Project Overview: Kidney Stone Detection Using IR

Kidney stones are a common health issue that affects millions of people worldwide. The traditional method of detecting kidney stones is through radiography, which is time-consuming and requires the use of ionizing radiation. However, with the advent of image recognition technology, it is now possible to detect kidney stones using non-invasive methods. Image recognition technology uses machine learning algorithms to analyze medical images of the kidneys and urinary tract to detect the presence of stones in the kidneys. A study conducted by Muhammed Talo et al. proposed an automated detection of kidney stones using coronal computed tomography (CT) images with deep learning (DL) technique, which showed an accuracy of 96.82% in detecting kidney stones

White box testing is a software testing technique that examines the internal workings of an application or system. In the context of kidney stone detection using image recognition, white box testing can be used to test the accuracy and reliability of the machine learning algorithms used in image recognition technology. The testing process involves examining the source code and internal structure of the application to identify any potential vulnerabilities or defects that could affect its performance. By conducting white box testing, it is possible to ensure that the application is functioning as intended and that it is capable of accurately detecting kidney stones

Objectives: The objectives of this White Box Testing are:

1. To Verify the accuracy of image detection
2. To Ensure error-free handling of functions
3. To validate the results outputted by the program

Testing Environment:

→ Windows 11 / Debian
→ Python 3.11 (with Venv) or iPython (with Jupyter / Colab)
→ Testing Framework: Unittest

Test Cases & Results:

Test case 1: Image without Kidney Stone

→ Input: An image of a Kidney scan without a stone
→ Expected Output: Kidney Stone is Absent
→ Result: Test Passed Successfully

Test case 2: Image with Kidney Stone

- → Input: An image of a Kidney scan with a stone
- → Expected Output: Kidney Stone is Present
- → Result: Test Passed Successfully

Test case 3: Image with Kidney Stone (slightly faded)

- → Input: An image of a Kidney scan with a stone
- → Expected Output: Kidney Stone is Present
- → Result: Test Passed Successfully (Fails sometimes if resolution is too low)

Conclusion : In the case of "Kidney Stone Detection using Image recognition", we have used white box testing to test the internal structure of the software and ensure that it is working as expected. The software has been tested using various techniques and results of these tests indicate that the software is functioning correctly and meets all the requirements specified in the project. The software has been designed to detect kidney stones using image recognition, and it does so with a high degree of accuracy.

White-Box Testing:

```
import unittest
import cv2
from tensorflow.keras.models import load_model

class TestKidneyStoneModel(unittest.TestCase):
    def test_predict_kidney_stones(self):
        # Load the model
        model = load_model('path_to_your_model.h5')

        # Load an image file for testing
        test_image = cv2.imread('test_image.jpg')

        # Preprocess your image here in the same way as you did when training the model
        # test_image = preprocess(test_image)

        # Reshape or expand dimensions if needed, for example if your model expects a 4D input (batch_size, height, width, channels)
        # test_image = np.expand_dims(test_image, axis=0)

        # Call the function with the test image
        prediction = model.predict(test_image)

        # Postprocess your prediction here if needed, for example taking the argmax for a multi-class classification problem
        # prediction = np.argmax(prediction)

        # Check the type of the prediction result
        self.assertIsInstance(prediction, str)

        # Check that the prediction result is one of the expected classes
        self.assertIn(prediction, ['Positive', 'Negative'])

if __name__ == '__main__':
    unittest.main()
```