**Team Alt + F4 Solution Outline**

* Training images and image annotations are used to train a custom Ultralytics YOLOv8 model to detect the bounding boxes around the potholes and measuring sticks.
* Using the above model, it identifies the respective bounding boxes. The image is then cropped into two separate images, one resembling the bounding box of the pothole and another the bounding box of the detected side of the measuring stick. If both sides of the measuring stick are detected, the left side is used.
* Using a custom-defined set of rules using OpenCV to detect red colours, a mask is added to the image of the measuring stick. This masked image is then converted to a grayscale bitmap image, with the red parts of the image being white pixels. This image is then processed again to remove noise and only include parts of the measuring stick.
* OpenCV contours are then used to determine the outlines of the red parts on the measuring stick.
* Note that the width of the measuring stick has been determined as **INSERT WIDTH** by the ratio of length to width of the measuring stick.
* The length in pixels of one of the shorter contours of the red parts of the stick is then determined (i.e. finding the width of the measuring stick in pixels).
* This pixel value is then squared and the square of the actual width is divided by it. This determines the ratio of area in pixels to area in square millimetres.
* Once this is determined, the area of the actual pothole(in pixels) is determined using OpenCV edge detection. This area is then converted to square millimetres by multiplying it with the ratio mentioned above.
* The area of the input pothole is then fed into a neural network **IS THIS CORRECT** that was trained using the sklearn MLPRegressor module on a modified version of the training data provided, finally resulting in a prediction.
* This model is brought together visually using a react webpage linked to a node.js server. All the running of the python scripts used to predict the amount of tar needed for the pothole is also orchestrated by the node.js server.