A Project Report

On

Real Time Content Moderation

Submitted in partial fulfilment of the requirement of

University of Mumbai

For the Degree of

Bachelor of Engineering

in

COMPUTER ENGINEERING

Submitted by

Cyrus Britto
Anuj Dalvi
Animesh Srivastava

Supervised by

Ms Kavita Shelke

&

Mr Harshit Rai

Department of Computer Engineering

Fr. Conceicao Rodrigues Institute of Technology
Sector 9A, Vashi, Navi Mumbai - 400703

UNIVERSITY OF MUMBAI

2019-2020

APPROVAL SHEET

This is to certify that the project entitled

"Real Time Content Moderation"

Submitted by

Cyrus Britto	**101607**
Anuj Dalvi	**101610**
Animesh Srivastava	**101655**

Supervisors : ____________________

Project Coordinator : ____________________

Examiners : 1. ____________________

2. ____________________

Head of Department : ____________________

Date :

Place :

Declaration

We declare that this written submission for B.E. Declaration entitled "**Real Time Content Moderation**" represent our ideas in our own words and where others' ideas or words have been included. We have adequately cited and referenced the original sources. We also declared that we have adhere to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any ideas / data / fact / source in our submission. We understand that any violation of the above will cause for disciplinary action by institute and also evoke penal action from the sources which have thus not been properly cited or from whom paper permission have not been taken when needed.

Project Group Members:

1. Cyrus Britto, 101607

2. Anuj Dalvi, 101610

3. Animesh Srivastava, 101655

Abstract

With the increase in the availability of the internet, people are consuming more and more content on a regular basis and hence the chances of getting exposed to explicit content increase exponentially. To detect this explicit content from images or videos this project presents a novel Convolutional Neural Network (CNN) model for moderating explicit content in real time. A survey by the BBC has revealed that almost 62% of children who were 14 years old or younger were first exposed to inappropriate content online when they weren't expecting it. Existing solutions such Google's Parental Control and antiviruses like Quick Heal are limited to URL-blacklisting and keyword-based algorithms to prevent access to websites containing inappropriate content. Our project Real-Time Content Moderator (RTCM) uses deep learning to address the issues such as blocking entire website irrespective of the amount of obscene content, keyword-based scanning instead of content-based scanning left behind by traditional mechanisms. RTCM uses current state-of-the-art techniques to protect users from getting exposed to various explicit images and videos. RTCM is capable of censoring content present on local storage devices.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

With easy availability of gadgets, use of superior technology, rising internet speeds, access to enormous amounts of content has become possible. Internet has changed our lives in numerous ways. With just one search one can get a plethora of information on any topic. This makes tasks such as research, self learning extremely smooth and straightforward. However, such easy and unmoderated access to the internet may come at a cost . The web is an environment during which individuals are susceptible to cyber bullying, fraud, loss of privacy, most importantly, exposure to explicit and abusive content directly or indirectly. This is all the more concerning when the individual consuming such content is a child. Cell phones and the Internet have even led to new ways for the younger audience to be unintentionally exposed to explicit content. This often leaves teenagers with:

- Less interest in school
- Low academic achievement
- Depression and Anxiety
- Normalises sexual harm
- Shapes negative attitude and behaviour
- Can lead to addiction

The simple and easy solution is if guardians are involved with what kids are doing on their personal device or the internet, they are less likely to get in trouble. Of course, one can't watch their children all the time so it makes sense to use some utility tools to enforce some restriction on their access to the internet, while they're not watching them. The traditional parental control software tools use URL filtering and blacklisting of websites to completely block their access altogether for the listed websites. Widely used search engines like Google, Bing and Yahoo provide an inbuilt feature called the 'Safe Search' which prevents the search engine from displaying any adult content in the search results. A more modern and advanced approach towards solving the problem at hand is provided by the parental control applications. These applications are more efficient in solving our problems as compared to the filtering technique because a majority of the parental control applications today work both offline and online and also provide some of the advanced features like logging and reporting.

1.2 Motivation

Most of the available solutions lack in thorough scanning of content, no matter if it's online or on the device's storage disk, thus makes a pathway for exposure of inappropriate contents to users. Web filtering software only search for keywords in the URL and IP address visited irrespective of the content provided on the webpage. Scanning the content of the website can help to identify the type of content. The application software to be developed will use deep learning based explicit content classification system to address the issues left behind by available solutions.

1.3 Aim and Objective

Our aim is to create a system for moderating content in real time, i.e, detecting and eliminating explicit content from images and videos. The task would be to create a desktop application that can schedule manual and automatic scans to scan computer directories for explicit content. The software will be able to run in the background without utilizing too many computer resources.

1.4 Report Outline

This report presents a detailed approach to building the desktop application to detect explicit content from images and videos. We have provided a detailed explanation about the tools and technologies used for building the project while also presenting information about image annotation, accuracy of the trained models, training time, loss. We have also mentioned the future scope of the project and how the accuracy of the core deep learning model be improved if we were to use object detection instead of image classification.

Chapter 2

Study Of the System

2.1 Related Works

2.1.1 Explicit Content Detection System: An Approach towards a Safe and Ethical Enviornment

Ali Qamar Bhatti et al [1] proposed an explicit content detection (ECD) system to detect Not Suitable For Work (NSFW) media (i.e., image/ video) content. The proposed ECD system is based on residual network (i.e., deep learning model) which returns a probability to indicate the explicitness in media content. The value is further compared with a defned threshold to decide whether the content is explicit or nonexplicit. The proposed system not only diferentiates between explicit/nonexplicit contents but also indicates the degree of explicitness in any media content, i.e., high, medium, or low. In addition, the system also identifes the media fles with tampered extension and label them as suspicious.

2.1.2 Pornography Detection Using Support Vector Machine

Yu-Chun Lin et al [2] proposed an easy scheme for detecting pornography. They exploit primitive information from pornography and use this knowledge for determining whether a given photo belongs to pornography or not. They extract skin region from photos, and find out the correlation in skin region and non-skin region. Then, we use these correlations as the input of support vector machine (SVM) for classification.

2.1.3 An algorithm of pornographic image detection

Hong Zhu. et al [3] proposed a skin model based on the combination of YIQ, YUV, and HSV. In the step of the pre-dealing, they used white balance algorithm to achieve better skin area. Then, texture model based on Gray Level Co-Matrix (GLCM) and geometric structure of human beings were used to decrease the disruptions of the background region similar with the skin area. The features which were extracted from the last images dealt by color and texture model were input into Support Vector Machines (SVM), through which the pornographic images were classified successfully.

2.1.4 An Algorithm for Nudity Detection

Rigan Ap-apid et al [4] proposed an algorithm for detecting nudity in color images. A skin color distribution model based on the RGB, Normalized RGB, and HSV color spaces is constructed using correlation and linear regression. The skin color model is used to identify and locate skin regions in an image. These regions are analyzed for clues indicating nudity or non-nudity such as their sizes and relative distances from each other. Based on these clues and the percentage of skin in the image, an image is classified nude or non-nude.

2.2 Literature Review

2.2.1 Computer Vision

Computer vision is the field of computer science that focuses on replicating parts of the complexity of the human vision system and enabling computers to identify and process objects in images and videos in the same way that humans do. Until recently, computer vision only worked in limited capacity. Thanks to advances in artificial intelligence and innovations in deep learning and neural networks, the field has been able to take great leaps in recent years and has been able to surpass humans in some tasks related to detecting and labeling objects. One of the driving factors behind the growth of computer vision is the amount of data we generate today that is then used to train and make computer vision better.

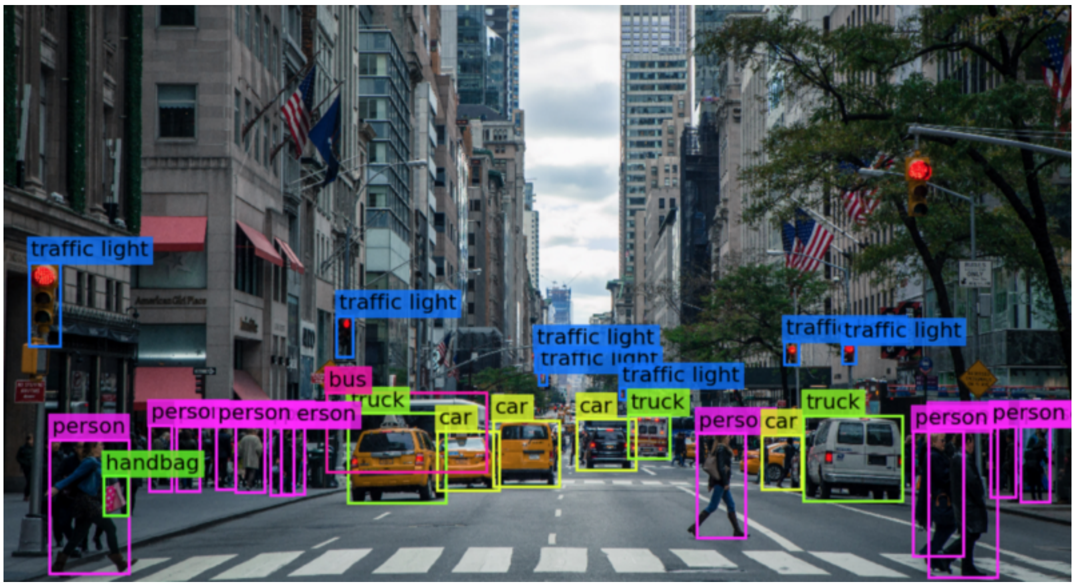


Figure 2.1: YOLO Multi-Object Detection And Classification

Along with a tremendous amount of visual data (more than 3 billion images are shared online every day), the computing power required to analyze the data is now accessible. As the field of computer vision has grown with new hardware and algorithms so has the accuracy rates for object identification. In less than a decade, today's systems have reached 99 percent accuracy from 50 percent making them more accurate than humans at quickly reacting to visual inputs.

2.2.2 Deep Learning

Deep learning is an artificial intelligence function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence (AI) that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network. In deep learning, a computer model learns to classify tasks directly from text, sound, images or video clips. Deep learning models can achieve state-of-the-art accuracy, sometimes even surpassing human-level performance. Models are trained by using huge labeled data sets and neural network architectures that contain various layers. In simple words, it is a procedure of automatically predicting and analyzing the given information by the user[5].

2.2.3 Convolutional Neural Networks

Most deep learning methods usually use a neural network architecture, which is the reason why deep learning models are often known as deep neural networks. The term “deep” usually refers to the number of hidden layers in the neural network. Our neural network contains only 2-3 hidden layers, while deep networks can have as many as 200. There is no need of manually extracting the features because deep learning models are already trained using large sets of labeled data and neural network architectures which learn features from the given data. [5]One of the most used deep neural network is convolutional neural network (CNN).

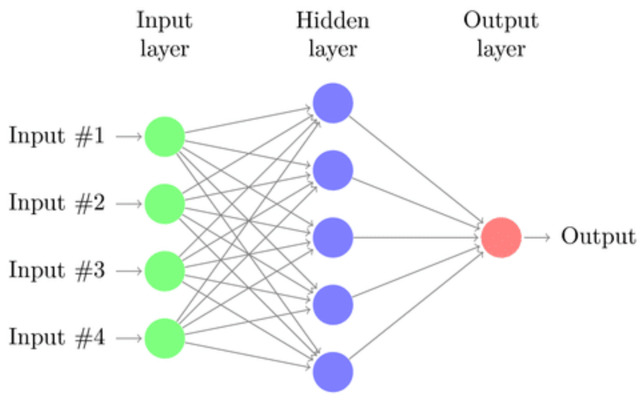


Figure 2.2: Neural Network with interconnected hidden layers

A ConvNet convolves learned features with given input data and uses convolutional layers making a well suited architecture for extracting objects directly from the images, so there is no need of identifying the features used to classify images. This way of extraction makes deep learning models to give least inaccuracy in larger context to classify objects in computer vision[6].

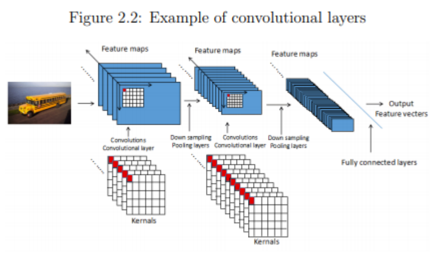


Figure 2.3: Example of convolutional layers

2.2.4 Image Classification

The convolutional neural network (CNN) is a class of deep learning neural networks. CNNs represent a huge breakthrough in image recognition. Image classification is to assign one or more labels to an image, which is one of the most fundamental tasks in computer vision and pattern recognition. In traditional image classification, low-level or mid-level features are extracted to represent the image and a trainable classifier is then used for label assignments.

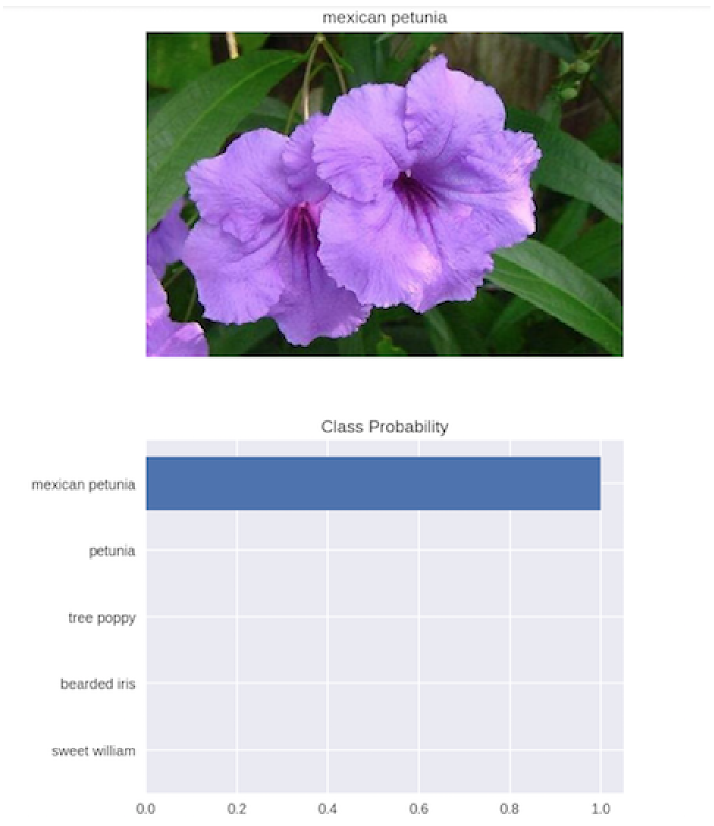


Figure 2.4: Image Classification

In recent years, the high-level feature representation of deep convolutional neural networks has proven to be superior to hand-crafted low-level and mid-level features. In the deep convolutional neural network, both feature extraction and classification networks are combined together and trained end-to-end. Deep learning techniques have also been applied to medical image classification and computer-aided diagnosis.

2.2.5 Object Detection

Object Detection is a common Computer Vision problem which deals with identifying and locating object of certain classes in the image. Interpreting the object localisation can be done in various ways, including creating a bounding box around the object or marking every pixel in the image which contains the object (called segmentation). Object detection algorithms typically use extracted features and learning algorithms to recognize instances of an object category. Refer Figure 2.1.

2.2.6 Electron JS

Electron enables you to create desktop applications with pure JavaScript by providing a runtime with rich native (operating system) APIs. You could see it as a variant of the Node.js runtime that is focused on desktop applications instead of web servers. This doesn't mean Electron is a JavaScript binding to graphical user interface (GUI) libraries. Instead, Electron uses web pages as its GUI, so you could also see it as a minimal Chromium browser, controlled by JavaScript. As far as development is concerned, an Electron application is essentially a Node.js application.

2.2.7 Adobe XD

Adobe XD is a vector-based user experience design tool for web apps and mobile apps, developed and published by Adobe Inc. XD supports website wireframing and creating click-through prototypes.

2.3 Image Processing Techniques

2.3.1 Thresholding

An image processing method that creates a bitonal (aka binary) image based on setting a threshold value on the pixel intensity of the original image. While most commonly applied to grayscale images, it can also be applied to color images. The threshold of image intensity (relative image lightness) is set manually at a specific value or automatically set by an application. Pixels below that set threshold value are converted to black (bit value of zero), and pixels above the threshold value are converted to white (a bit value of one). The thresholding process is sometimes described as separating an image into foreground values (black) and background values (white).

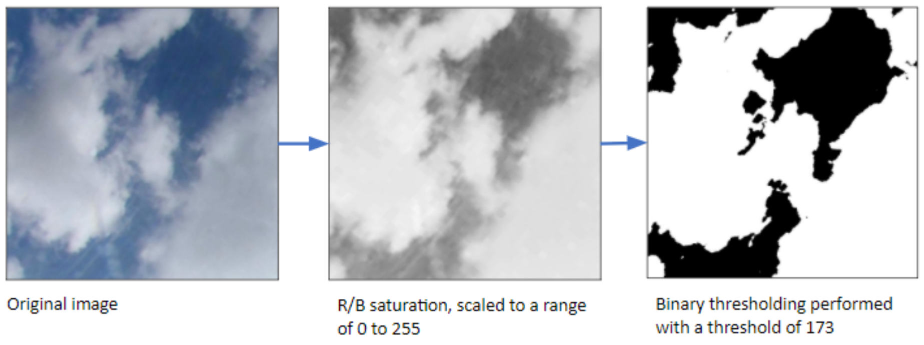

Original image
R/B saturation, scaled to a range of 0 to 255
Binary thresholding performed with a threshold of 173

Figure 2.5: Thresholding on a sample image

2.3.1.1 Binary Thresholding

The basic Thresholding technique is Binary Thresholding. For every pixel, the same threshold value is applied. If the pixel value is smaller than the threshold, it is set to 0, otherwise, it is set to a maximum value.

Binary

$$\texttt{dst}(x,y) = \begin{cases} \texttt{maxval} & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ 0 & \text{otherwise} \end{cases}$$

Figure 2.6: Binary Thresholding

2.3.1.2 Adaptive Thresholding

Adaptive thresholding, also called dynamic or local thresholding, establishes the threshold level for determining whether to convert to white or black at a regional level. The region sampled and method of evaluation vary between applications. Adaptive thresholding at a pixel level (in comparison with neighboring pixels) can yield highly superior results compared to global thresholding, particularly for images with varying levels of regional contrast differences.

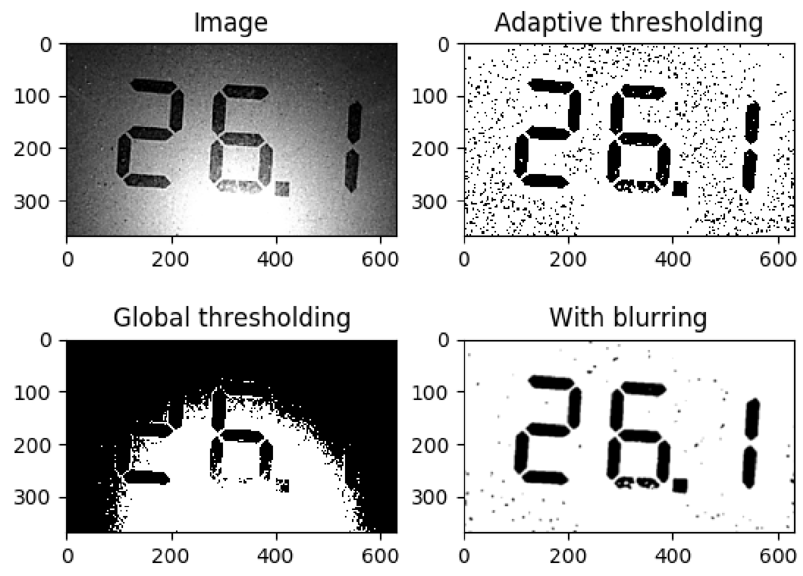


Figure 2.7: Adaptive thresholding on a sample image

2.3.1.3 OTSU Thresholding

The algorithm assumes that the image contains two classes of pixels following bi-modal histogram (foreground pixels and background pixels), It then calculates the optimum threshold separating the two classes so that their combined spread (intra-class variance) is minimal, or equivalently (because the sum of pairwise squared distances is constant), so that their inter-class variance is maximal. The aim is to find the threshold value where the sum of foreground and background spreads is at its minimum.

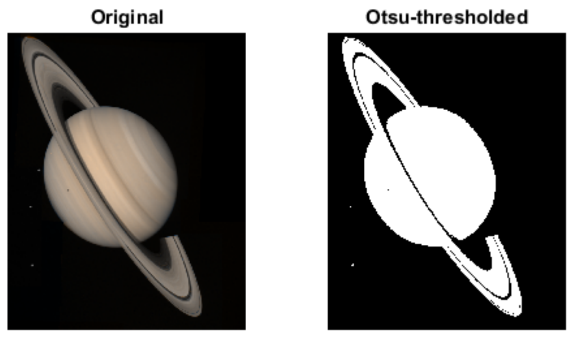


Figure 2.8: OTSU thresholding on a sample image

2.3.2 Smoothing

Smoothing is often used to reduce noise within an image or to produce a less pixelated image. Most smoothing methods are based on low pass filters. Smoothing is also usually based on a single value representing the image, such as the average value of the image or the middle (median) value.

2.3.2.1 Gaussian Blur

All images contain some amount of noise. The image is first smoothed by applying a gaussian filter A gaussian filter with $\sigma = 1.4$ is :

$$\frac{1}{273}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 7 & 4 & 1 \\ \hline 4 & 16 & 26 & 16 & 4 \\ \hline 7 & 26 & 41 & 26 & 7 \\ \hline 4 & 16 & 26 & 16 & 4 \\ \hline 1 & 4 & 7 & 4 & 1 \\ \hline \end{array}$$

Figure 2.9: Gaussian filter example

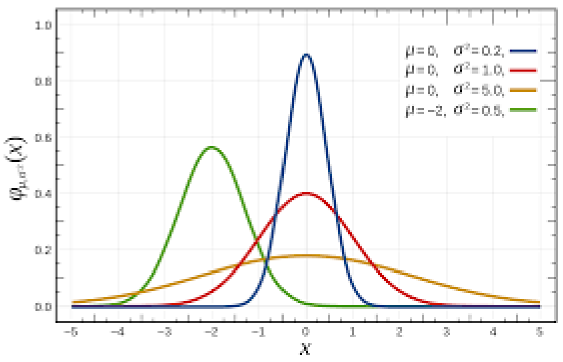


Figure 2.10: Gaussian curve

A gaussian filter prioritizes the middle pixel (anchor element) and gives less weights to the pixels away from it. (radially) This helps in preserving edges.

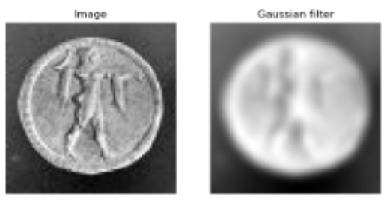


Figure 2.11: Gaussian Blur on a sample image

2.3.2.2 Median Blur

The Median blur operation is similar to the other averaging methods. The central element of the image is replaced by the median of all the pixels in the kernel area. This operation processes the edges while removing the noise. Such noise reduction is a typical pre-processing step to improve the results of later processing (for example, edge detection on an image). Median Filter: A Matrix that is used to find the median of the pixel with respect to its neighbouring pixels.

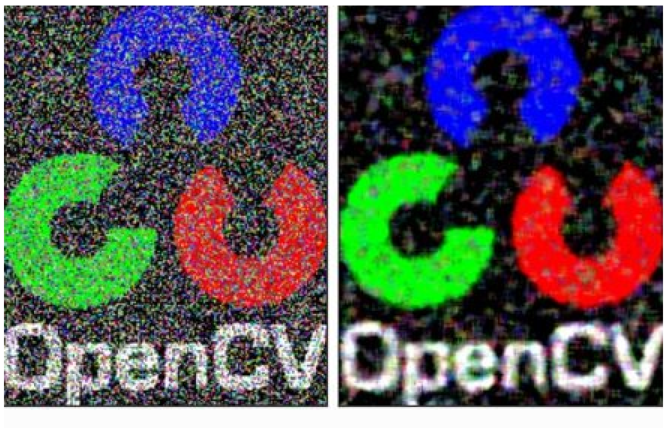


Figure 2.12: Median Blur on a sample image

2.4 Dimensionality Reduction Techniques

2.4.1 t-Distributed Stochastic Neighbor Embedding (t-SNE)

t-Distributed Stochastic Neighbor Embedding (t-SNE) is an unsupervised, non-linear technique primarily used for data exploration and visualizing high-dimensional data. In simpler terms, t-SNE gives you a feel or intuition of how the data is arranged in a high-dimensional space. It was developed by Laurens van der Maatens and Geoffrey Hinton in 2008. The t-SNE algorithm calculates a similarity measure between pairs of instances in the high dimensional space and in the low dimensional space. It then tries to optimize these two similarity measures using a cost function. The following image shows the implementation of tSNE on the MNIST dataset of 60,000 small square 28 × 28 pixel grayscale images of handwritten single digits between 0 and 9.

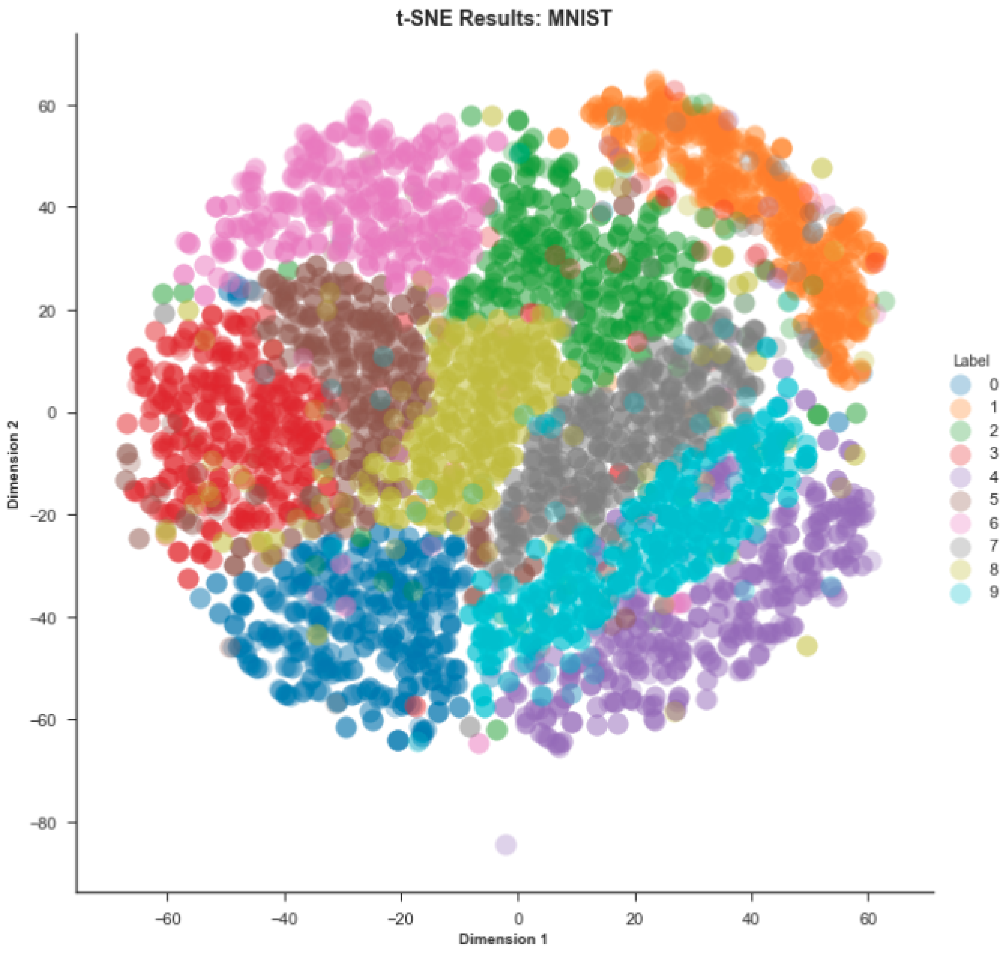


Figure 2.13: tSNE on MNIST dataset

2.4.2 Principal Component Analysis (PCA)

Principal component analysis (PCA) was developed in 1933. It is a linear dimensionality reduction technique which uses a Singular Value Decomposition of the data to project it to a lower dimensional space. The input data is centered but not scaled for each feature before applying the SVD. In other words, things that are different end up far apart. This can lead to poor visualization especially when dealing with non-linear manifold structures like cylinder, ball, curve. The following image shows the implementation of PCA on the MNIST dataset of 60,000 small square 28 × 28 pixel grayscale images of handwritten single digits between 0 and 9.

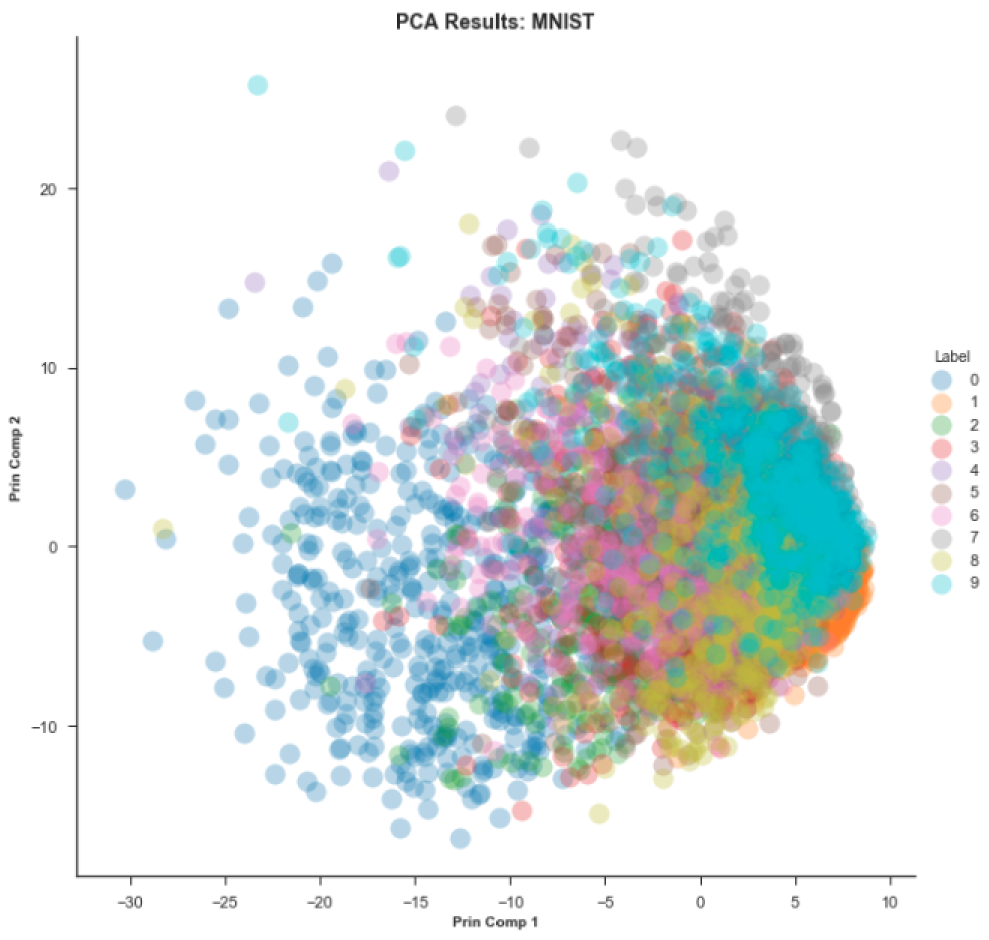


Figure 2.14: PCA on MNIST dataset

Chapter 3

Proposed System

3.1 Problem Statement

In the recent years,there has been a massive increase in the generation and consumption of digital contents which are freely available to the users. Even the categories of available contents vary a lot depending on the websites and there is no effective identity checking technique available as of now. This is not the only case, there might be scenarios where the inappropriate contents, which are not be suitable for a specific category of users like kids, are present offline on the user's device. The consumption of adult contents by a younger audience has a negative impact on their physical as well as psychological well-being, in some cases which can lead to some complexities. There are cases of employees accessing NSFW (Not Suitable For Work) contents during office hours or on organizational asset like Laptop, PC, Phone. Hence, organizations try to block the NSFW contents because it causes distraction and can compromise the organization asset by the means of viruses, ransomwares and other malicious softwares which are often present on websites. Organizations with strict compliance monitor the activities of their employees like contents accessed by them online and other various activities they perform on organizations' network and assets. It has been observed quite frequently that even the strictest compliance policies cannot guarantee blocking of NSFW contents, because in several cases these contents are present on trusted website like Reddit, Tumblr,etc. The traditional content censoring systems work by the technique of blocking URL of unsafe websites which is both inefficient and can also be fooled easily by the use of proxies. In Order to prevent users from using alternative ways to access the explicit content, there is a need of an application which is monitoring the type of content accessed. Another problem with URL blocking approach is that numerous websites are activated every second arid even the URL list maintained for is also not up-to-date. Also, these lists only contain the URL of topmost sites and thus this approach fails for new URLs and unpopular websites. To tackle the above problems, we are proposing an application which performs content-based censoring using CNN (Convolutional Neural Network). The proposed application will work for contents which are accessible offline as well as which are available on the internet. This application will overcome the problems faced by existing systems and will improve the restriction on the access of explicit contents. Thus, this application will give ability to parents and organizations to regulate the type of content being consumed on their devices.

3.2 Scope

The purpose of this system is to create a safe environment for the users that prevents exposure to any type of obscene content. The system mainly provides two functionalities:

- Image Censoring
- Video Censoring

using deep learning models. The main aim of the system is to provide a system environment that allows to prevent accidental or deliberate exposure to any type of obscene content. The system will be capable of predicting the category of the images or videos and act accordingly. The extension will fetch all image and videos in the system. The system will feed images to the prediction model which will return the predicted class of the image. On classification the system will decide if the content should be censored or not. The system will censor the content files accessed from the offline storage on the local disk storage. The image will be fed through a prediction model which will be preloaded in the system for faster processing and using deep compression the size of the model can be reduced. The system can be installed in personal computers at home, work-space in corporates, library in schools/colleges. The primary objective of this system is to enforce an environment that prevents an accidental or deliberate exposure to the user. The user will not be able to access any explicit or objectionable content on the local drive. This system can help to deflate the NSFW (Not Safe For Work) access in the corporate work-spaces and other inappropriate places. Ultimately this software can be used as a protective shield against the exposure to any type of explicit content.

3.3 Proposed System

3.3.1 Approach

MobileNetV1 use Depthwise Separable Convolution which dramatically reduce the complexity cost and model size of the network, which is suitable to Mobile devices, or any devices with low computational power. In MobileNetV2, a better module is introduced with inverted residual structure. Non-linearities in narrow layers are removed this time. With MobileNetV2 as backbone for feature extraction, state-of-the-art performances are also achieved for object detection and semantic segmentation. Overall, the MobileNetV2 models are faster for the same accuracy across the entire latency spectrum. In particular, the new models use 2x fewer operations, need 30% fewer parameters and are about 30-40% faster than MobileNetV1 models, all while achieving higher accuracy.

In order to train the classification model we have to scrap the internet to collect explicit and non explicit images to create the dataset. Our goal is to integrate this prediction model with a Graphical User Interface (GUI) and thus creating a desktop application capable of scanning through the machine's file system to detect explicit images and videos and thus censor the content.

3.3.2 Computer Vision Module

This module consists of:

- Classification
- Object Detection

3.3.2.1 Classification

Dataset Collection for Classification :
A python package called "google_images_download" was used to scrap google images using keywords. This enabled us to simply use keywords such as "explicit images", "violence", "pornography", "gruesome" and scrap google images and create the dataset. One disadvantage of this is that it scraped only 100 images of each keyword every time the script was run. Hence by using variation of the keywords like "pornography male", "pornography female", "violence blood", we were eventually able to create a dataset with 7000 unique images, i,e, roughly 1000 images of each class. Next, we split our dataset into train and test datasets. The training and test split was 90-10, i.e, 900 images from each class made the training dataset and 100 images from each class made the testing dataset. As a result we had 6300 train images and 700 test images.

Traing the Classification model:
MobileNet V2 model accepts one of the following formats: (96, 96), (128, 128), (160, 160), (192, 192), or (224, 224). In addition, the image has to be 3 channel (RGB) format. Therefore, We resized & converted our images. from (28×28) to $(96 \times 96 \times 3)$. MobileNet is pretrained with ImageNet dataset. ImageNet is an image dataset organized according to the WordNet hierarchy. Each meaningful concept in WordNet, possibly described by multiple words or word phrases, is called a "synonym set"

or ”synset”. There are more than 100,000 synsets in WordNet, majority of them are nouns (80,000+). ImageNet provides on average 1000 images to illustrate each synset. Images of each concept are quality-controlled and human-annotated [7]. We transfer-learned the MobileNet v2 on our dataset which consisted of 6300 training images from 7 classes. The dataset contained both explicit and non-explicit images. One essential step in our approach is fine tuning where we changed the learning rate of the model to a smaller one, so it does not have a significant impact on the already adjusted weights. We ran the training over 10 epochs and monitored the accuracy and the loss after every epoch.

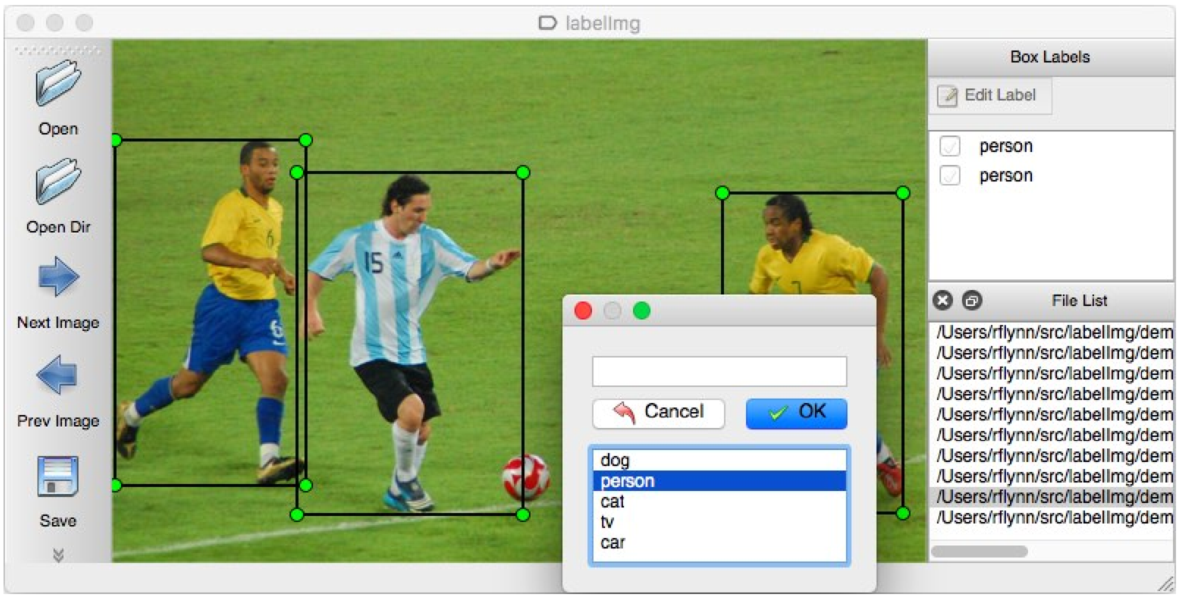
Testing the Classification model:
We tested the model on 700 images from the 7 classes. The loss over 10 epochs was 1.5% and the accuracy was 85.8%. The number of output classes was 2, i.e ”Explicit” and ”Non-Explicit”
Training and testing of the classification model for explicit content detection was performed on the Nvidia GeForce 1050 Ti which has a memory size of 4GB to observe the performance in terms of detection accuracy and speed.

3.3.2.2 Object Detection

Dataset Collection for Object Detection :
The dataset consists of 500 images from 5 explicit classes, i.e, 100 images of each class. The images in the dataset were generated by crawling the internet. Out of the 100 images from each class, 80 were used for training and 20 were used for testing ,i.e, 400 images were train images and 100 were test images. The images consisted of one or more naked human subjects both male and female. Also the images consisted of subjects with different skin colors, body hair, lighting conditions, intensity, percentage of body exposed and grayscale images. This presents a significant additional challenge as convolutional neural networks can be expected to perform best, in general, when the input data is as uniform and standardized as possible. This includes standardization in terms of colour, contrast, scale, and class balance.

Ground Truth Generation:
Preparing the data involved manually annotating the image, i.e creating bounding boxes around the object to be detected using the labelImg image annotator tool [8]. The tool can be downloaded from its github page and follows a simple installation. This tool is used to define regions in an image

and create textual descriptions of those regions. Annotations are saved as XML files in PASCAL VOC format, the format used by ImageNet. Besides, it also supports YOLO format. An illustration of this tool is given in 3.1.



/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem

Figure 3.1: Manually annotated image using labelImg

Training the Object Detection model:
We trained two object detection models:

- Faster R-CNN
- SSD MobileNet v2

The implementation of the Faster R-CNN is done in Tensorflow [9] and it utilizes Inception-ResNet [10] as a backbone. Due to the very less amount of Training data, transfer learning was deemed as a viable option in which the pretrained weights of the MS-COCO dataset [11] are used as a starting point for training. We trained the model for 150k steps with initial learning rate as 0.0002 and batch size 1.
The implementation of the SSD MobileNet v2 is done in Tensorflow. We trained this model for 60k steps with initial learning rate as 0.004 and batch size 24.

Testing the Object Detection model:
The inference tables for the results on images and videos are as follows:

Model	Inference Speed	COCO mAP
Faster R-CNN	15ms	28

Table 3.1: Inference for Faster R-CNN on images

Model	Average FPS in videos
Faster R-CNN	30

Table 3.2: Inference for Faster R-CNN on videos

Model	Inference Speed	COCO mAP
SSD MobileNetv2	10ms	22

Table 3.3: Inference for SSD MobileNet v2 on images

Model	Average FPS in videos
SSD MobileNetv2	75

Table 3.4: Inference for SSD MobileNetv2 on videos

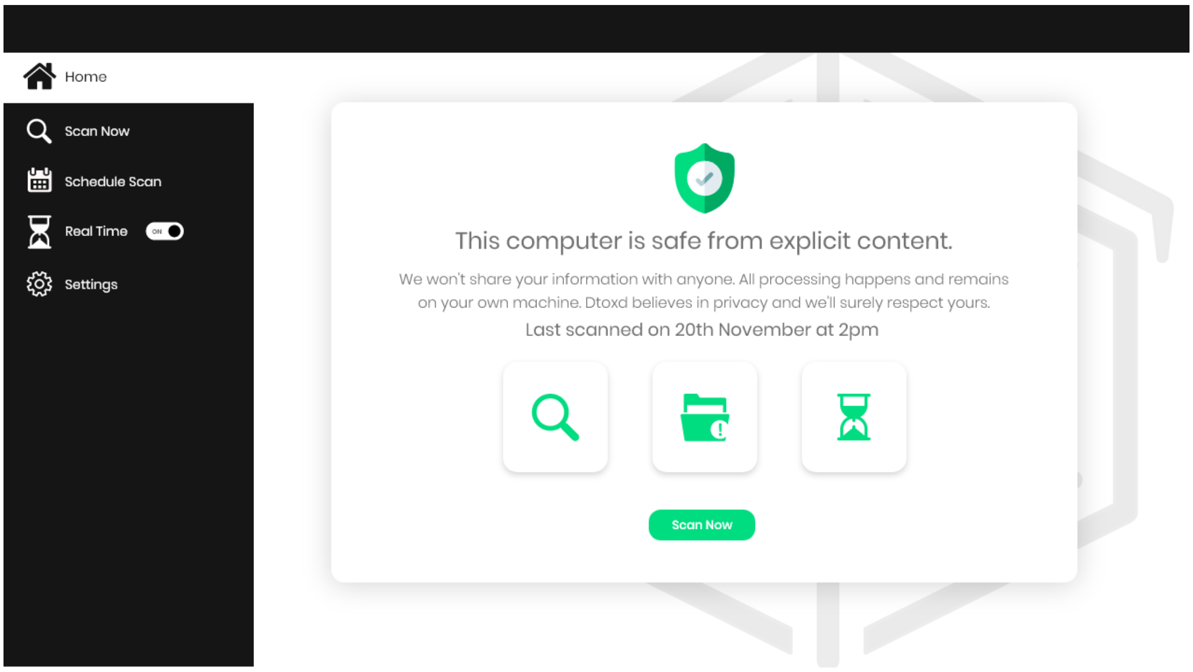
Training and testing of both Object Detection models was performed on the Nvidia GeForce 1050 Ti which has a memory size of 4GB to observe the performance in terms of detection accuracy and speed.

3.3.3 Graphical User Interface module

The Graphical User Interface was prototyped in Adobe XD and the front end was developed using Electron JS. The server was a python server which received requests from Electron JS.

3.3.3.1 Home

The .exe file is installed on to the PC. On running the app, the user is greeted with a Home Page with three toggles and a nav bar on the left.



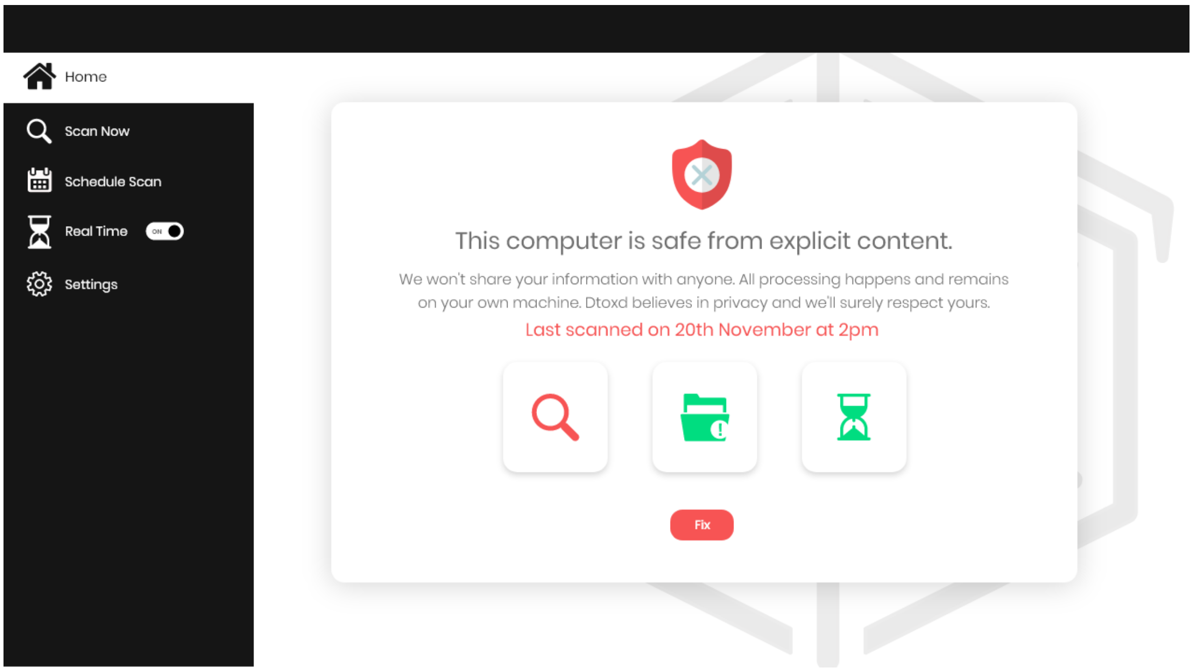
This computer is safe from explicit content.

We won't share your information with anyone. All processing happens and remains on your own machine. Dtoxd believes in privacy and we'll surely respect yours.

Last scanned on 20th November at 2pm

Figure 3.2: Home Menu

The "Search" toggle shows you when your PC was last scanned.



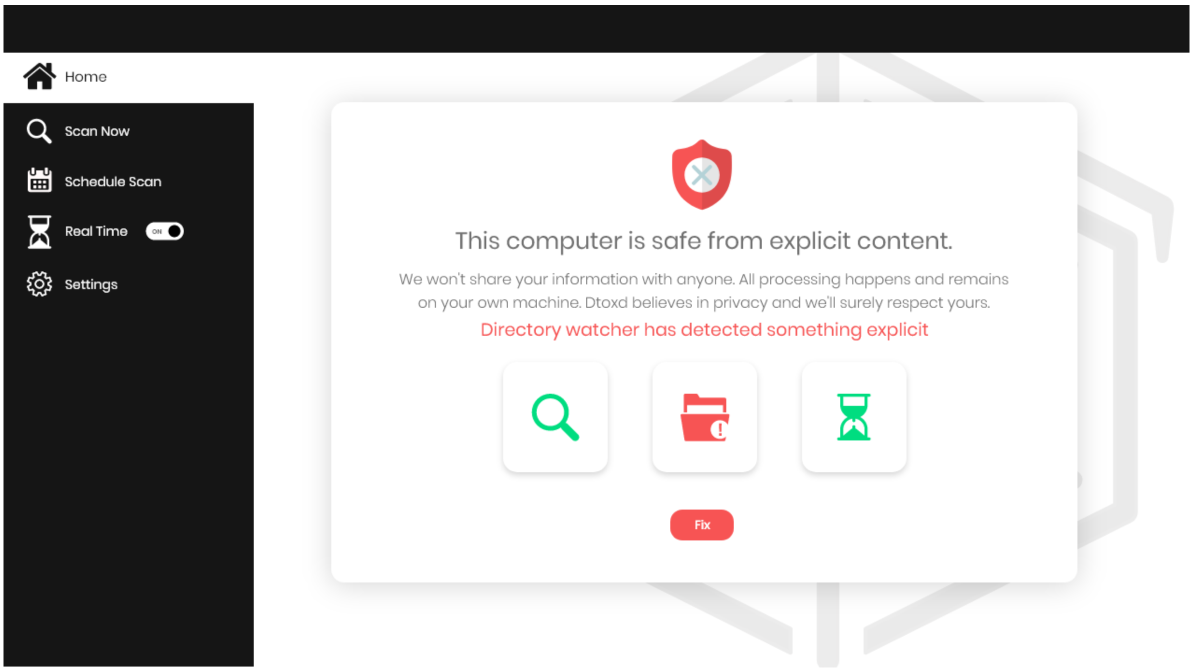
This computer is safe from explicit content.

We won't share your information with anyone. All processing happens and remains on your own machine. Dtoxd believes in privacy and we'll surely respect yours.

Last scanned on 20th November at 2pm

Figure 3.3: Search Toggle

The "Directory Watcher" toggle shows you the directories with explicit content it found the during the last scan and the "Fix" button allows the user to delete all the explicit content from all the directories.



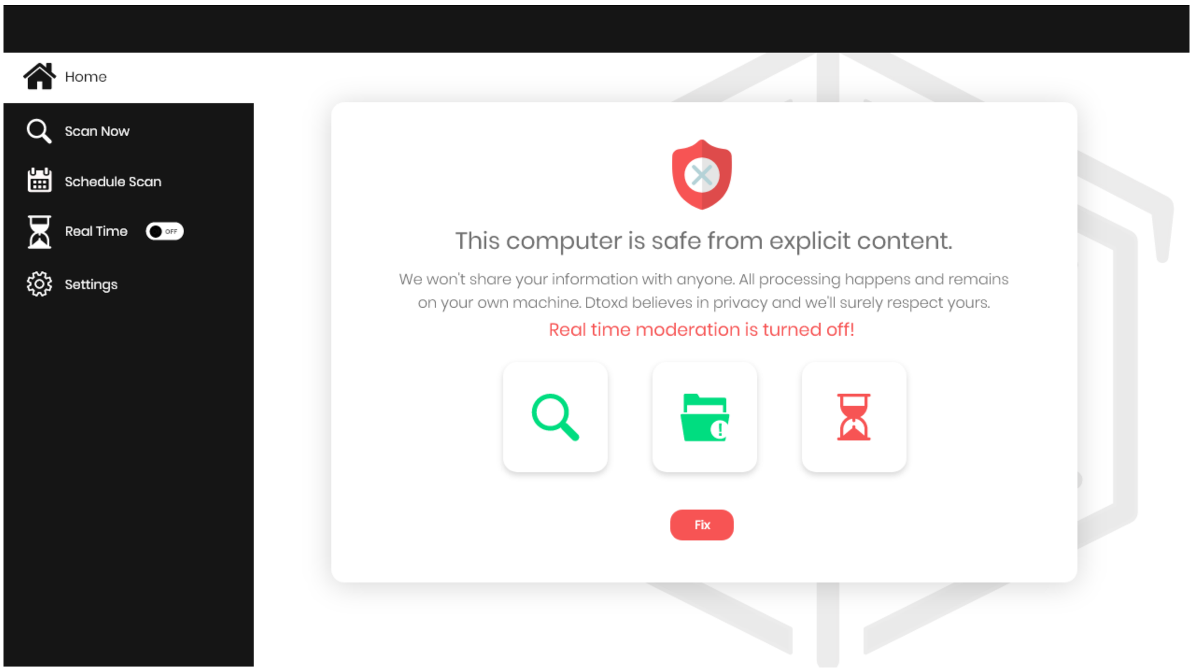
This computer is safe from explicit content.

We won't share your information with anyone. All processing happens and remains on your own machine. Dtoxd believes in privacy and we'll surely respect yours.

Directory watcher has detected something explicit

Figure 3.4: Directory Watcher Toggle

The "Real Time" toggle turns the real-time scan ON or OFF.



This computer is safe from explicit content.

We won't share your information with anyone. All processing happens and remains on your own machine. Dtoxd believes in privacy and we'll surely respect yours.

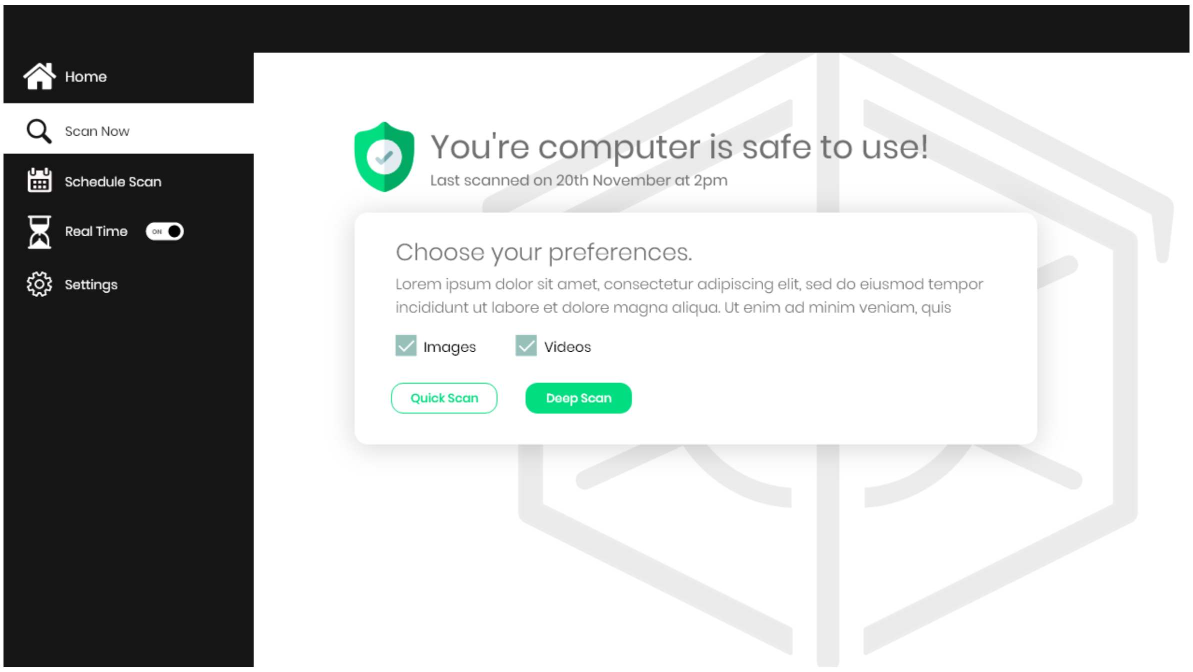
Real time moderation is turned off!

Figure 3.5: Real Time Toggle

3.3.3.2 Scan Now

The second toggle on the nav bar on the left is the ”Scan Now” menu. The user can select either ”Quick Scan” or ”Deep Scan” and also images or videos or both.

- Quick Scan: Scans till the 3rd sub-directory of the offline storage system and thus is quicker.
- Deep Scan: Scans the entire file system of the offline storage and hence takes time depending on the user's machine.



You're computer is safe to use!

Last scanned on 20th November at 2pm

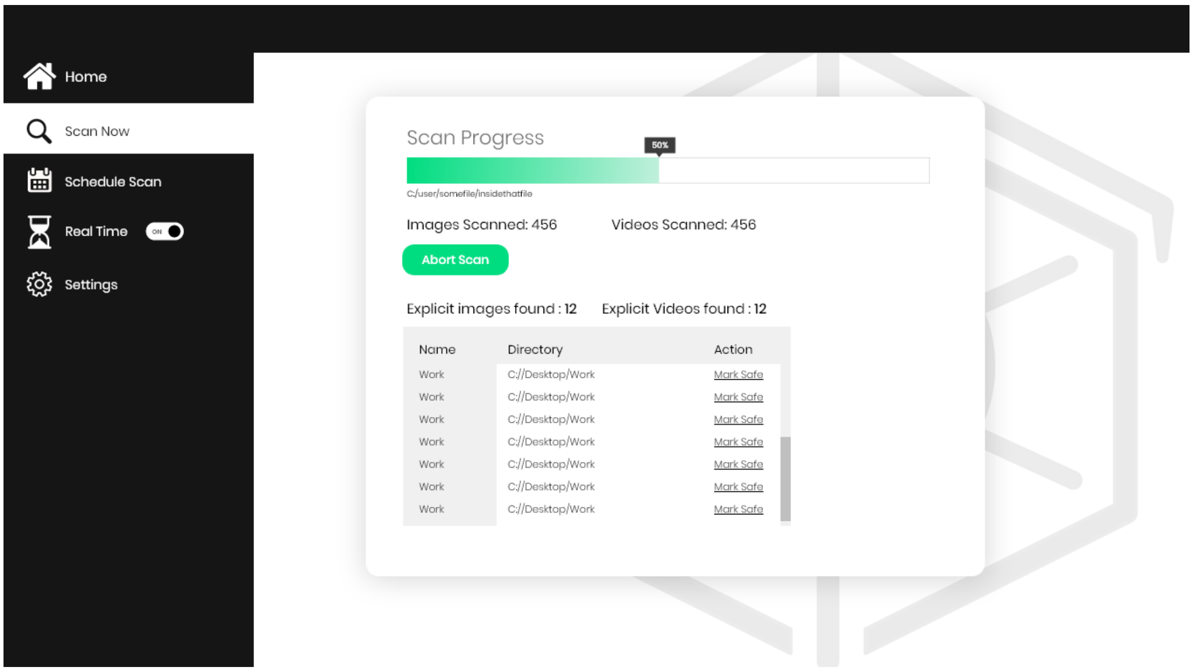
Choose your preferences.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis

✓ Images ✓ Videos

Figure 3.6: Scan Type and Image/Videos Selection

Once the user clicks on ”Scan Now” the scan begins. The progress of the scan is displayed on the screen. All of the directories being scanned are shown alongwith the total images and videos scanned and the number of explicit images and videos found. The user may also ”Abort Scan”.



Scan Progress

C:/user/somefile/insidethatfile

Images Scanned: 456 Videos Scanned: 456

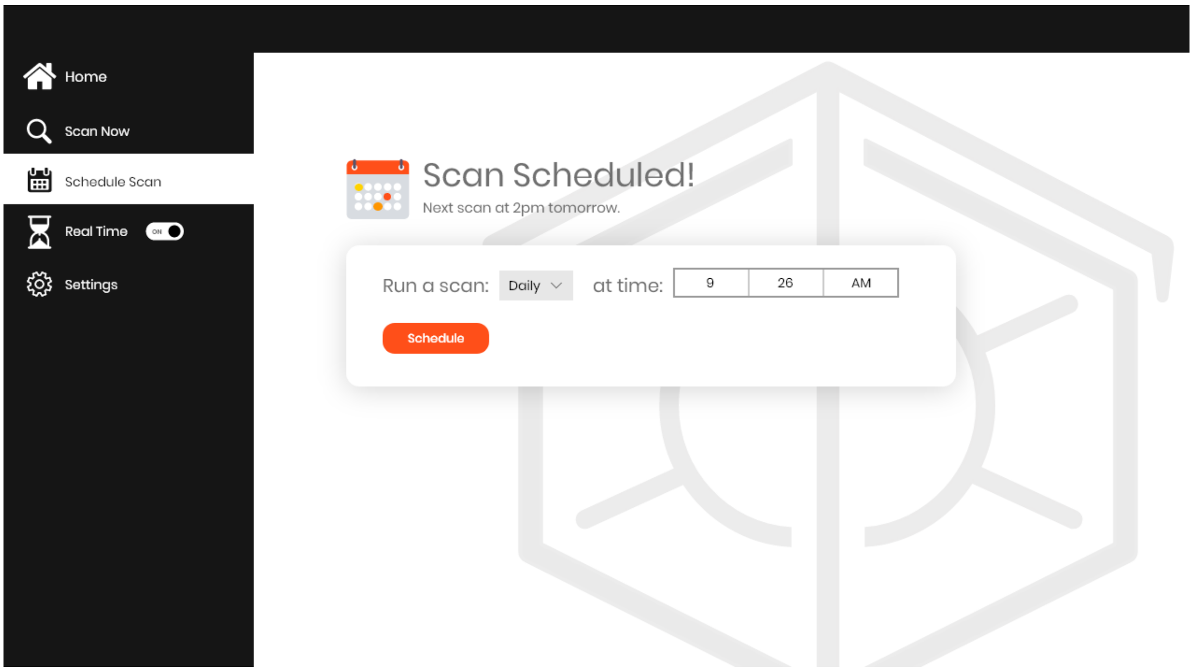
Explicit images found : 12 Explicit Videos found : 12

Name	Directory	Action
Work	C://Desktop/Work	Mark Safe
Work	C://Desktop/Work	Mark Safe
Work	C://Desktop/Work	Mark Safe
Work	C://Desktop/Work	Mark Safe
Work	C://Desktop/Work	Mark Safe
Work	C://Desktop/Work	Mark Safe
Work	C://Desktop/Work	Mark Safe

Figure 3.7: Scan Progress with scan details

3.3.3.3 Schedule Scan

The third toggle on the nav bar on the left is the "Schedule Scan" menu. Schedule Scan allows the user to schedule a Deep or a Quick Scan whenever they want. Once scheduled, the scan runs in the background utilizing minimal resources(CPU & RAM) and thus not disrupting any other process running simultaneously.



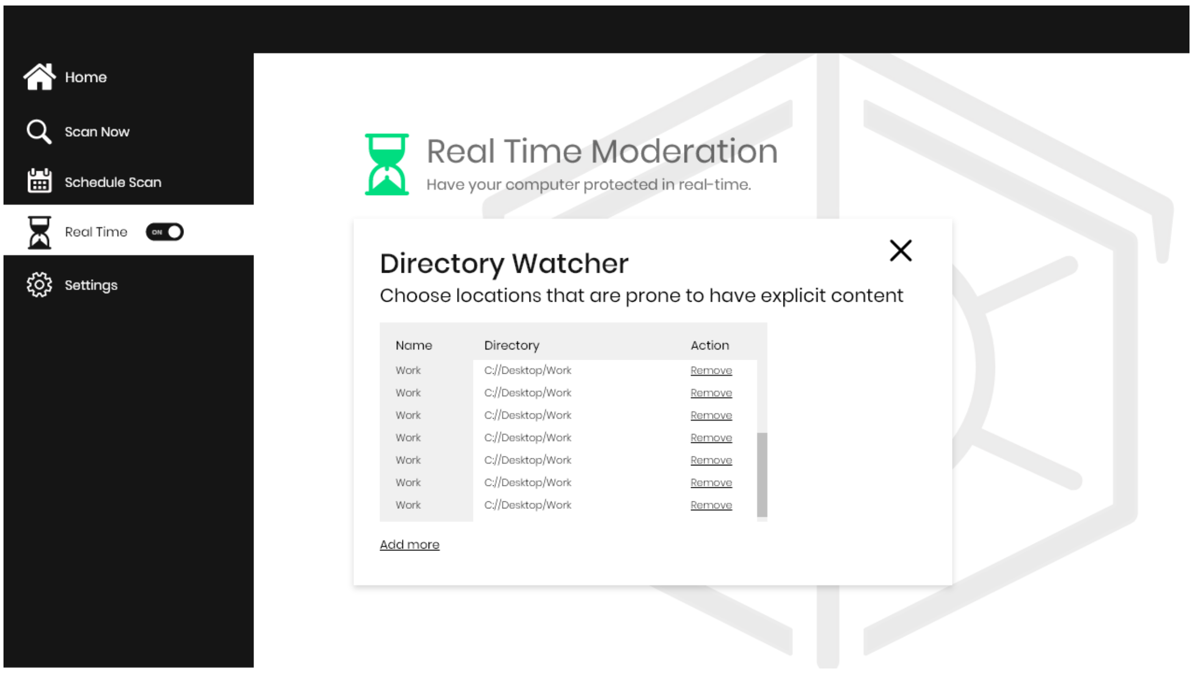
Scan Scheduled!
Next scan at 2pm tomorrow.

Run a scan: Daily at time: 9 26 AM

Figure 3.8: Schedule Scan

3.3.3.4 Real Time

This allows the user to add add directories which may be high risk. For instance, the downloads directory or the C drive. This may be turned ON or OFF as mentioned in 3.5. If turned OFF the scan may proceed as it is. If it is a Deep Scan it may scan all the directories anyway. However, with Quick Scan enabled the user may add other directories too since it does not scan the entire file system.



Directory Watcher

Choose locations that are prone to have explicit content

Name	Directory	Action
Work	C://Desktop/Work	Remove
Work	C://Desktop/Work	Remove
Work	C://Desktop/Work	Remove
Work	C://Desktop/Work	Remove
Work	C://Desktop/Work	Remove
Work	C://Desktop/Work	Remove
Work	C://Desktop/Work	Remove

Figure 3.9: Add Directory

3.3.3.5 Settings Toggle

The settings toggle is the last toggle on the nav bar. It provides extra functionalities to enhance the user experience of the application.

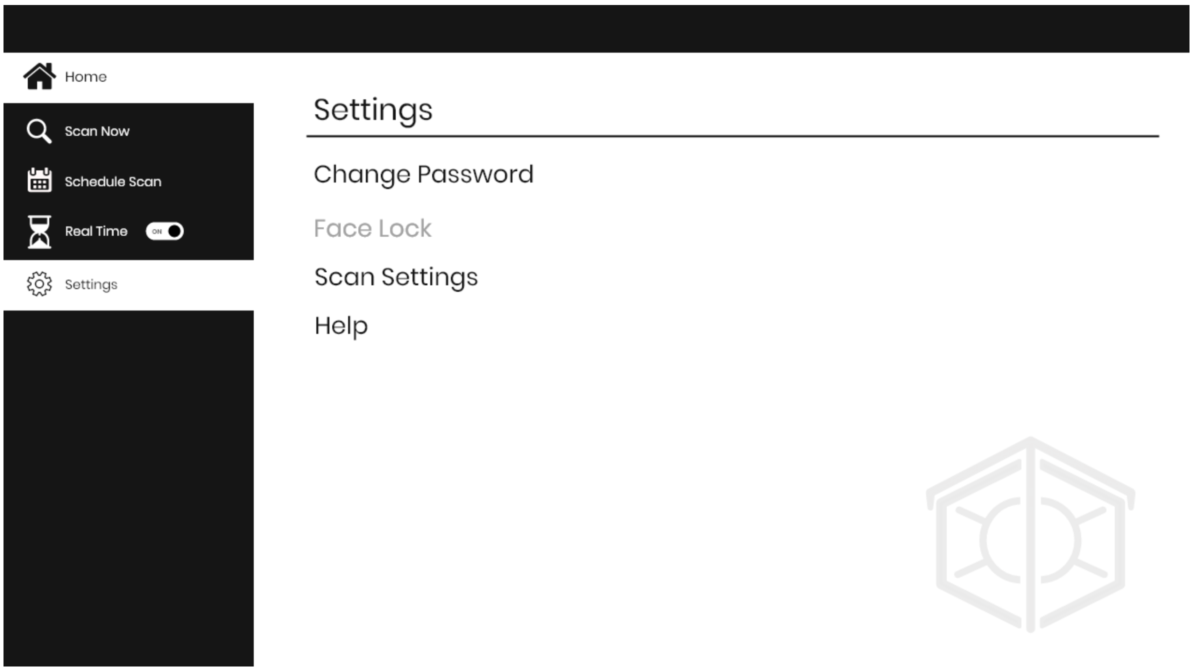


Figure 3.10: Settings Menu

It allows the user to:

- Change Password

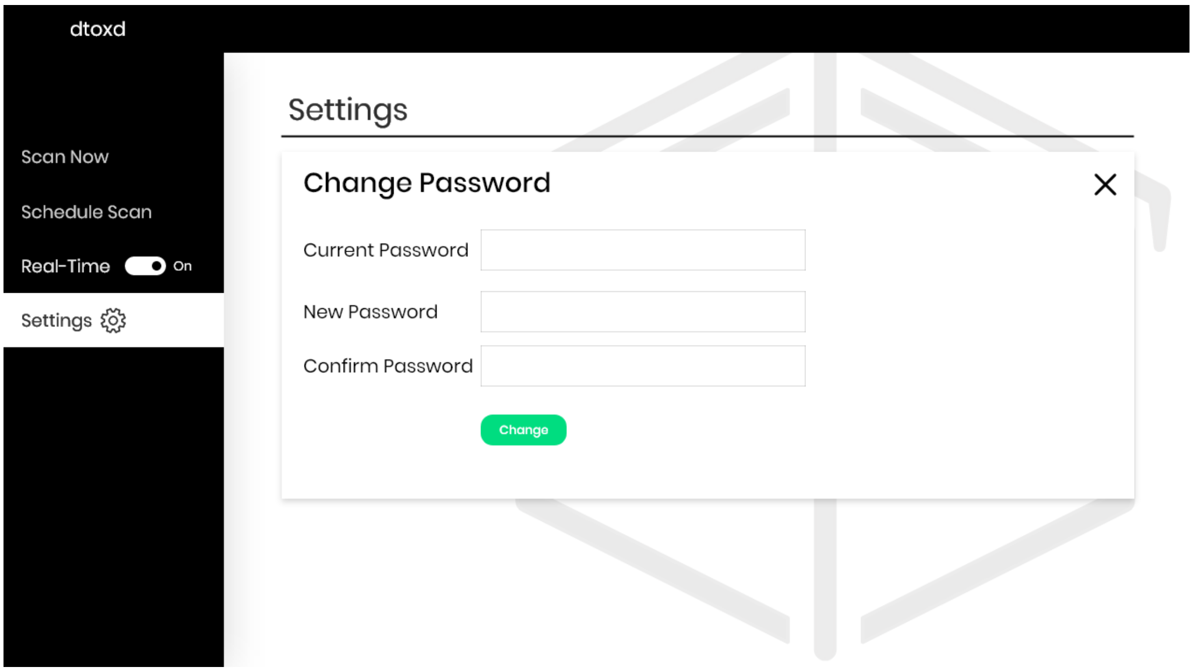
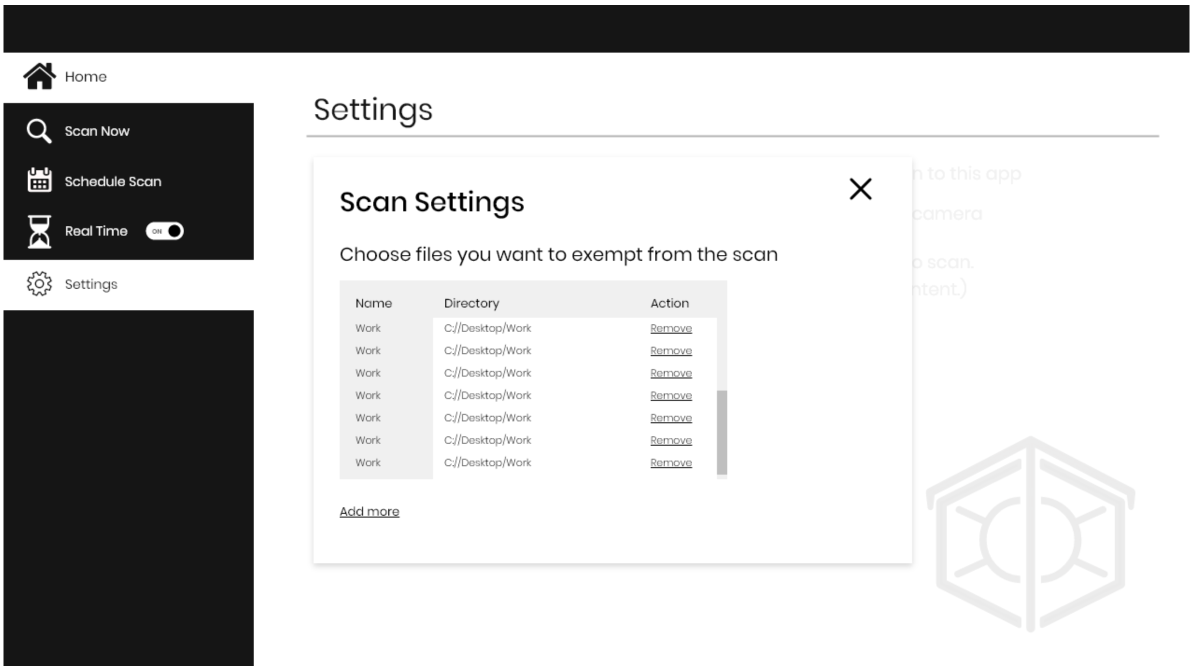


Figure 3.11: Change Password

- Change Scan Settings



Settings

Scan Settings

Choose files you want to exempt from the scan

Name	Directory	Action
Work	C://Desktop/Work	Remove
Work	C://Desktop/Work	Remove
Work	C://Desktop/Work	Remove
Work	C://Desktop/Work	Remove
Work	C://Desktop/Work	Remove
Work	C://Desktop/Work	Remove
Work	C://Desktop/Work	Remove

Add more

Figure 3.12: Change Scan Settings

Chapter 4

Design Of the System

4.1 Requirement Engineering

4.1.1 Requirement Elicitation

Sr.No	Type	Short Description
F1	Functional	User should be able to click on Home Page toggles
I1	Informational	The system displays date of last scan,directories where explicit content was found after the previous scan, real time toggle ON/OFF, on the home page
F2	Functional	User should be able to select Images, Videos or both to be scanned
F3	Functional	User should be able to abort scan if needed
F4	Functional	User should be able to add directories to be scanned
I2	Informational	The system should display which directory is being scanned
I3	Informational	The system should display the results after the scan is complete
F5	Functional	User should be able to exempt directories from the scan
I4	Informational	The system should consist of guides to help the user

Table 4.1: Requirement Elicitation

4.1.2 Software lifecycle model

The Software lifecycle model used was Agile Framework. The entire System was completed in an iterative manner wherein each phase modules were developed and at the end of the lifecycle are integrated. Reasons to use agile:-

Faster Time to Market:
Agile frameworks emphasize the Minimum Viable Product (MVP) and the Minimal Marketable Product (MMP). These are two Lean concepts that allow for a small, safe investment that can be delivered quickly.

Stakeholder Satisfaction:
Stakeholders are users, customers, or anyone that has a vested interest in the software. In Agile approaches, stakeholders are heavily engaged in the process and their feedback is constantly solicited. To ensure that we're adding value every step of the way, we ask the stakeholders their opinions during iterations.

Organizational Transparency:
Agile is a total, holistic process that creates organizational transparency and camaraderie. People are honest and work towards a common goal and achieve what they have set out to achieve. The political landscape is lowered. The open office environments that are encouraged flatten the organizational hierarchy and create a greater sense of community.

4.1.3 Requirement Analysis

4.1.3.1 Data Flow Diagram

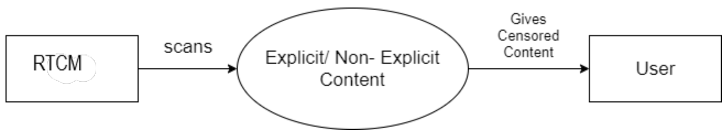


Figure 4.1: Level 0 DFD

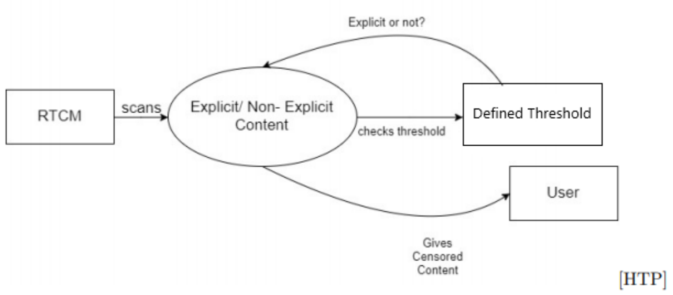


Figure 4.2: Level 1 DFD

4.1.3.2 Activity Diagram

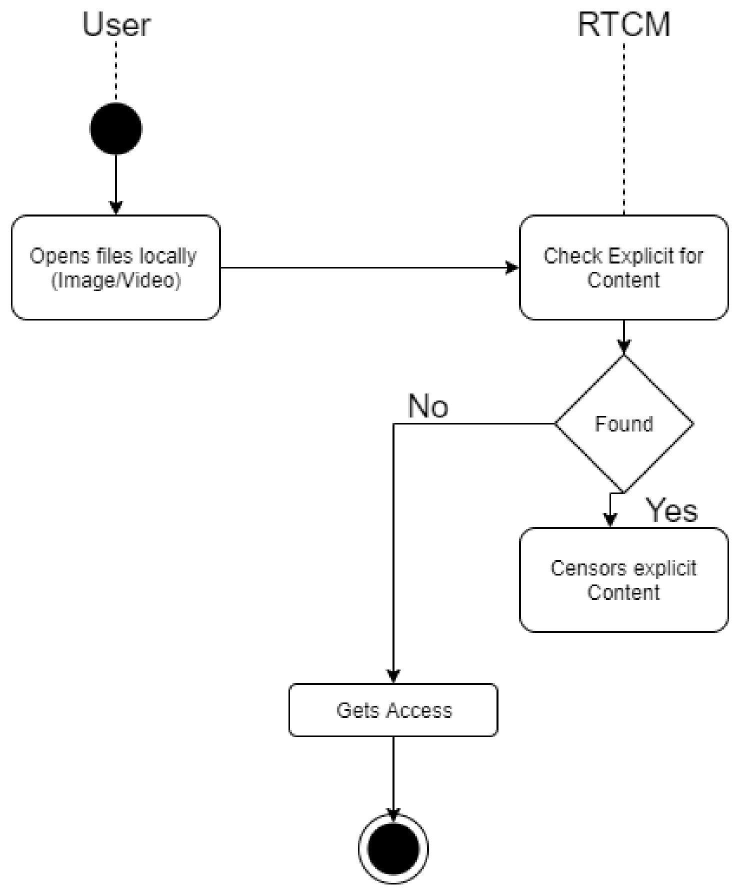


Figure 4.3: User Activity Diagram

4.1.3.3 Cost Analysis

Laptop/ Desktop Computer: The software requires a Windows based system to install and run. The application in its current state will run smoothly if the system has a quad-core CPU with a dedicated GPU. The cost incurred to buy a laptop/computer is subjective.

4.1.3.4 Software Requirements

- Operating System: Windows 7 or higher
- Drivers: Windows Defualt Drivers
- Front End: Node JS

- Backend: Python 3
- Tensorflow Environment

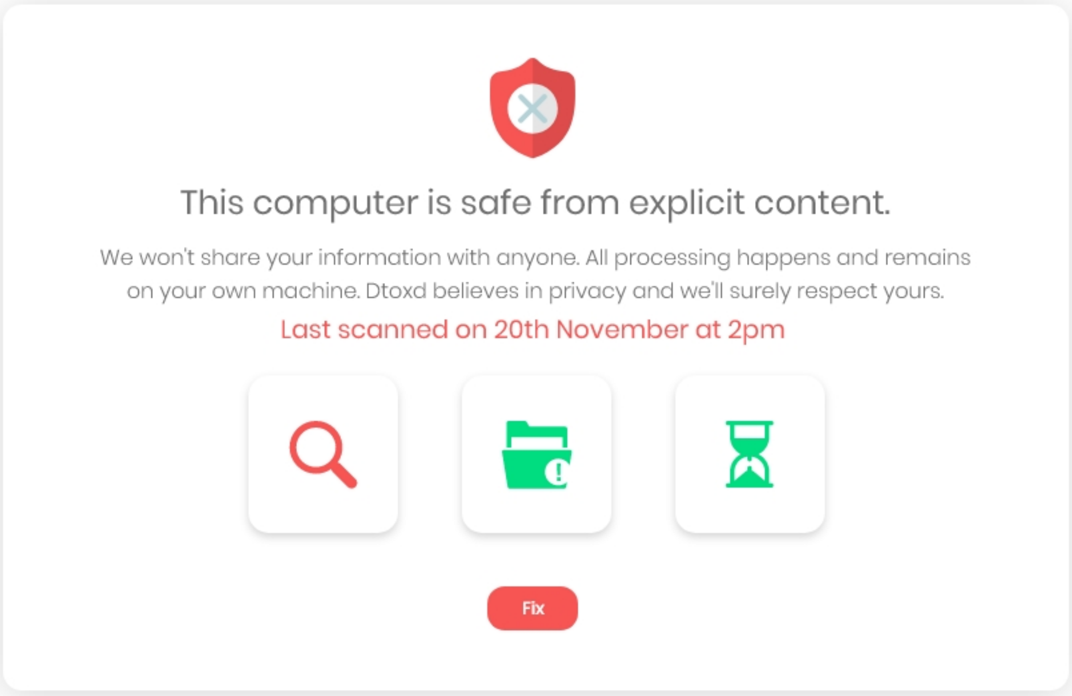
4.1.3.5 Hardware Requirements

- Intel Pentium IV or above
- $\geq$ 4GB RAM
- $\geq$ 500 GB HDD Memory
- Internet Connectivity

4.2 System architecture

4.2.1 UI/UX diagram

Dashboard:
The Dashboard features everything the user need to know at a glance. It includes a welcome message and some flags that the user might want to know about.



This computer is safe from explicit content.

We won't share your information with anyone. All processing happens and remains on your own machine. Dtoxd believes in privacy and we'll surely respect yours.

Last scanned on 20th November at 2pm

Figure 4.4: Dashboard User Interface

Navigation Bar:
The Navigation bar groups all the important features into separate tabs. This maintains a fixed visual hierarchy for the user and focuses on the task at hand. Less relevant features are tucked away in their own subcategories.

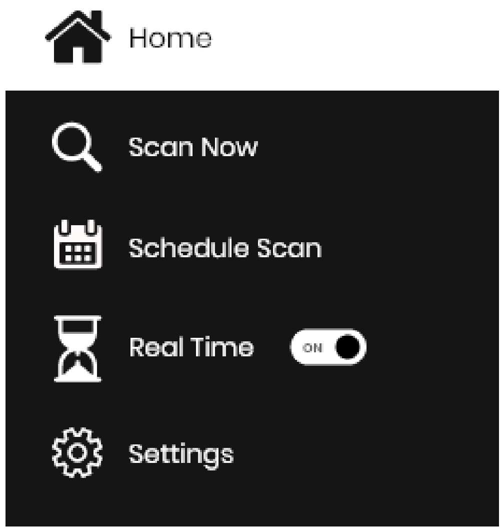
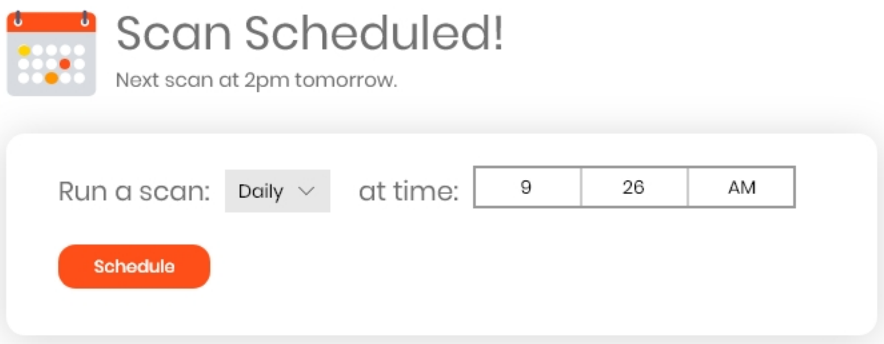


Figure 4.5: Navigation Bar User Interface

Schedule Scan:
Schedule Scan allows the user to schedule a Deep or a Quick Scan whenever they want. Once scheduled, the scan runs in the background utilizing minimal resources(CPU & RAM) and thus not disrupting any other process running simultaneously.

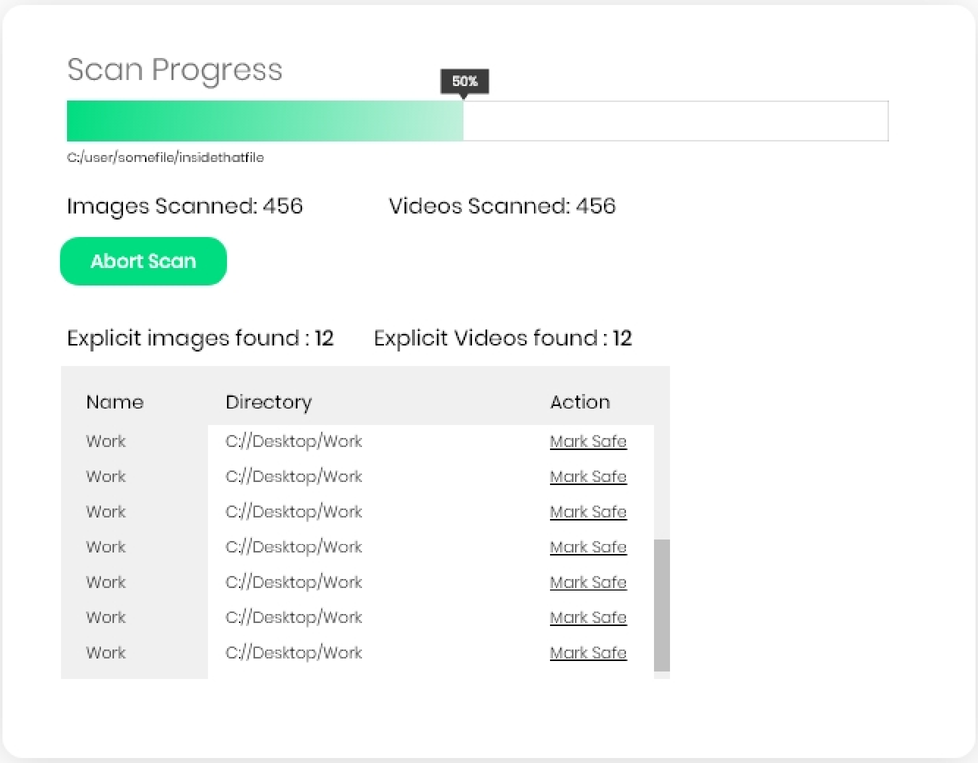

Scan Scheduled!

Next scan at 2pm tomorrow.

Run a scan: Daily at time: 9 26 AM

Figure 4.6: Schedule Scan User Interface

Scan Progress:
It is very important that the user is continuously notified about a task that's being performed. The progress is the best visual tool there is to achieve that. We also plan to display files that are currently being scan in the production version of the app. And of course, the user can always cancel or stop the scan whenever necessary. Granting complete control to the user.


Scan Progress

50%

C:/user/somefile/insidethatfile

Images Scanned: 456 Videos Scanned: 456

Abort Scan

Explicit images found : 12 Explicit Videos found : 12

Name	Directory	Action
Work	C://Desktop/Work	Mark Safe
Work	C://Desktop/Work	Mark Safe
Work	C://Desktop/Work	Mark Safe
Work	C://Desktop/Work	Mark Safe
Work	C://Desktop/Work	Mark Safe
Work	C://Desktop/Work	Mark Safe
Work	C://Desktop/Work	Mark Safe

Figure 4.7: Scan Progress User Interface

Style Guide:
Here's some of the visual guidelines we followed while designing the app. The color schemes, the font family and icons were predetermined to make the overall design uniform and less cluttered.

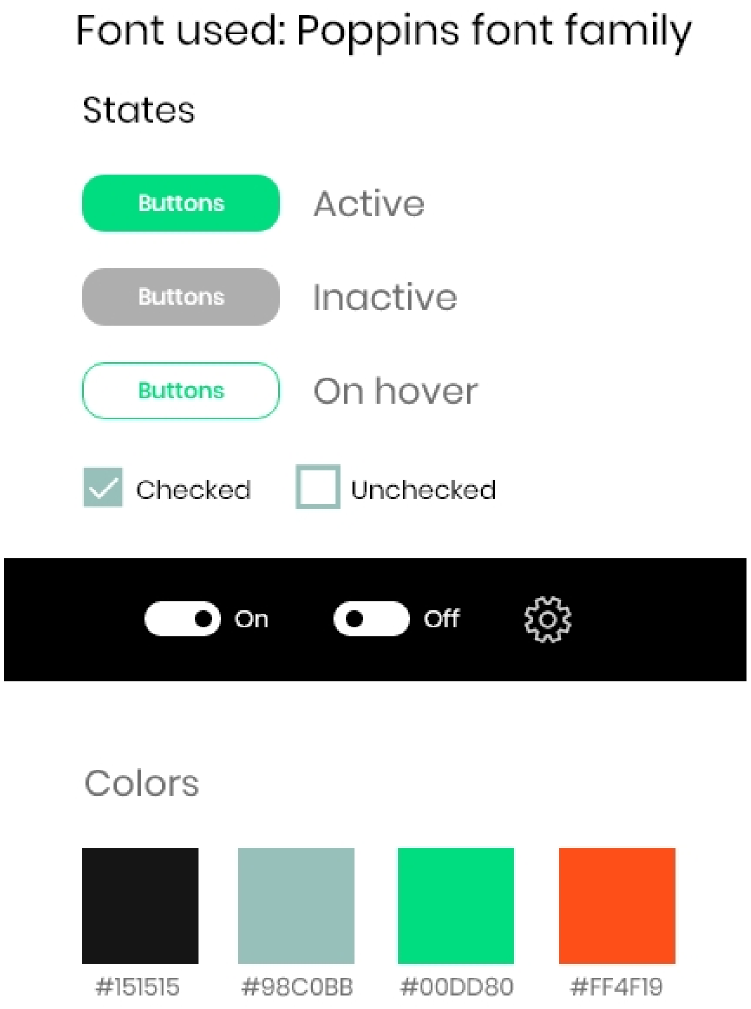


Figure 4.8: Style Guide Used

4.2.2 User Flow Diagram

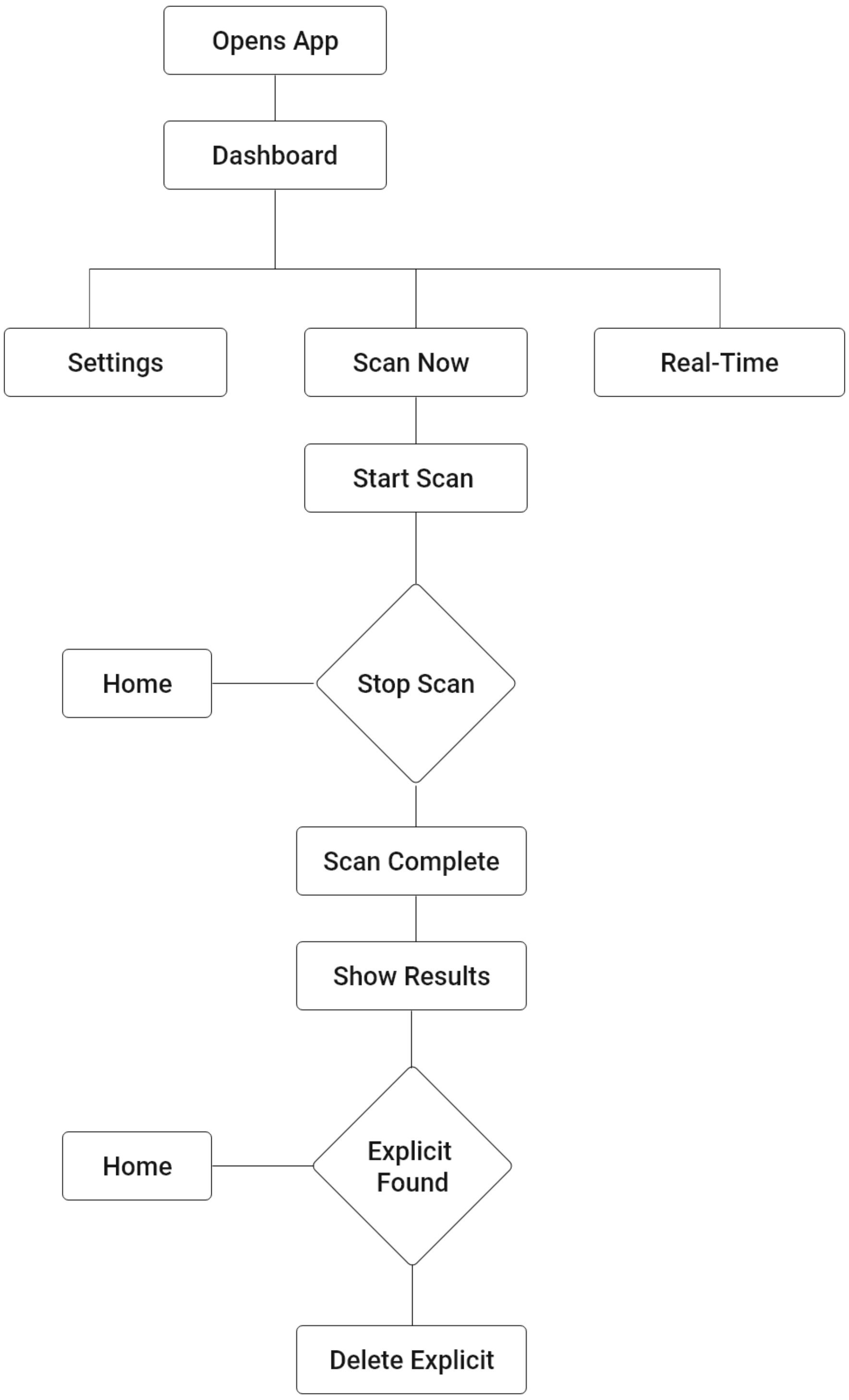


Figure 4.9: User Flow Diagram for RTCM

4.2.3 Block Diagram

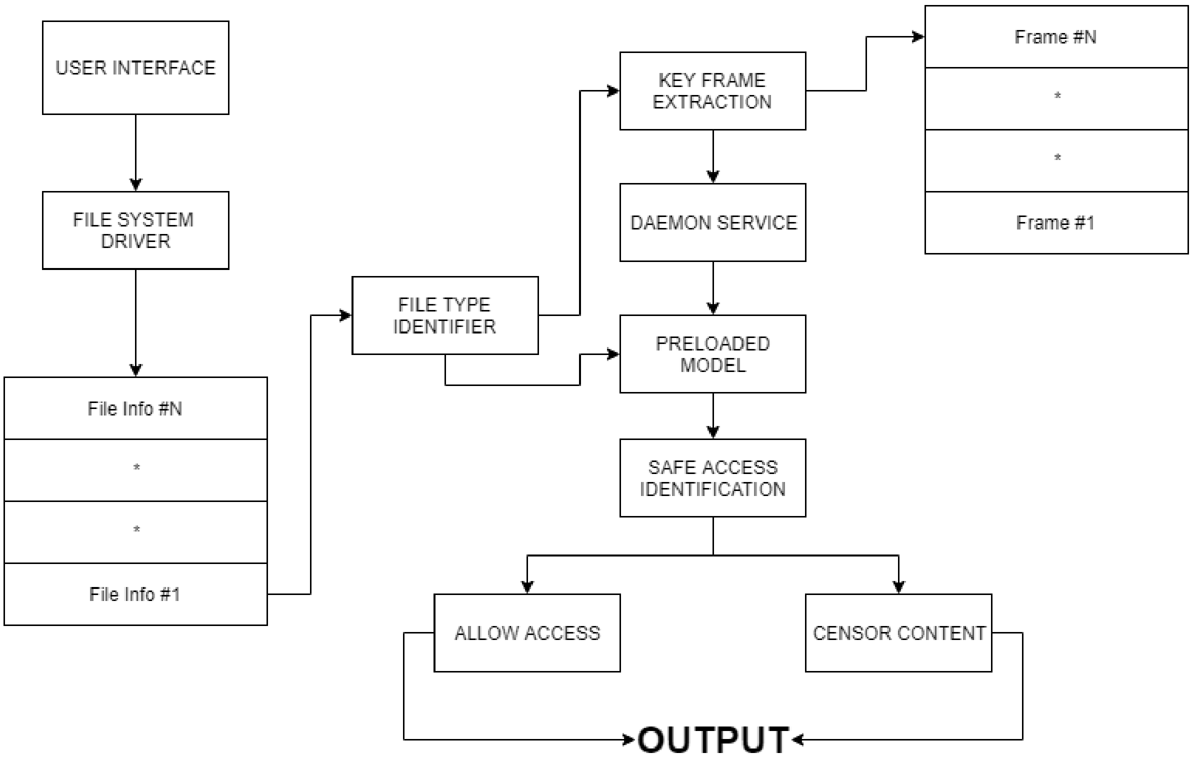
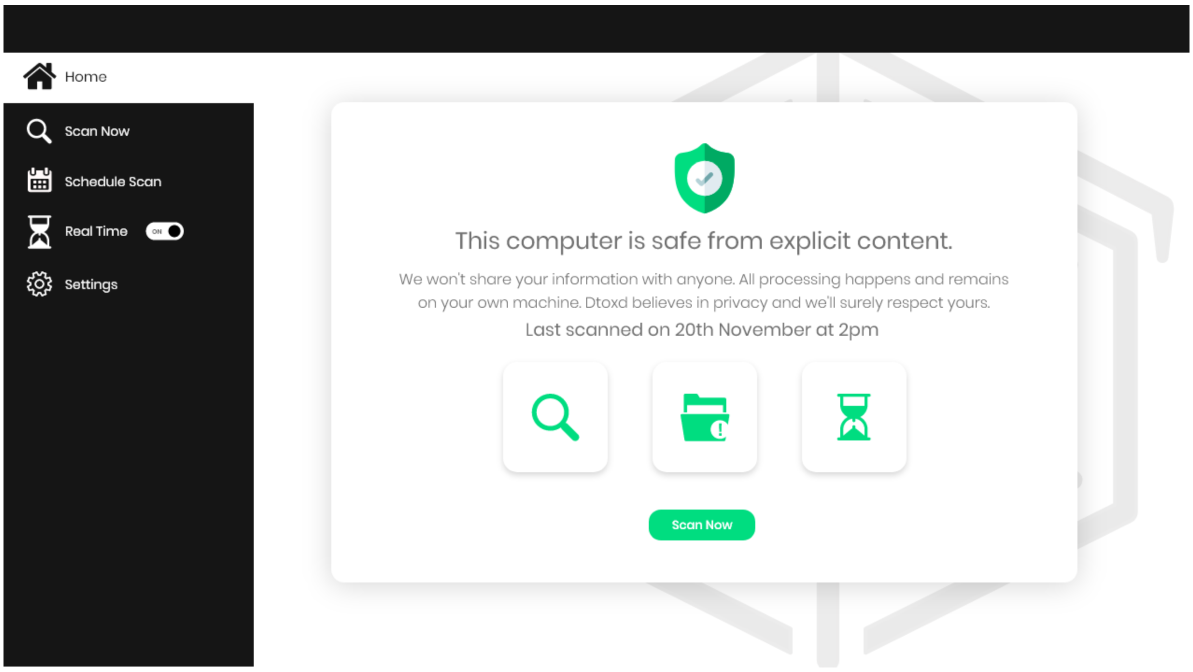


Figure 4.10: Block Diagram of RTCM

Chapter 5

Result and Discussion

5.1 Screenshots of the System

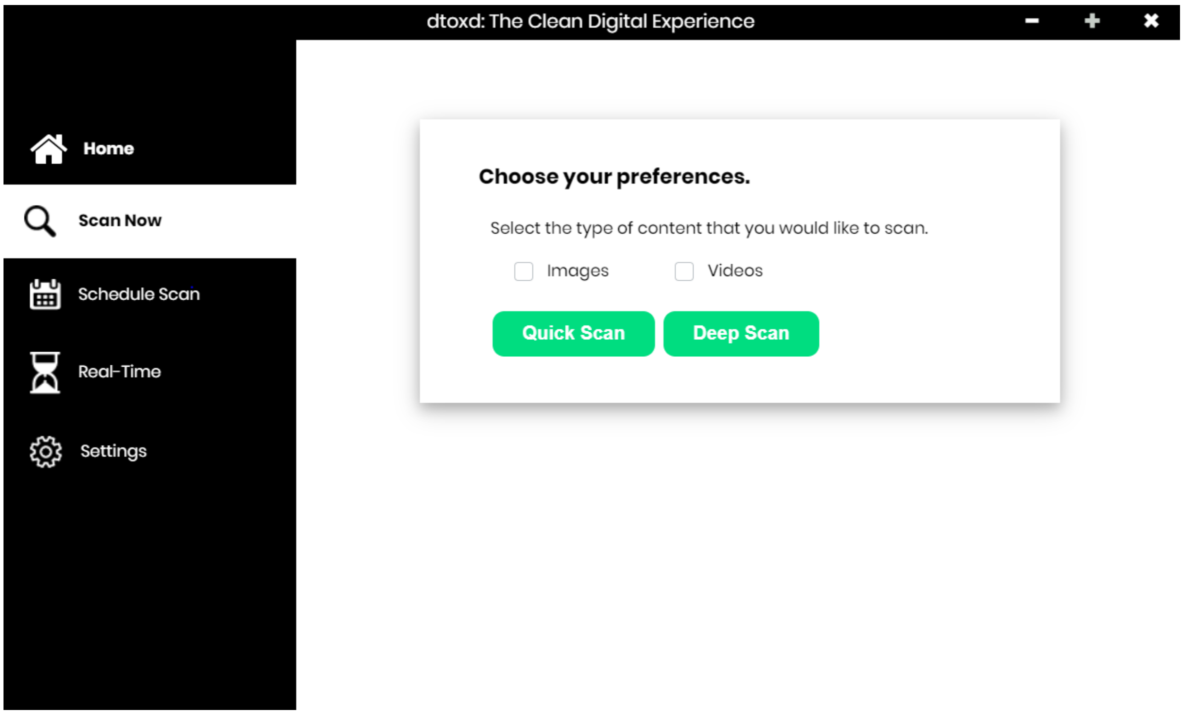


This computer is safe from explicit content.

We won't share your information with anyone. All processing happens and remains on your own machine. Dtoxd believes in privacy and we'll surely respect yours.

Last scanned on 20th November at 2pm

Figure 5.1: Dashboard

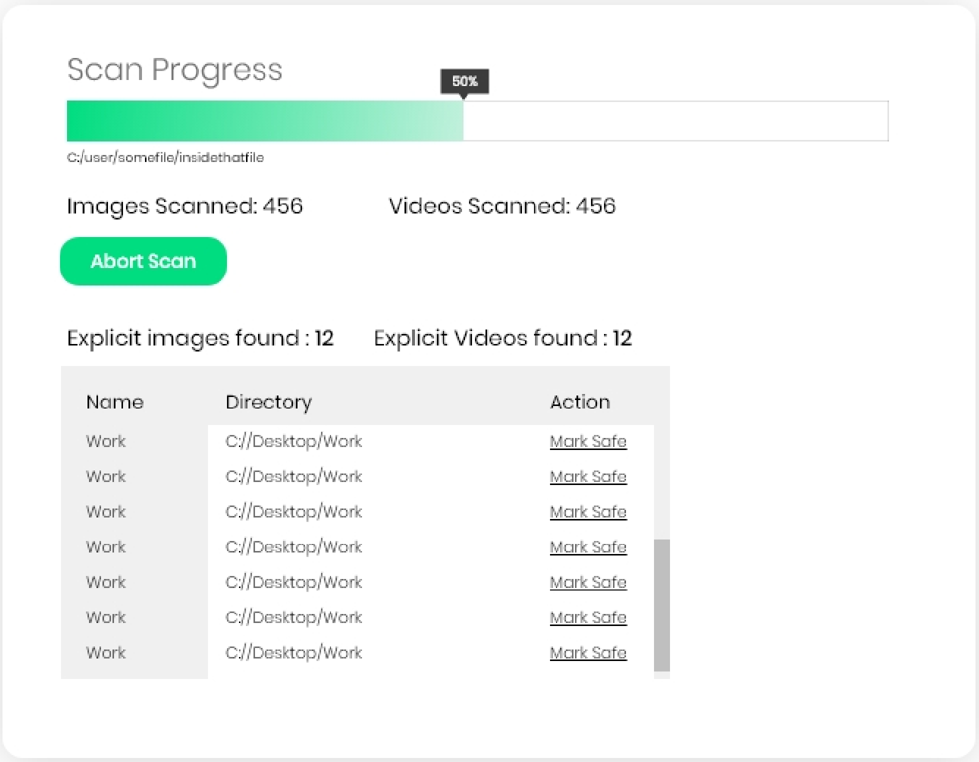


Choose your preferences.

Select the type of content that you would like to scan.

☐ Images ☐ Videos

Figure 5.2: Quick Scan & Deep Scan Preferences



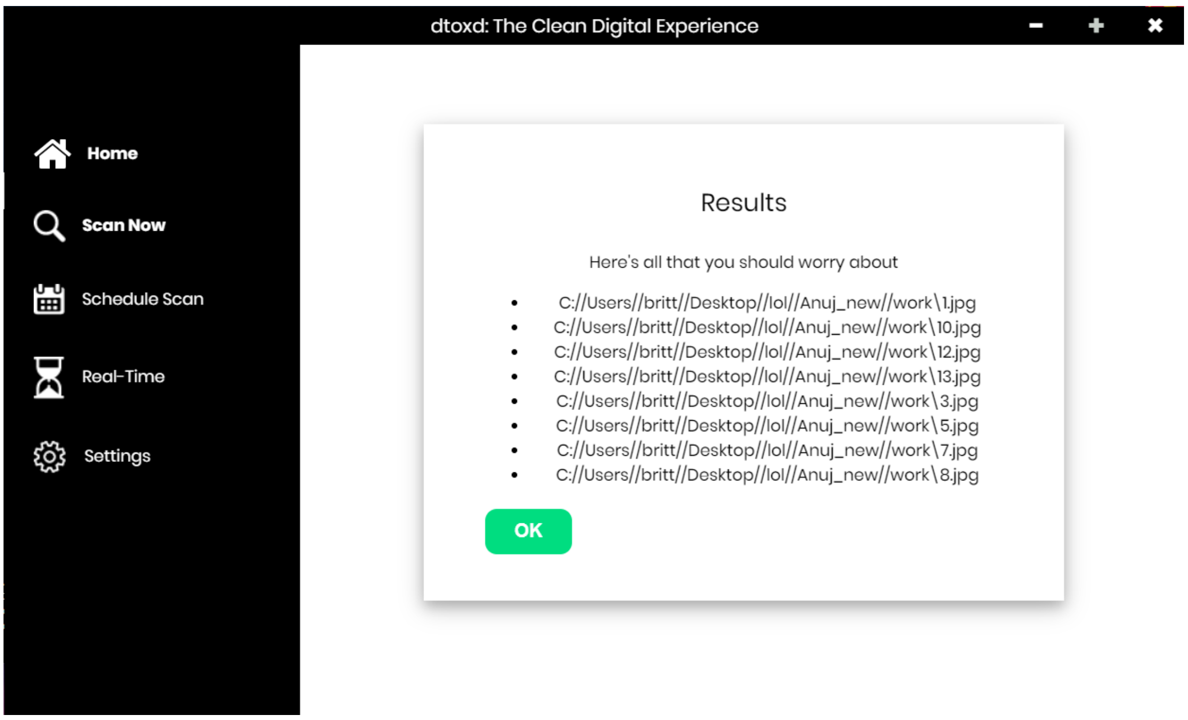
C:/user/somefile/insidethatfile

Images Scanned: 456 Videos Scanned: 456

Explicit images found : 12 Explicit Videos found : 12

Name	Directory	Action
Work	C://Desktop/Work	Mark Safe
Work	C://Desktop/Work	Mark Safe
Work	C://Desktop/Work	Mark Safe
Work	C://Desktop/Work	Mark Safe
Work	C://Desktop/Work	Mark Safe
Work	C://Desktop/Work	Mark Safe
Work	C://Desktop/Work	Mark Safe

Figure 5.3: Scan Progress

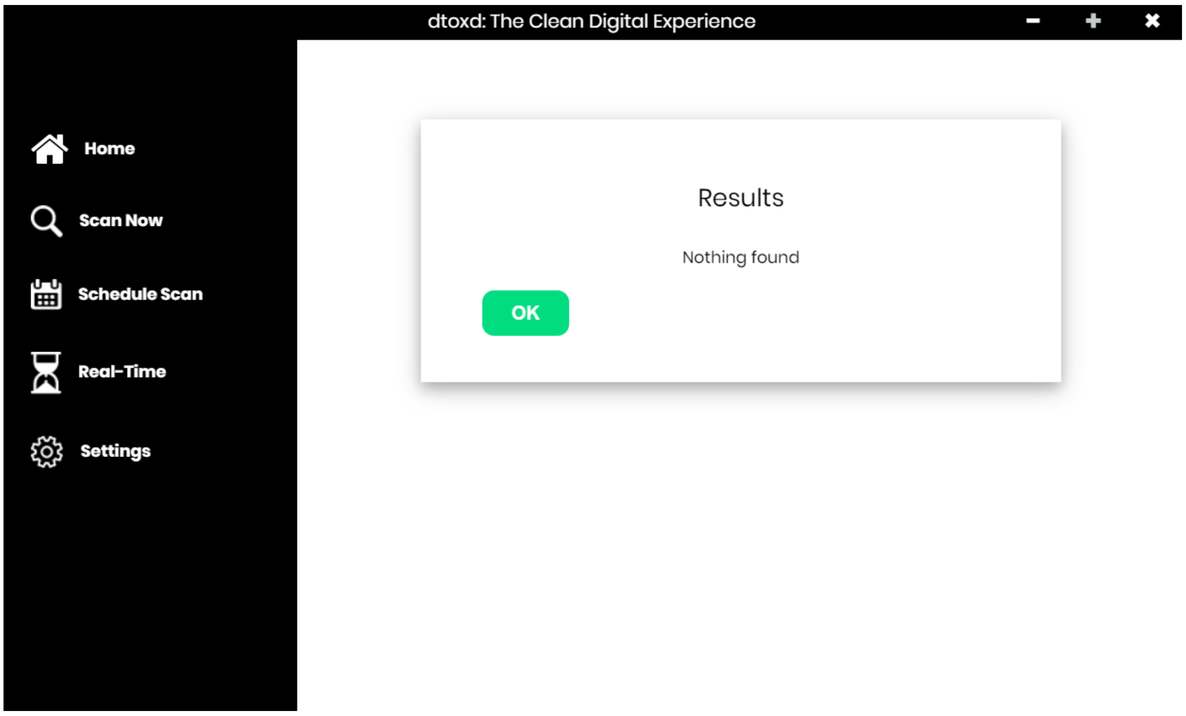


Results

Here's all that you should worry about

- C://Users//britt//Desktop//lol//Anuj_new//work\1.jpg
- C://Users//britt//Desktop//lol//Anuj_new//work\10.jpg
- C://Users//britt//Desktop//lol//Anuj_new//work\12.jpg
- C://Users//britt//Desktop//lol//Anuj_new//work\13.jpg
- C://Users//britt//Desktop//lol//Anuj_new//work\3.jpg
- C://Users//britt//Desktop//lol//Anuj_new//work\5.jpg
- C://Users//britt//Desktop//lol//Anuj_new//work\7.jpg
- C://Users//britt//Desktop//lol//Anuj_new//work\8.jpg

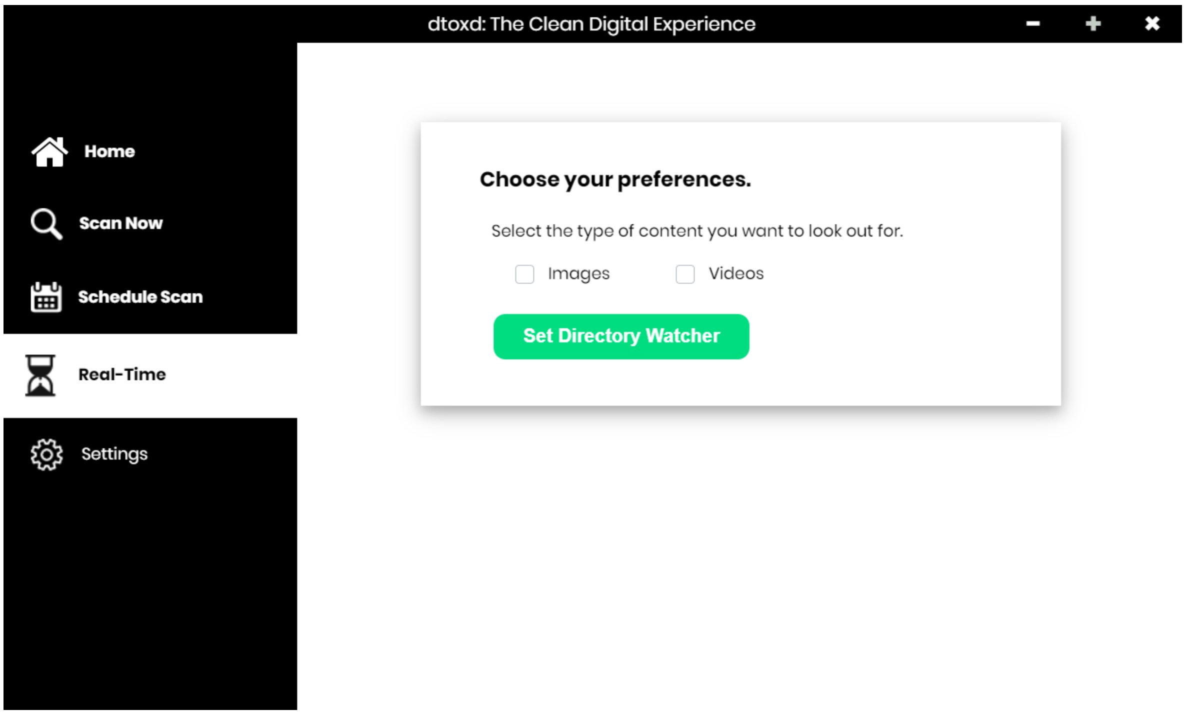
Figure 5.4: Results after the scan is complete and explicit content found



Results

Nothing found

Figure 5.5: Results when no explicit content found

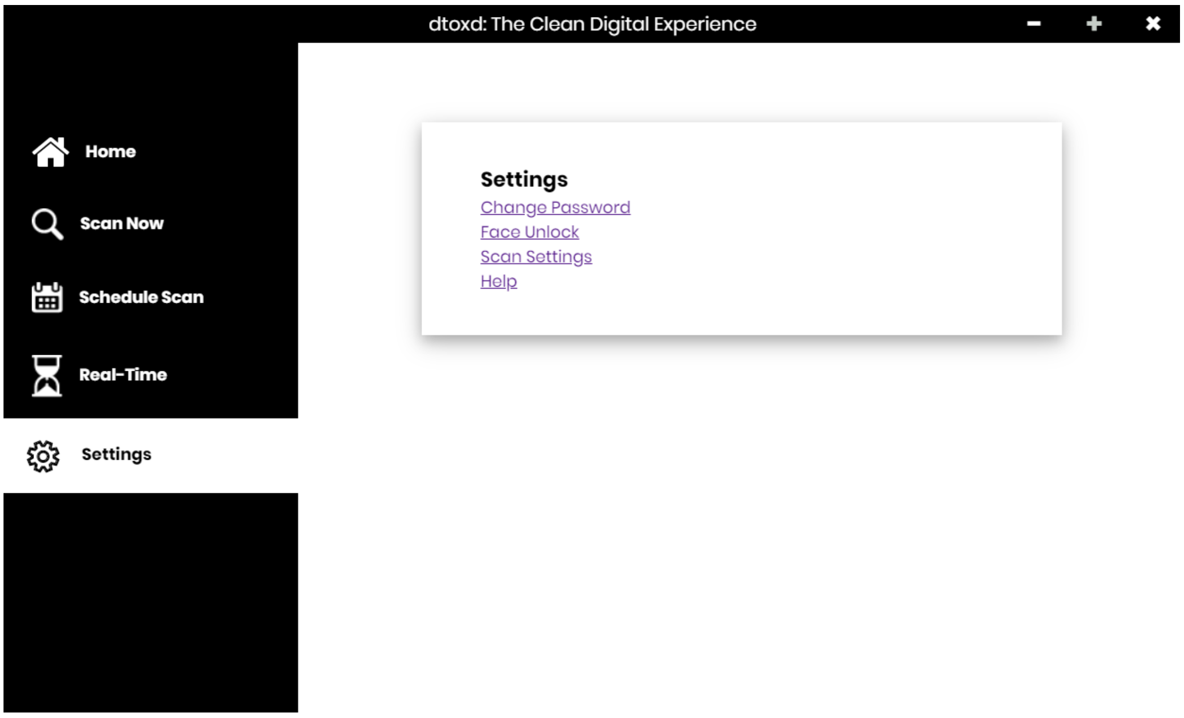


Choose your preferences.

Select the type of content you want to look out for.

☐ Images ☐ Videos

Figure 5.6: Real Time Directory watcher to add more directories to scan

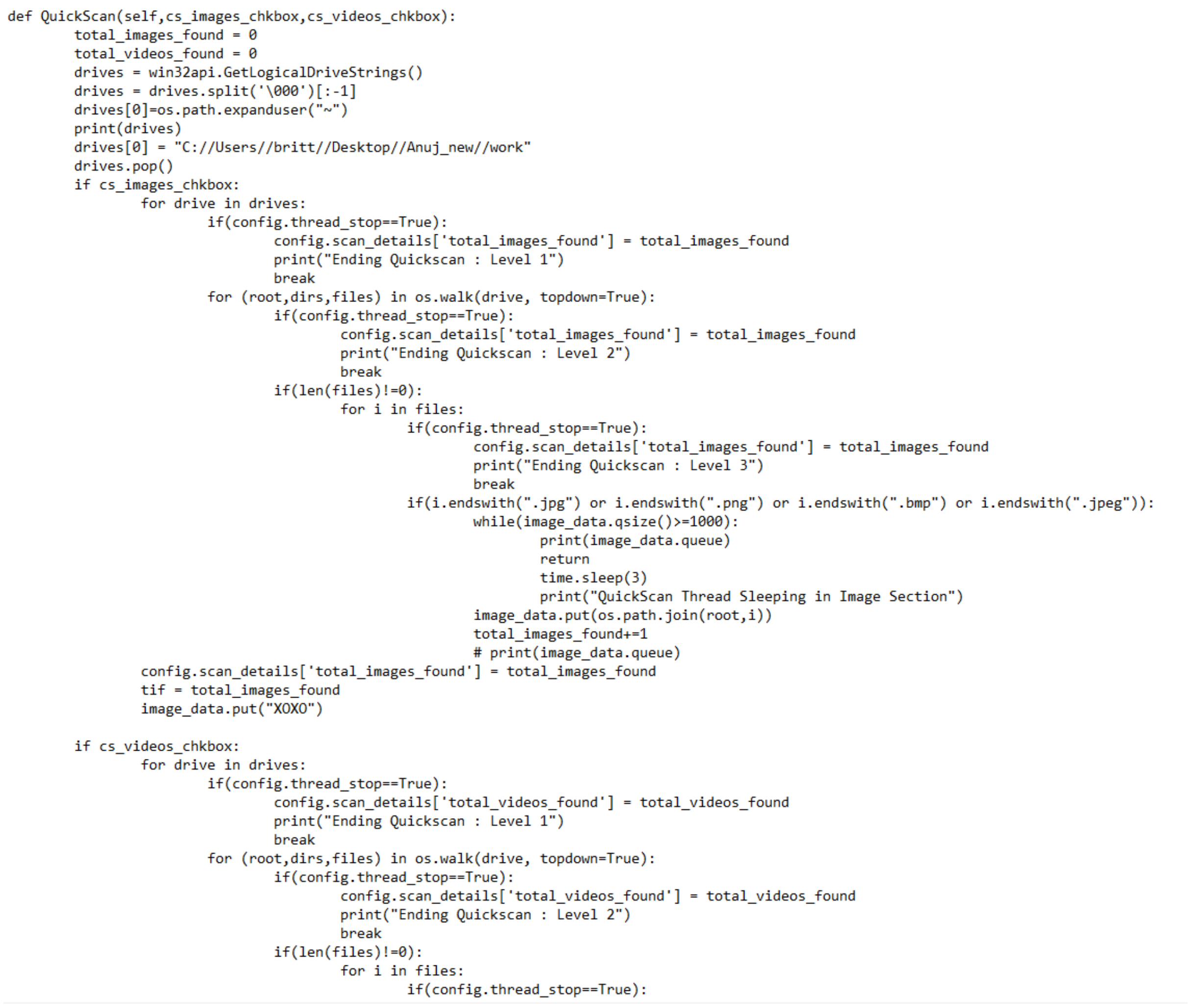


Settings
Change Password
Face Unlock
Scan Settings
Help

Figure 5.7: Settings Menu

5.2 Important Code Snipets

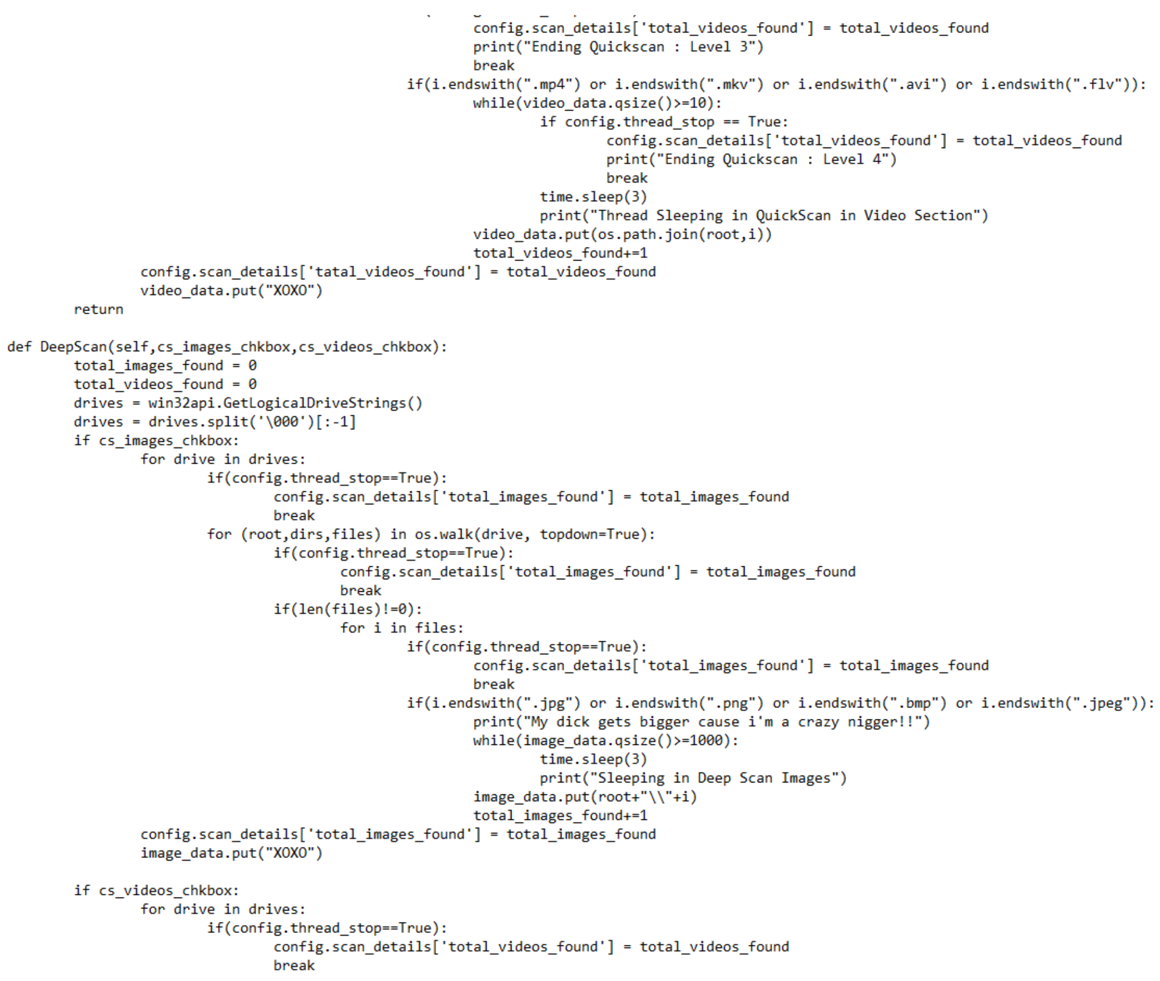
5.2.1 Quick Scan, Deep Scan and Prediction functions for Images and Videos



```
def QuickScan(self,cs_images_chkbox,cs_videos_chkbox):
    total_images_found = 0
    total_videos_found = 0
    drives = win32api.GetLogicalDriveStrings()
    drives = drives.split('\000')[:-1]
    drives[0]=os.path.expanduser("~")
    print(drives)
    drives[0] = "C://Users//britt//Desktop//Anuj_new//work"
    drives.pop()
    if cs_images_chkbox:
        for drive in drives:
            if(config.thread_stop==True):
                config.scan_details['total_images_found'] = total_images_found
                print("Ending Quickscan : Level 1")
                break
            for (root,dirs,files) in os.walk(drive, topdown=True):
                if(config.thread_stop==True):
                    config.scan_details['total_images_found'] = total_images_found
                    print("Ending Quickscan : Level 2")
                    break
                if(len(files)!=0):
                    for i in files:
                        if(config.thread_stop==True):
                            config.scan_details['total_images_found'] = total_images_found
                            print("Ending Quickscan : Level 3")
                            break
                        if(i.endswith(".jpg") or i.endswith(".png") or i.endswith(".bmp") or i.endswith(".jpeg")):
                            while(image_data.qsize()>=1000):
                                print(image_data.queue)
                                return
                                time.sleep(3)
                                print("QuickScan Thread Sleeping in Image Section")
                            image_data.put(os.path.join(root,i))
                            total_images_found+=1
                            # print(image_data.queue)
        config.scan_details['total_images_found'] = total_images_found
        tif = total_images_found
        image_data.put("XOXO")

    if cs_videos_chkbox:
        for drive in drives:
            if(config.thread_stop==True):
                config.scan_details['total_videos_found'] = total_videos_found
                print("Ending Quickscan : Level 1")
                break
            for (root,dirs,files) in os.walk(drive, topdown=True):
                if(config.thread_stop==True):
                    config.scan_details['total_videos_found'] = total_videos_found
                    print("Ending Quickscan : Level 2")
                    break
                if(len(files)!=0):
                    for i in files:
                        if(config.thread_stop==True):
```

Figure 5.8: Quick Scan for Images & Videos

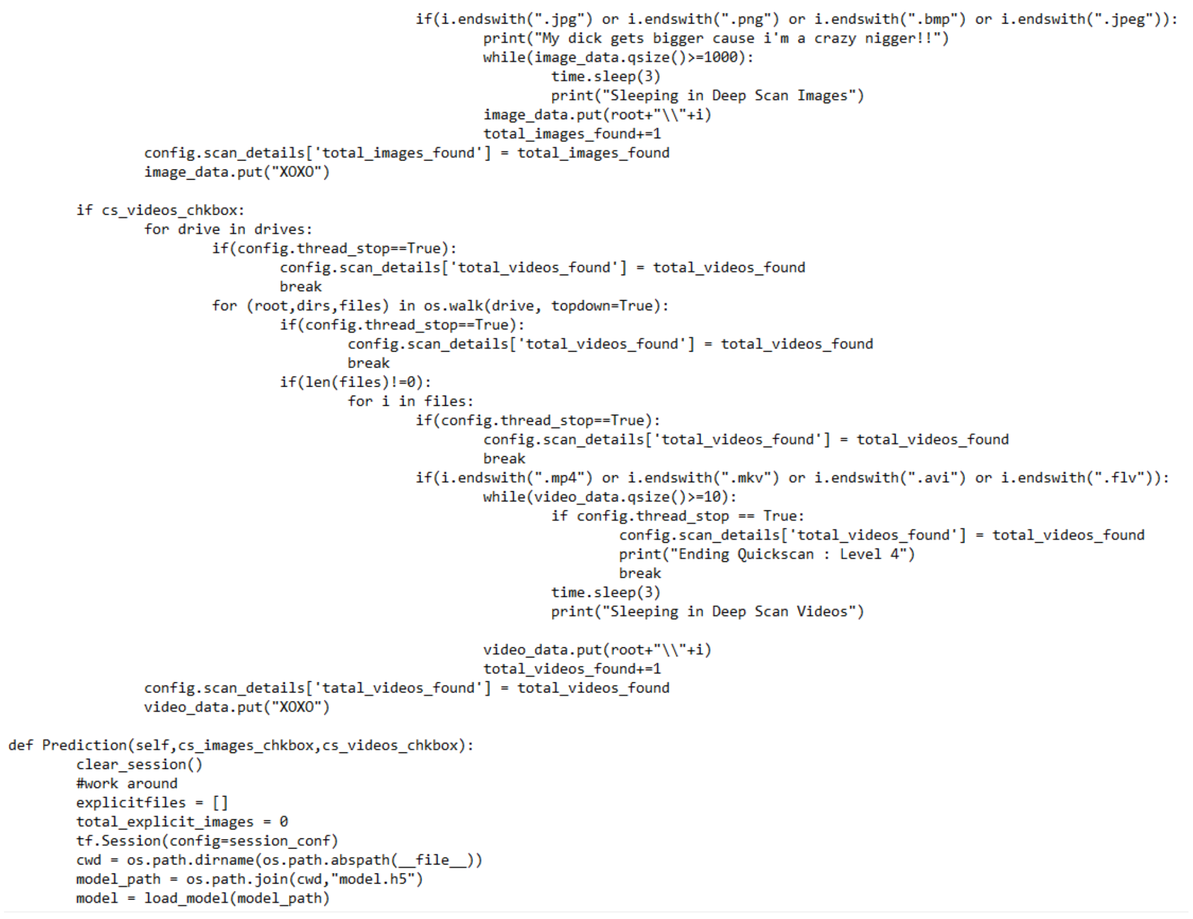


```
                            config.scan_details['total_videos_found'] = total_videos_found
                            print("Ending Quickscan : Level 3")
                            break
                        if(i.endswith(".mp4") or i.endswith(".mkv") or i.endswith(".avi") or i.endswith(".flv")):
                            while(video_data.qsize()>=10):
                                if config.thread_stop == True:
                                    config.scan_details['total_videos_found'] = total_videos_found
                                    print("Ending Quickscan : Level 4")
                                    break
                                time.sleep(3)
                                print("Thread Sleeping in QuickScan in Video Section")
                            video_data.put(os.path.join(root,i))
                            total_videos_found+=1
        config.scan_details['tatal_videos_found'] = total_videos_found
        video_data.put("XOXO")
    return

def DeepScan(self,cs_images_chkbox,cs_videos_chkbox):
    total_images_found = 0
    total_videos_found = 0
    drives = win32api.GetLogicalDriveStrings()
    drives = drives.split('\000')[:-1]
    if cs_images_chkbox:
        for drive in drives:
            if(config.thread_stop==True):
                config.scan_details['total_images_found'] = total_images_found
                break
            for (root,dirs,files) in os.walk(drive, topdown=True):
                if(config.thread_stop==True):
                    config.scan_details['total_images_found'] = total_images_found
                    break
                if(len(files)!=0):
                    for i in files:
                        if(config.thread_stop==True):
                            config.scan_details['total_images_found'] = total_images_found
                            break
                        if(i.endswith(".jpg") or i.endswith(".png") or i.endswith(".bmp") or i.endswith(".jpeg")):
                            print("My dick gets bigger cause i'm a crazy nigger!!")
                            while(image_data.qsize()>=1000):
                                time.sleep(3)
                                print("Sleeping in Deep Scan Images")
                            image_data.put(root+"\\"+i)
                            total_images_found+=1
        config.scan_details['total_images_found'] = total_images_found
        image_data.put("XOXO")

    if cs_videos_chkbox:
        for drive in drives:
            if(config.thread_stop==True):
                config.scan_details['total_videos_found'] = total_videos_found
                break
```

Figure 5.9: Deep Scan for Images & Videos



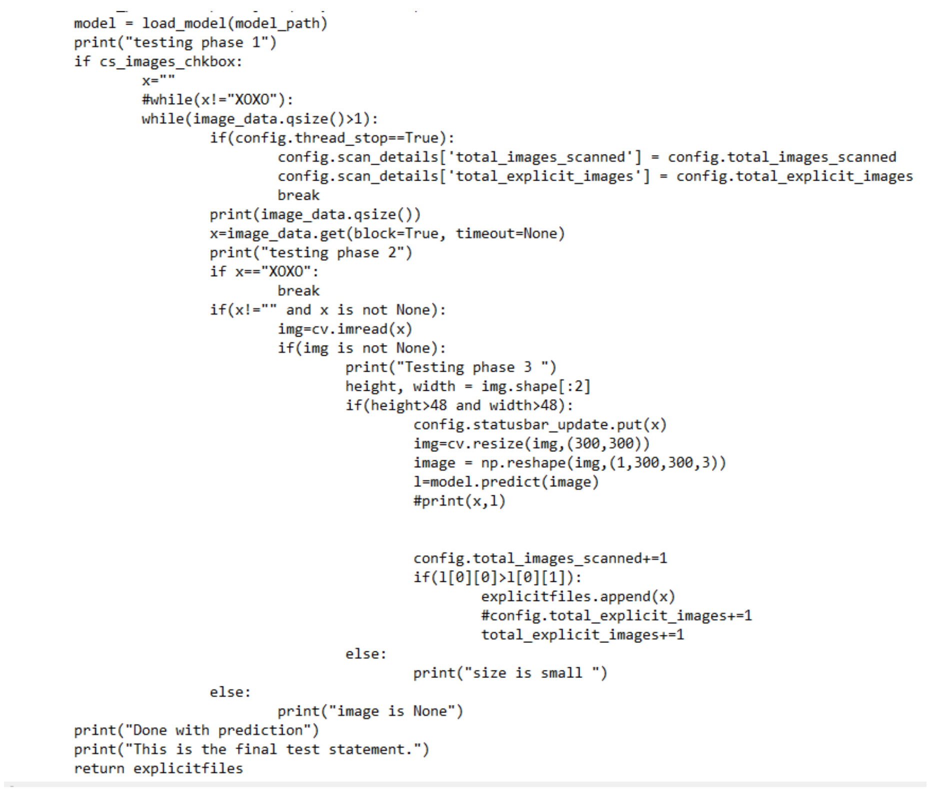
```
                        if(i.endswith(".jpg") or i.endswith(".png") or i.endswith(".bmp") or i.endswith(".jpeg")):
                            print("My dick gets bigger cause i'm a crazy nigger!!")
                            while(image_data.qsize()>=1000):
                                time.sleep(3)
                                print("Sleeping in Deep Scan Images")
                            image_data.put(root+"\\"+i)
                            total_images_found+=1
        config.scan_details['total_images_found'] = total_images_found
        image_data.put("XOXO")

    if cs_videos_chkbox:
        for drive in drives:
            if(config.thread_stop==True):
                config.scan_details['total_videos_found'] = total_videos_found
                break
            for (root,dirs,files) in os.walk(drive, topdown=True):
                if(config.thread_stop==True):
                    config.scan_details['total_videos_found'] = total_videos_found
                    break
                if(len(files)!=0):
                    for i in files:
                        if(config.thread_stop==True):
                            config.scan_details['total_videos_found'] = total_videos_found
                            break
                        if(i.endswith(".mp4") or i.endswith(".mkv") or i.endswith(".avi") or i.endswith(".flv")):
                            while(video_data.qsize()>=10):
                                if config.thread_stop == True:
                                    config.scan_details['total_videos_found'] = total_videos_found
                                    print("Ending Quickscan : Level 4")
                                    break
                                time.sleep(3)
                                print("Sleeping in Deep Scan Videos")

                            video_data.put(root+"\\"+i)
                            total_videos_found+=1
        config.scan_details['tatal_videos_found'] = total_videos_found
        video_data.put("XOXO")

def Prediction(self,cs_images_chkbox,cs_videos_chkbox):
    clear_session()
    #work around
    explicitfiles = []
    total_explicit_images = 0
    tf.Session(config=session_conf)
    cwd = os.path.dirname(os.path.abspath(__file__))
    model_path = os.path.join(cwd,"model.h5")
    model = load_model(model_path)
```

Figure 5.10: Prediction Function



```
    model = load_model(model_path)
    print("testing phase 1")
    if cs_images_chkbox:
        x=""
        #while(x!="XOXO"):
        while(image_data.qsize()>1):
            if(config.thread_stop==True):
                config.scan_details['total_images_scanned'] = config.total_images_scanned
                config.scan_details['total_explicit_images'] = config.total_explicit_images
                break
            print(image_data.qsize())
            x=image_data.get(block=True, timeout=None)
            print("testing phase 2")
            if x=="XOXO":
                break
            if(x!="" and x is not None):
                img=cv.imread(x)
                if(img is not None):
                    print("Testing phase 3 ")
                    height, width = img.shape[:2]
                    if(height>48 and width>48):
                        config.statusbar_update.put(x)
                        img=cv.resize(img,(300,300))
                        image = np.reshape(img,(1,300,300,3))
                        l=model.predict(image)
                        #print(x,l)


                        config.total_images_scanned+=1
                        if(l[0][0]>l[0][1]):
                            explicitfiles.append(x)
                            #config.total_explicit_images+=1
                            total_explicit_images+=1
                    else:
                        print("size is small ")
            else:
                print("image is None")
    print("Done with prediction")
    print("This is the final test statement.")
    return explicitfiles
```

Figure 5.11: Prediction Function

5.3 Testing

Functionality	**Test Cases Executed**	**Test Cases Passes**	**Remark**
Starting python server	10	10	Pass
Running Graphical User Interface	10	10	Pass
Clicking "Home" Page Screen Toggles	13	11	Pass
Clicking Checkbox for images & videos on the "Scan Now" Page	10	10	Pass
Selecting Quick Scan & running scan	20	18	Pass
Selecting Deep Scan & running scan	10	7	Pass
Displaying directories that are being scanned in real time during scan	10	7	Pass
Displaying number of images or videos scanned in real time during scan	10	9	Pass
Displaying number of Explicit images or videos found in real time during scan	10	10	Pass
Aborting scan and displaying results	10	8	Pass
Displaying results after completing scan	13	12	Pass
Adding a directory on the "Real Time" Page	10	8	Pass
Scheduling scan	15	15	Pass
Changing Password on the "Settings" Page	10	8	Pass

Table 5.1: Testing Report

Chapter 6

Conclusion & Future Scope

Real Time Content Moderation(RTCM) uses Deep Learning to censor explicit content from images or videos in real time that shows great potential to be deployed as a stand alone application on to college, school servers. This report presented an intelligent, user friendly, robust system that is capable of censoring explicit content in real time using two deep learning models: Classification and Object Detection. RTCM stands out from the existing softwares mentioned in the report such that it has a very user friendly and modern yet simple User Interface and a state-of-the-art machine learning model. The system in it's current state is built upon the Classification Model where we achieved an accuracy of close to 86%. However, the Object Detection models show promising results and through more data and learning, they can substitute the classification model as the Core Deep Learning module of RTCM. The system is designed in such a way that the user is not burdened with technical jargon and can easily navigate through the app. RTCM intends to become a desktop application where the software can be easily installed on Windows based machines and eventually become cross-platform. The system is designed in such a way that the user data remains secured thus giving users a sense of reliability and security.

References

[1] A. Qamar Bhatti, M. Umer, S. H. Adil, M. Ebrahim, D. Nawaz, and F. Ahmed, “Explicit content detection system: An approach towards a safe and ethical environment,” *Applied Computational Intelligence and Soft Computing*, vol. 2018, 2018.

[2] Y.-C. Lin, H.-W. Tseng, and C.-S. Fuh, “Pornography detection using support vector machine,” in *16th IPPR Conference on Computer Vision, Graphics and Image Processing (CVGIP 2003)*, vol. 19, pp. 123–130, 2003.

[3] H. Zhu, S. Zhou, J. Wang, and Z. Yin, “An algorithm of pornographic image detection,” in *Fourth International Conference on Image and Graphics (ICIG 2007)*, pp. 801–804, IEEE, 2007.

[4] R. Ap-Apid, “An algorithm for nudity detection,” in *5th Philippine Computing Science Congress*, pp. 201–205, 2005.

[5] “Caglar gulcehre. http://deeplearning.net/.,”

[6] C.-T. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. Bradski, K. Olukotun, and A. Y. Ng, “Map-reduce for machine learning on multicore,” in *Advances in neural information processing systems*, pp. 281–288, 2007.

[7] “http://www.image-net.org/,”

[8] L. Tzutalin, “Git code (2015).”

[9] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, “Tensorflow: A system for large-scale machine learning,” in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pp. 265–283, 2016.

[10] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[11] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014.

Acknowledgement

Success of a project like this involving high technical expertise, patience and massive support of guides, is possible when team members work together. We take this opportunity to express our gratitude to those who have been instrumental in the successful completion of this project. We would like to appreciate the constant mentorship and support of our guide **Mr Harshit Rai** from **dtoxd** in our project and aiding us in developing a flair for the field of Computer Vision and Machine Learning. We would also like to show our appreciation to **Ms. Kavita Shelke** for her tremendous support and help, without whom this project would have reached nowhere. We would also like to thank our project coordinator **Mrs. Rakhi Kalantri** for providing us with regular inputs about documentation and project timeline. A big thanks to our HOD **Dr. Lata Ragha** for all the encouragement given to our team. We would also like to thank our principal, **Dr. S. M. Khot**, and our college, **Fr. C. Rodrigues Institute of Technology, Vashi**, for giving us the opportunity and the environment to learn and grow.

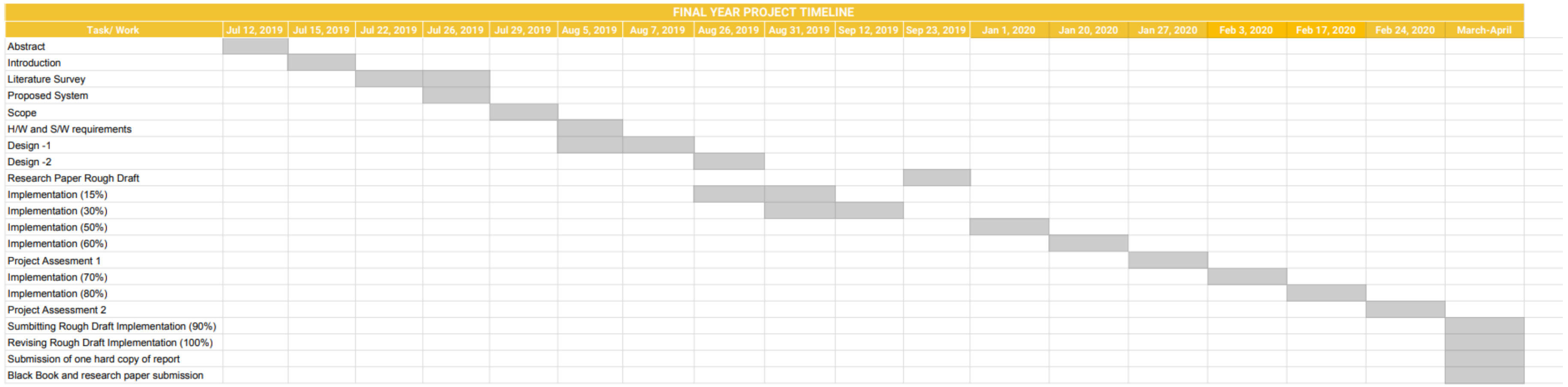
Project Group Members:

1. Cyrus Britto, 101607

2. Anuj Dalvi, 101610

3. Animesh Srivastava, 101655

Appendix A : Timeline Chart



FINAL YEAR PROJECT TIMELINE																		
Task/ Work	Jul 12, 2019	Jul 15, 2019	Jul 22, 2019	Jul 26, 2019	Jul 29, 2019	Aug 5, 2019	Aug 7, 2019	Aug 26, 2019	Aug 31, 2019	Sep 12, 2019	Sep 23, 2019	Jan 1, 2020	Jan 20, 2020	Jan 27, 2020	Feb 3, 2020	Feb 17, 2020	Feb 24, 2020	March-April
Abstract	▨																	
Introduction		▨																
Literature Survey			▨	▨														
Proposed System				▨														
Scope					▨													
H/W and S/W requirements						▨												
Design -1						▨	▨											
Design -2								▨										
Research Paper Rough Draft											▨							
Implementation (15%)								▨	▨									
Implementation (30%)									▨	▨								
Implementation (50%)												▨						
Implementation (60%)													▨					
Project Assesment 1														▨				
Implementation (70%)															▨			
Implementation (80%)																▨		
Project Assessment 2																	▨	
Sumbitting Rough Draft Implementation (90%)																		▨
Revising Rough Draft Implementation (100%)																		▨
Submission of one hard copy of report																		▨
Black Book and research paper submission																		▨

Figure 6.1: Timeline Chart for project

Appendix B : Publication Details

Title: Explicit Content Detection using Faster R-CNN & SSD MobileNet V2

Paper ID: FTP73F3116

Journal: International Research Journal of Engineering and Technology (IRJET)

Volume: 7

Issue: 3

Link: https://www.irjet.net/archives/V7/i3/IRJET-V7I31126.pdf