

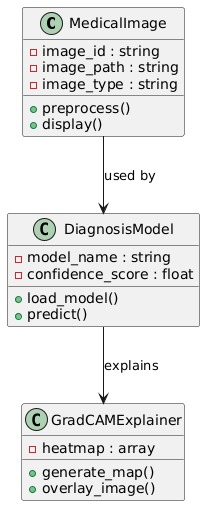
+ overlay\_image()

}

MedicalImage --> DiagnosisModel : used by

DiagnosisModel --> GradCAMExplainer : explains

@enduml



### Sequence Diagram

@startuml

class MedicalImage {

- image\_id : string

- image\_path : string

- image\_type : string

+ preprocess()

+ display()

}

class DiagnosisModel {

- model\_name : string

- confidence\_score : float

+ load\_model()

+ predict()

}

class GradCAMExplainer {

- heatmap : array

+ generate\_map()

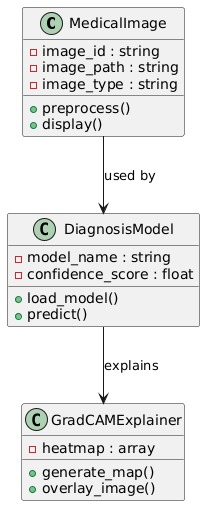
+ overlay\_image()

}

MedicalImage --> DiagnosisModel : used by

DiagnosisModel --> GradCAMExplainer : explains

@enduml



### Component Diagram

@startuml

package "Frontend" {

[User Interface]

}

package "Backend" {

[Flask Server]

[Diagnosis Engine]

[GradCAM Module]

}

package "Data" {

[Medical Image Dataset]

}

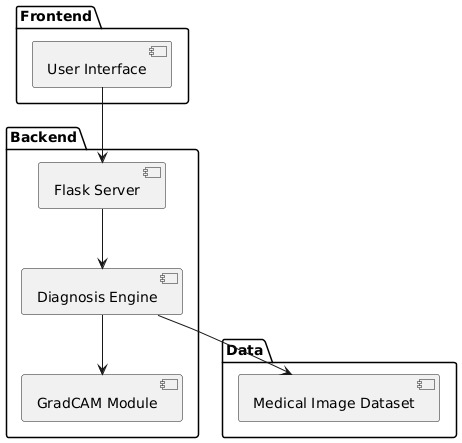
[User Interface] --> [Flask Server]

[Flask Server] --> [Diagnosis Engine]

[Diagnosis Engine] --> [GradCAM Module]

[Diagnosis Engine] --> [Medical Image Dataset]

@enduml



### Deployment Diagram

@startuml

node WebApp {

component UI

component FlaskServer

}

node Server {

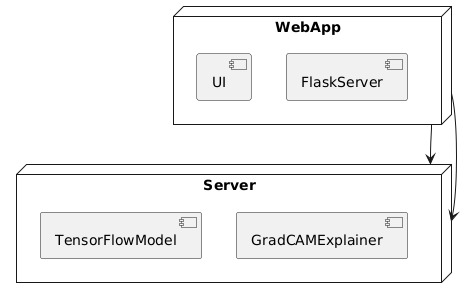
component TensorFlowModel

component GradCAMExplainer

}

WebApp --> Server

@enduml



### Object Diagram

@startuml

object img1 {

image\_id = "001"

image\_type = "X-ray"

}

object model {

model\_name = "ResNet50"

confidence\_score = 0.94

}

object explainer {

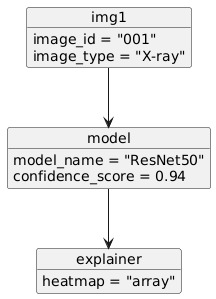
heatmap = "array"

}

img1 --> model

model --> explainer

@enduml



### State Diagram

@startuml

[\*] --> Idle

Idle --> Loading : image uploaded

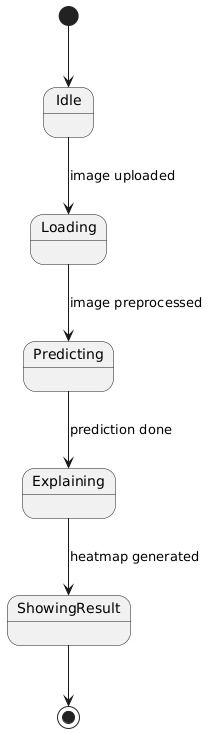
Loading --> Predicting : image preprocessed

Predicting --> Explaining : prediction done

Explaining --> ShowingResult : heatmap generated

ShowingResult --> [\*]

@enduml



### Communication Diagram

@startuml

object UI

object FlaskServer

object Model

object GradCAM

UI -> FlaskServer : sends image

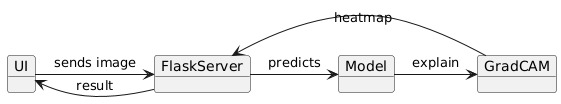
FlaskServer -> Model : predicts

Model -> GradCAM : explain

GradCAM -> FlaskServer : heatmap

FlaskServer -> UI : result

@enduml



**Interaction Overview Diagram**

@startuml

start

:User uploads image;

partition UI {

:Show upload form;

}

partition Backend {

:Preprocess image;

:Run prediction;

:Generate GradCAM;

}

stop

@enduml@startuml

start

:User uploads image;

partition UI {

:Show upload form;

}

partition Backend {

:Preprocess image;

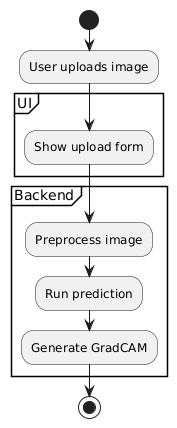
:Run prediction;

:Generate GradCAM;

}

stop

@enduml



### Timing Diagram (simplified)

@startuml

title Timing Diagram for Diagnosis Process

robust "UI" as UI

robust "Backend" as Backend

robust "Model" as Model

UI is idle

Backend is idle

Model is idle

@ 1000

UI is uploading

@ 2000

Backend is preprocessing

@ 3000

Model is predicting

@ 4000

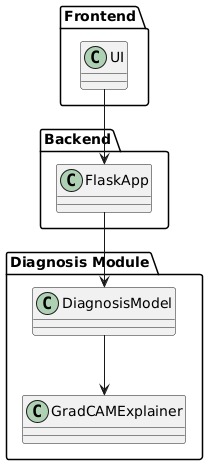
Backend is generating explanation

@ 5000

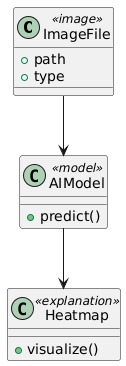
UI is displaying result

@enduml

### Package Diagram



### Profile Diagram (custom tags for modeling)



### Composite Structure Diagram

