# Image Detection Website

This system design document outlines the architecture and design of the AI-based web app which uses the Yolov3 AI model. The app provides users with the ability to analyse and classify images using machine learning algorithms. The system is built using a client-server architecture, where the frontend is responsible for user interaction and the backend handles image processing and analysis.

I have used the YOLOv3 model in the assignment. Yolov3 is a relatively large and complex model that requires significant computational resources to run so I have allocated the AI functionality on the backend. Additionally, running the model on the frontend may increase the latency of the application, as the model needs to be downloaded and initialized before it can be used.

Running the AI model on the backend can provide several benefits, such as:

1. Scalability
2. Security
3. Performance

## System design

### Architecture Overview

The web app consists of three main components: the frontend, the backend, and the machine learning model.

### Frontend

The front end has been built using HTML and CSS. The frontend provides users with a user-friendly interface to upload images, view analysis results, and interact with the app. The front end communicates with the backend using REST APIs.

### Backend

The backend is responsible for image processing and analysis. It has been built using Python and Flask web frameworks. The backend is responsible for handling image uploads, pre-processing, and feeding images to the machine-learning model for analysis.

### Machine Learning Model

The machine learning model is responsible for image classification and analysis. The model has been built using the darknet framework. The model is trained on a large dataset of labeled images to accurately classify new images uploaded by users. The trained model is deployed on the backend server for inference.

**AI Functionality Allocation**

The AI functionality is allocated on the backend.

The backend handles the following AI functionality:

**Image validation:** The backend validates images uploaded by users to ensure they are in the correct format and within size limits.

**Machine learning model inference:** The backend uploads the trained machine learning model and uses it to analyse images uploaded by users. The backend returns the analysis results to the front end for display.

## About the AI Model YOLOv3

YOLOv3 is built using a deep learning framework called Darknet, which is developed in C and CUDA. Darknet includes a number of libraries and tools that are used to build and train the neural network, including:

**CUDA and cuDNN**: These libraries are used to accelerate the computation of the neural network on NVIDIA GPUs.

**OpenCV**: This is an open-source computer vision library that is used for image processing and manipulation, including reading and writing image files, resizing and cropping images, and converting between color spaces.

**OpenSSL**: This is a cryptographic library that is used for secure communication between the client and server.

**POSIX threads**: These are used for multithreading, which allows the neural network to process multiple images simultaneously.

**BLAS (Basic Linear Algebra Subprograms):** These are a set of low-level linear algebra routines that are used for matrix multiplication, vector addition, and other operations.

In addition to these libraries, YOLOv3 also uses several Python libraries for data pre-processing and analysis, including NumPy, Pandas, and Matplotlib. These libraries are used to manipulate and visualize the input data before feeding it into the neural network.

## UI/UX Design

### Introduction

The app is designed to provide a user-friendly experience for users to upload and analyze images for object detection.

### Design Goals

The design goals of the web app are:

- To provide a clean and modern interface that is easy to navigate.
- To highlight the YOLOv3 AI model's object detection capabilities in an engaging way.
- To provide a seamless user experience from uploading an image to viewing the object detection results.
- To be mobile responsive and accessible on all devices.

### Design Guidelines

The design of the web app follows the following guidelines:

- Easy navigation: The navigation is easy to use, with clear instructions for uploading an image and viewing the object detection results.
- High-quality object detection results: The object detection results will be accurate and visually appealing, with clear outlines around the detected objects and labels for each object.
- User-friendly image upload process: The image upload process is easy to use, with clear instructions for selecting and uploading an image.

### Design Elements

The design of the web app includes the following elements:

### Upload Image Page

The upload image page allows users to select an image file from their computer. Once the image is uploaded, the user will click ‘Upload!’ to start the object detection process and the user will be redirected to another page.

**Object Detection Results Page**

The object detection results page will display an ‘Open Image’ button to open the uploaded image with clear outlines around the detected objects and labels for each object. Users will be able to zoom in and out of the image to view the object detection results in more detail. To get to the upload image page again user can click on back arrow.