Global Scientific JOURNALS

GSJ: Volume 12, Issue 3, March 2024, Online: ISSN 2320-9186
www.globalscientificjournal.com

# FACIAL ANALYSIS OF LAWBREAKERS FOR DETECTION AND RECOGNITION USING MACHINE LEARNING

Michelle Rehman, Ifrah Asif, and Waseemullah Nazir

*Michelle Rehman is a student at NED University of Engineering and Technology and is pursuing her bachelor's degree in Computer Science and Information Technology (Email: michelle9rehman@gmail.com)*
*Ifrah Asif is a student at NED University of Engineering and Technology and is pursuing her bachelor's degree in Computer Science and Information Technology (Email: ifrahasif30@gmail.com)*
*Waseemullah Nazir is an Assistant Professor at NED University of Engineering and Technology at the department of Computer Science and Information Technology. He has a research interest in Computer Visions (Email: waseemu@cloud.neduet.edu.pk)*

## KeyWords
Computer Vision, Convolutional Neural Network, Deep Learning, Face Detection, Face Recognition, Firebase, Haar Classifier, Image Processing, Real Time Detection, REACTJS, System Testing

## ABSTRACT
This research aims to offer facial recognition facilities to the police investigation department to enhance citizen security. The project addresses the issue of lawbreakers walking freely along the streets of Karachi, a city in Pakistan. We aim to aid the police in capturing criminals through facial recognition technology. This idea was inspired by such technology already deployed in first-world countries where security has been made more reliable due to the presence of facial analysis systems installed in CCTV cameras along the streets. The project involves the development of a responsive web application, Inspecto, that is backed up by many APIs leveraging the use of facial analysis statistics and detection techniques for detecting and recognizing criminal faces at different crime scenes at the same time. This prototype uses real-time face detection through a webcam and allows users to filter out, access, and update profiles of existing and new lawbreakers. The project utilizes deep learning algorithms for facial recognition as a large dataset was fed to the Machine Learning algorithm to enhance the Artificial Intelligence deployed to detect faces. Additionally, this has shown promising results.

## 1. INTRODUCTION
The main reason for choosing this topic as our final year project is that we want to help the Sindh police in detecting criminal activities with more efficiency. This would save a lot of hassle and time and the face of the criminal would also be stored in the project's database hence, whenever that criminal is spotted again by any of the CCTV cameras, if not caught the first time, the police will know about his/ her whereabouts and cuff them. Detection and recognition can be done in two ways; either by uploading a screenshot taken from a video tape, or by live detection.

This project was designed to detect and recognize criminal faces in their respective crime scenes. Face detection can be regarded as a generalized version of face localization **[1]**. An application, Inspecto, was made which is very user-friendly as every icon and button is self-explanatory. It provides a platform for the users to filter out the criminals they want to know about and access their profile details. They can also update criminal profiles. Users of the app will first have to register themselves in a controlled Intranet to ensure security and will then be able to use the project. Every employee in this app will be given full control in performing CRUD (Create, Read, Update, and Delete) applications regarding criminal details but cannot change their own information with which they have registered themselves with – for security purposes.

### *1.1 Goals and Objectectives*
Currently, there are numerous criminals walking freely in the streets of Karachi mostly because our police have not been able to capture them on time. Criminals also go undetected because witnesses can't really identify them with full accuracy. With this application, the face of the criminal can be picked up by a CCTV camera attached in that area and the user can then feed the face into the database so whenever that face appears again in any of the cameras during live detection, the police will be notified and will reach the place with much efficiency.

### *1.2 System Context*
This system is built solely for police personnel who need to work fast in catching criminals and making society a better place to live. The type of crimes will be assessed and can further be researched regarding the places at which those crimes were easier to commit. This app can also help the users in monitoring the nature and frequency of certain crimes that can thus, help in predicting dangerous areas and places where most criminal activity might take place.

During this age, everything is solved primarily using the internet. This Web App can prevent our police officers from being on patrol 24/7 to ensure safety of the citizens, now, the safety of the people can be guarded by the police from their office only and thus, reducing labor. These are Covid-19 times, and the health of our police force is also of much importance than any other citizen, even though most of the people are now vaccinated, the new variants are very deadly so it will be in the favor of Inspecto's app users to patrol the city through their monitors in a safe place from an even safer distance than to patrol the city manually at all times.

### *1.3 Theoretical Background for Project*
In recent times, Karachi has been ranked as the fourth worst city to live in the whole world **[2]** because of how unsafe the city is. The main idea behind this project is detecting and recognizing criminals, alerting the police personnel, getting the criminals caught and thus, making Karachi a better place to live in. By doing so, we can promote a better picture of our city that'll allow visitors to explore Karachi too, which will in turn promote our economy.

### *1.4 Technology and Tools Used in Project*
The technologies used for this project are included below in their respective sections:

#### *1.5.1 Front-End Technologies*
The framework for JavaScript, REACTJS, was used as our developmental platform. HTML was employed to set up a basic index due to its ease in usability. The User Interface (UI) was enhanced using CSS (Cascading Style Sheets). Bootstrap and Semantic UI were used to give well-designed routing capabilities to the application.

#### *1.5.2 Back-End Technologies and Local Server*
ReactJS works best with Node.js since they both use JavaScript as their primary coding language. So Node.js seemed to be the perfect option to host and run our ReactJS application. Our project runs on localhost:3000.

#### *1.5.3 Database*
Google's Firebase was used as a cloud-based database due to its ability to update records in real time. Firebase's Firestore was utilized for storing criminal records, registered user information, and contacting messages between users and developers. Firebase's Storage was used to store pictures of lawbreakers and lastly, Firebase's Authentication was employed to authenticate user registration.

## 2. USER INTERACTION
To make the UX (User Experience) as easy as possible, the application promotes a single-user-one-platform approach. This means that each user on the Intranet can access the app in the same manner as the rest whilst connected to the server. Hence a single entity can be seen as either: 1) Unregistered User or 2) Registered User.

### *2.1 Unregistered User*
Unregistered users can be defined as those users that haven't given their necessary credentials to access the web application. To differentiate between legitimate and non-legitimate users, there is a constraint that unregistered users cannot access the application completely because of security purposes. This is done to keep confidential data private.

#### *2.1.1 Use Case for Unregistered Users*
This application does not support any functionality to be associated with any unregistered user so if a user is needed to be added into the system, they will first have to give their necessary credentials (simmered down to only the basics for the sake of simplicity) to be recognized as a member of the team. A use case diagram shows how a system interacts with external entities **[3]**. The following use case is used to describe this scenario:

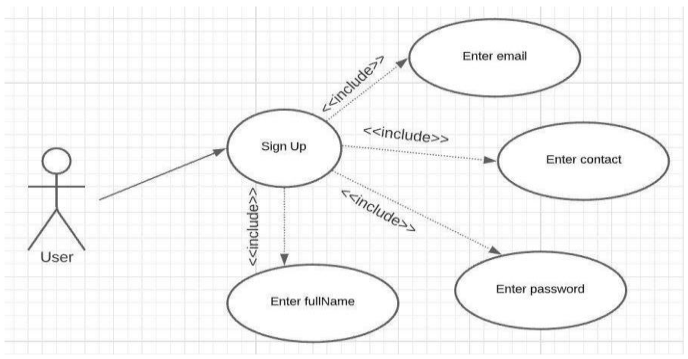


**Figure 1. Use Case When Users Register Themselves**

*2.2 Registered User*

Users that have given their credentials like their name, contact number, email (also taken as username) and have set a password following our standard password acceptance scheme, during registration can now successfully login because their data is now within the authentication database and hence, are now allowed to use the application. These users are termed as Registered Users.

*2.2.1 Use Cases for Registered Users*

Registered users can access the whole application. From logging-in to entering criminal records and updating/ deleting it, to also uploading images for detection and recognition. Registered users can also send queries to the technical team/ developers. The following use-cases can be derived from these scenarios:

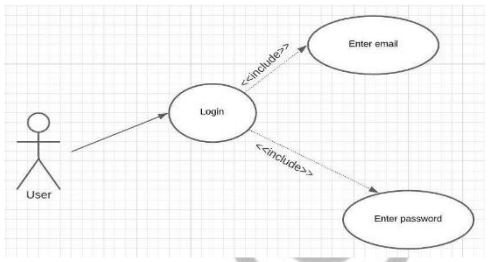


**Figure 2. Use Case When User Logs In**

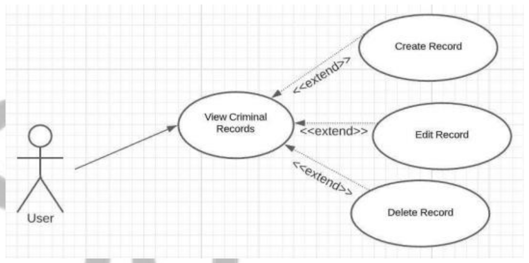


**Figure 3. Use Case When User Views Records**

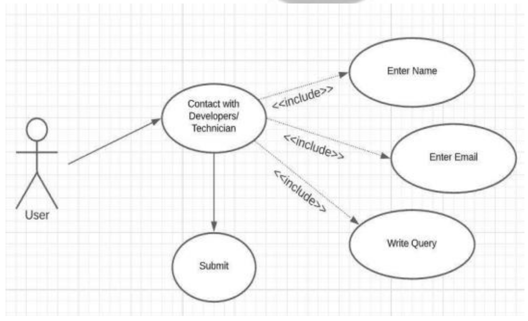


**Figure 4. Use Case When User Sends a Query**

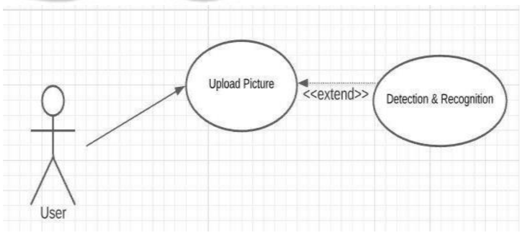


**Figure 5. Use Case When User Uploads Picture (to the Database) for Detection**

## 3. ARCHITECTURE AND DATA DESCRIPTION

The application contains REACTJS as it gives a more effective and light weight document object model **[4]**. This app contains separate pages for criminal face detection and recognition. There are two ways of detection either through image input or through real-time detection. It also contains multiple forms that are connected with Firebase. All the information and data regarding criminal activity is stored in Firestore and Firebase storage.

*3.1 System Architecture*

The application is divided into three subsystems:

i. **Data Layer:** Data layer contains the Firebase database, which stores the data of the criminals including, their description and images. The database also contains user logs, which store users and admins' information along with their required credentials (email, password etc.).

ii. **Application Layer:** This layer contains the face-api.js, which is a JavaScript library, built on top of tensorflow.js core framework. This API allows face detection with the help of CNN (Convolutional Neural Network), a deep learning algorithm.

iii. **Presentation Layer:** CSS and Bootstrap are used to make the application more user-friendly. This layer consists of all the pages present within the application (Login-in/ Sign-up, Dashboard, Records page, Adding/ Updating/ Deleting Records page, Contact Us page, Mode of Detection page, and About Us page). Buttons and icons are used to make the aesthetics more understandable and easier to approach.

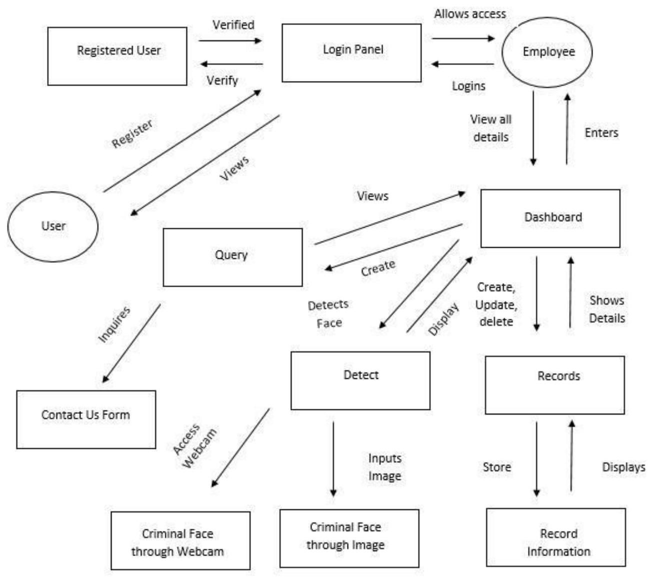


**Figure 6. Architecture Model of Inspecto**

*3.2 Data Description*

Data is stored in Google's Firebase which is a NoSQL, document-oriented database hence, there are no tables or rows. Instead, data is stored in multiple documents which are organized into collections. The Inspecto database contains complete information of the criminals, registered users, and the chats between the members. The following Entity Relationship Diagram (ERD) can be used to understand the layout of data for this project:

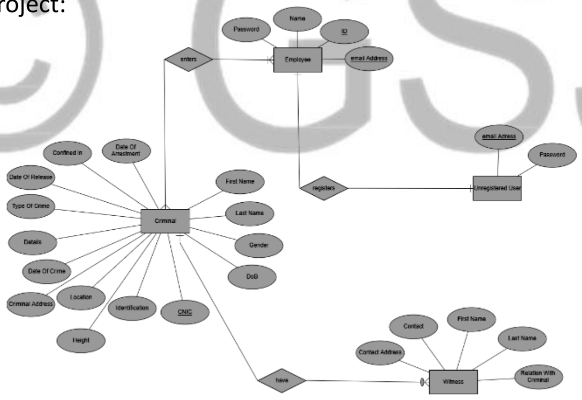


**Figure 7. Entity Relationship Diagram (ERD) Showing Data Layout**

*3.3 System Interface Design*

This section provides an overall idea of how the application can be accessed by users. Since the target audience are the police personnel, the users are required to be connected to the system through the central Intranet. There are no external machine interfaces. This application can be accessed by a PC or a laptop. The main entity of Inspecto is the 'User' which is further classified into 'Unregistered User' and 'Registered User' as discussed previously.

## 4. BEHAVIORAL MODEL AND DESCRIPTION

This application facilitates its user to manage all the criminal records from one place and always perform detection and recognition automatically. Although this prototype uses a webcam functionality, if programmed in an actual CCTV camera, it could be discreetly placed outside vulnerable places like jewelry shops, convenience stores, etc. to detect any suspicious activity. The criminals can be identified, and the authorities can be notified immediately. The database stores the recordings automatically, but records are to be updated manually. In another way, the footage can also be helpful for the investigation department in case of an aftermath of a big accident.

*4.1 Events or Interrupts*

Following events/ interrupts occur throughout the system:

- When a user creates an account through Sign-in form by providing right credentials.
- When a user logs in through the Login form with valid credentials.
- When the user provides valid credentials to access the dashboard.
- When user performs CRUD operations on records.
- When user searches up a record in the database.
- When a user fills up the 'Contact Us' form to send their query and clicks the submit button.
- When the user performs face detection either using an image or through live detection.
- When a user chats with others on the team.
- When user clicks on the logout button from the admin panel.

### *4.2 States*

Events return states of the application. In each state, explicit interpretations of all possible inputs are made **[5]**. The following State Transition Diagram summarizes the states that are resulted after each interrupt:

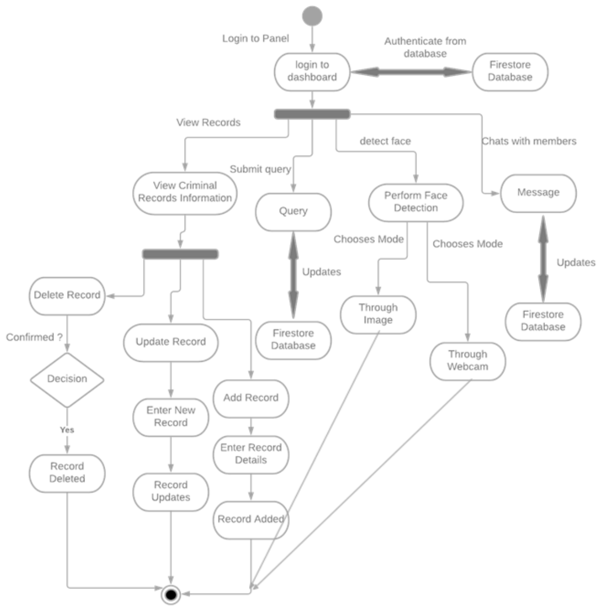


**Figure 8. State Transition Diagram for States Following Certain Events**

### *4.3 Control Specifications*

In this project, the control specifications of dashboard would be managed by employees and all the backend work like the authorization and image recognition would be validated by the database maintained on Google's Cloud Firebase.

## 5. TEST PLAN

This part gives a detailed explanation of the testing plans and strategies that we employed to test this project. The purpose of this phase was to ensure the quality of our software. Efforts were made to ensure that the application becomes a reliable, durable, and scalable product within the estimated budget and time.

### *5.1 System Test and Procedure*

Following tests were performed so we could check whether:

i. user can properly register themselves in the system,
ii. user can place queries,
iii. the Machine Learning algorithm and face-api.js detected the face properly,
iv. the algorithm can be altered without any error,
v. the accuracy level of face-api.js for face detection,
vi. verification performance over a set of images taken under different lighting conditions is acceptable,
vii. performance time was up-to-mark.

### *5.2 Testing Strategies*

Our testing strategies swirled around the face detection functionalities present in the CNN deep learning algorithm by using face-api.js since CNN algorithm correctly identifies tiny objects **[6]**, faces, in our case. Complications were faced due to differences in image resolutions. It was found that lowresolution images, 144p, were not being supported by this algorithm, but any resolution above

144p (360p and above) was accepted and enhanced the detection and recognition procedures. We performed various testing techniques throughout the development process and at each level to ensure maximum quality.

### *5.2.1 Testing Different Algorithms for Live Detection*

To confirm the efficiency of our chosen CNN model, we compared it with two other detection algorithms: PCA (Principal Component Analysis) and Haar-Cascade. PCA is considered a traditional algorithm whereas both CNN and Haar are considered more modern deep learning algorithms since their threshold values enhance detections **[7]**. The three algorithms were compared to see whether faces were detected and recognized in the following settings:

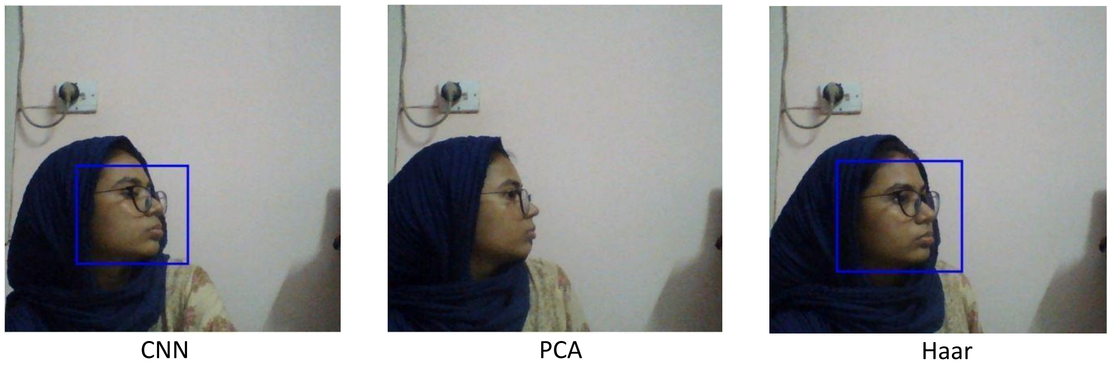


**Figure 9. Facial Detection During Face Movements and at Different Angles**

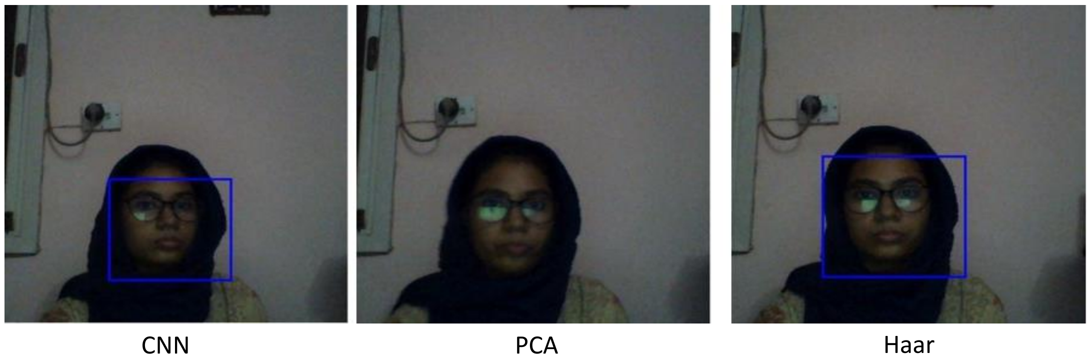


**Figure 10. Facial Detection During Dim Lightings**

**Table 1: Results of Different Machine Learning Algorithms in Various Settings**

| | **CNN** | **PCA** | **Haar** |
|---|---|---|---|
| **Face Detected at Different Angles?** | Yes | No | Yes |
| **Face Detected During Head Movements?** | Yes | No | Yes |
| **Face Detected in Dim Lighting?** | Yes | No | Yes |
| **Tiny Face Detection?** | Yes | No | Yes |

It was concluded that PCA gave the worst results in all the tests by inducing a lot of false positives. Haar and CNN had very similar results, but CNN was able to identify faces in pictures and live videos accurately with a faster rate of detection. Hence, for general computer vision problems, CNN model was preferred since it worked best with occlusion, quick head movements, side angles of faces, dim lightings, and tiny faces.

### *5.2.2 Unit Testing*

A unit is basically a component, function or group of functions performing any specific action. For unit testing, we tested the source code in single chunks in isolation from the remaining system. This was performed for every functionality present within each page as follows:

I. We ensured that the login functionalities of login panel and User Registration are working correctly.
II. We tested the CRUD operations of criminal record by dividing them into different pages and in separate small modules.
III. We tested that if the user can search the criminal records according to his/her needs correctly so, we separated dif-

ferent searches from one another and then tested them.

IV. We checked that the employee could post a query and whether it is updated in the Firebase database or not.
V. We checked whether each button and icon on every page is performing its desired task correctly or not.
VI. We checked whether the employees could send and receive messages to and from other members.
VII. We checked whether the employee could view and edit their profile information.
VIII. We checked whether the Firebase database followed the correct customized rules that we set up manually.
IX. We tested multiple images of the same person to ensure that it gave close accuracy.
X. We tested the live real-time detection at different locations and in different lightning.
XI. We tested the performance and accuracy rate of CNN algorithm present within the Tiny Face Detector Model of face-api.js and compared it to other models and algorithms.

*5.2.3 Integration Testing*

After Unit Testing, all the components were integrated, and Integration Testing was performed. The three approaches to Integration Testing include the following:

1. Top-Down Approach
2. Bottom-Up Approach
3. Big Bang Approach

We employed the Bottom-Up Approach since we worked on low-level units first, tested them, and then integrated them into the system. All the testing was done manually. After integrating all the components, an extra testing technique was employed called **'Black Box Testing'** which is a method that designs test cases based on the information from the specifications **[8]**. We employed the following procedures to ensure zero code-breaks:

1. We combined the dashboard functionalities and then checked whether the employee can easily login to the system and can perform CRUD operations on the cloud Firebase.
2. We ensured that the employee could view all the criminal records retrieved from Firebase and can log out correctly from the dashboard.
3. We combined all the front-end pages in the login panel which facilitates the users so that he/she can make an account in our database and can login anytime he/she wants.
4. We integrated all the required components to check if the user could perform face detection and recognition successfully.

*5.2.4 Validation Testing*

After the completion of integration testing and assembling all the units together, final testing of the system begins. To ensure that the system fulfills the user/client requirements, Validation Testing is done. After the validation testing there is a possibility of one of the two outcomes: all the requirements are fulfilled and approved by the user/client, or an insufficient list is created which ensures the presence of bugs in the system.

For Validation Testing we approached a few software developers and people who worked in crime and investigation agencies and let them use Inspecto from scratch without any guidance. They provided us with some feedback, and we worked on it. Finally, the results were positive. They used it easily and hence approved our application. We validated our system by testing whether all the links, buttons, and functionalities were working as expected or not. We validated that the unregistered user and employee of Inspecto can enter correct credentials in the input fields and when they are filled incorrectly, they are refused to login. We also validated whether our APIs gave correct results or not and are fetching the data correctly from the database.

*5.2.5 High-Order Testing (System Testing)*

We performed high-order testing of the overall website after it went through unit and integration testing. After combining all the modules and every page on front-end, we tested the complete website. Following are the different types of testing that were done to test our system:

1. **Performance Testing:** Performance testing was done on our system to check the efficiency and speed of the system. As this project is developed on REACTJS, it is efficient and fast with respect to processing time. We tried to make it as user friendly as possible by making sure that the code is minimal and intelligently written so that it does not take much time to process the user-generated requests.
2. **Usability Testing:** We have performed usability testing on Inspecto by making sure that the application is extremely simple and easy to use. The interface of all the connectors is kept simple so it can be used by everyone. We also en-

sured that the app provided users with all the necessary features.

3. **Alpha Testing:** This is a type of software testing performed to identify bugs before releasing the product **[9]**. Alpha testing was performed by the group members in a working environment since we're the ones who created this system. We tested the entire system according to our requirements and found that the system performed all the required functionalities.
4. **Security Testing:** We wanted to prevent any unauthorized access to this app. Online password attacks are very common and can break a system easily if the security is not properly applied. To ensure the security of Inspecto, we used Firebase's authentication service. Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to an app. It supports authentication using passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter, and more. The credentials are then passed as OAuth token to the Firebase Authentication SDK which will then verify those credentials and return a response to the client. We tested it to make our application safer and more secure.
5. **Stress Testing:** We tried to make sure the application worked efficiently if the load increases with time. As the data come from different machines and fetched though different APIs on the backend so, it was necessary to test if whenever the load increased, the data that was being fetched was displaying appropriately or not. In addition, if the system can bear the load or not. So, to check this phenomenon, we made multiple user requests at the same time and accessed the database simultaneously. In the end, the application performed its tasks correctly.

## *5.3 Test Metrics*

To see how easily and quickly a user can interact with the application, the two metrics that we focused on to check the quality of Inspecto were 'Response Time' and 'Accuracy'.

### *5.3.1 Response Time*

Response time was treated as an important metric to check if the data was being displayed on the front-end quickly or not and whether the users can get their desired output quickly or not. So, if the response is lower, the system is efficient. In our scenario, that was the case. For face detection and recognition through image, the response is quite fast (around 0.5 seconds for CNN) as compared to previous traditional face recognition algorithms like PCA algorithm. Whereas for webcam video capture, it took around 0.8 seconds. Hence making it quite effective and better than other traditional algorithms.

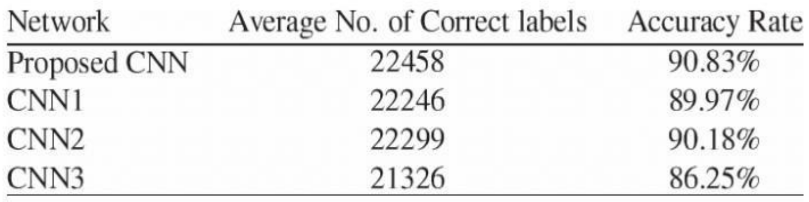
### *5.3.2 Accuracy*

Accuracy was also considered as an important test metric in our testing procedure. This includes the accuracy of data and major functionalities. The data that is being fetched from cloud Firebase database should be correct. Moreover, the functionalities that are related to that data should be accurate. We made sure that the accuracy of the whole system was tested in a timely manner. Informally, accuracy is the fraction of predictions our model got right **[10]**. Formally, accuracy has the following definition:

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions}$$

The accuracy of this project was determined by utilizing various sample sizes of the data we had. The result of CNN model working with different ranges of sets is summarized below:

**Table 2: Finding Accuracy Rate by Comparing Various Sample Sizes with the Same CNN Model**



| Network | Average No. of Correct labels | Accuracy Rate |
|---|---|---|
| Proposed CNN | 22458 | 90.83% |
| CNN1 | 22246 | 89.97% |
| CNN2 | 22299 | 90.18% |
| CNN3 | 21326 | 86.25% |

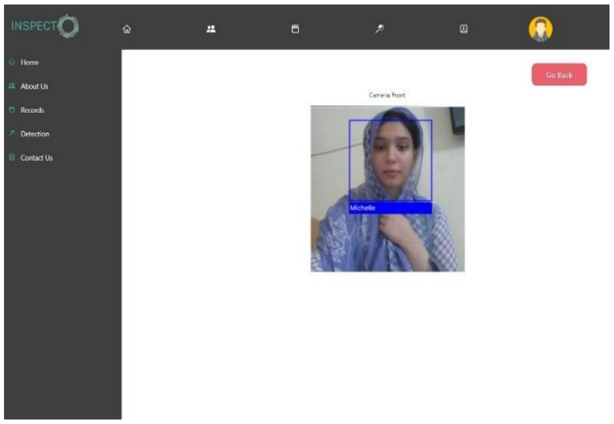


Figure 11. Real Time Live Video Face Detection

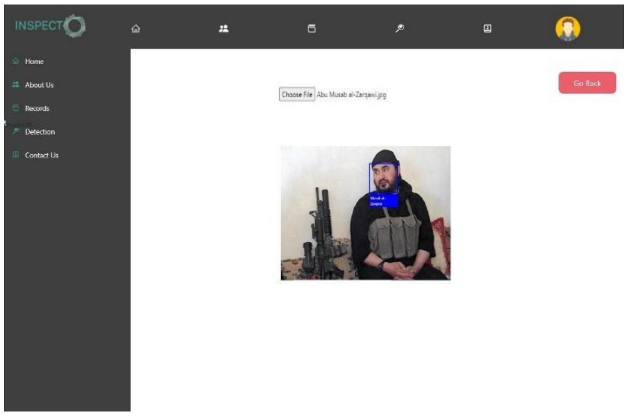


Figure 12. Instant Face Detection Upon Image Upload

Hence, it was noted that the larger the training data set, the greater the accuracy rate of the Machine Learning algorithm, CNN. The final accuracy came out to be 0.9083 or 90.83%. This means that our model is doing a great job at identifying faces.

### 5.4 *Testing Tools and Environment*

We mainly performed manual testing to test our application during the testing process. We did not use any automated tools and relied on our manual testing skills. We applied different input conditions to different fields to check the security and efficiency of the system. We verified the outputs of the smaller units first and then integrated the components to check them separately one after the other and in the end, we combined all the integrated components to form the whole system and tested every functionality. We also used a tool to test the face APIs, which is known as Postman. APIs were tested in isolation to check that every function is being performed correctly. Then, when each API was validated with this tool, we integrated it with front-end. The resources that were used as the testing environment included our laptops. We also tested faces at different locations and distances, in different lightning and angles.

## 6. ENHANCEMENTS AND RECOMMENDATIONS

Since this is a prototype and only entails the surface-level requirements for a facial analysis project, further enhancements could be made to turn this into a full-fledged working model:

### 6.1 *Attachment of CCTV Cameras*

This prototype used webcam functionality as a make-shift street CCTV camera. The code has already been made and is working perfectly. The only recommendation is to program the code into a system of CCTV cameras so as to get the real deal of this project.

### 6.2 *Chatbot Feature*

By scaling up the scope, employees will have more convenience if there is online assistance like a real time chatbot. The chatbot can be deployed to assist the employees and provide information and guidance about some frequently asked queries. They can also provide their concerns related to any functionality.

## 7. CONCLUSION

In this report, we proposed a CNN based face Recognition model for detecting and recognizing criminal faces. An almost up and running application that is made to aid our police in rounding up criminal activities and keeping a fully functional database for each record. After comparing with other state-of-the art algorithms and models, we concluded that CNN with Tiny Face Detector Model outperformed all the other traditional methods in facial recognition. Hence providing us with practical solutions for the problem of identifying criminals at vulnerable places ensuring protection and security for the citizens of Karachi.

## 8. ACKNOWLEDGEMENT

First of all, we are thankful to Almighty Allah who gave us the strength and ability to think, work and deliver what we were assigned to do. Secondly, we are grateful to our internal supervisor Dr. Waseemullah Nazir who guided us in this project. We also acknowledge our teachers who guided, taught, and helped us during our study period. We would also like to thank all departmental staff and university staff, who assisted us during our stay at the university.

## 9. REFERENCES

[1] S. K. Rath and S. S. Rautaray, "A survey on face detection and recognition techniques in different application domain," International Journal of Modern Education and Computer Science, pp 32- 35, Jul. 2014.

[2] Dawn News, "Karachi ranked among worst cities of the world to live in," Dawn News, Aug. 2017. https://www.dawn.com/news/1352353

[3] A.Y. Aleryani, "Comparative study between data flow diagram and use case diagram," International Journal of Scientific and Research Publications, vol. 6, no. 3, pp 124 -127, Mar. 2016.

[4] A. Bhalla, S. Garg, and P. Singh, "Present day web - development using reactjs," International Research Journal of Engineering and Technology, vol. 7, no. 5, pp 1154 - 1157, May 2020.

[5] R. J. K. Jacob, "A state transition diagram language for visual programming," Computer, vol. 18, no. 8, pp 51 -59, Aug. 1985.

[6] T. Akash, T. V. K. Ajay, K. D. Tarun, and J. K. Selva, "Real time object detection using cnn," International Journal of Engineering and Technology, vol. 7, no. 24, pp 33 - 36, Feb. 2018.

[7] K. M. Gopi and A. Srinivasulu, "Face detection system on adaboost algorithm using haar classifiers," International Journal of Modern Engineering and Research, vol. 2, no. 5, pp 3556 - 3560, Oct. 2012.

[8] S. Nidhra and J. Dondeti, "Black box and white box testing techniques -a literature review," International Journal of Embedded Systems and Applications, vol. 2, no. 2, pp 29 -50, Jun. 2012.

[9] "Alpha testing | software testing," GeeksforGeeks, Apr. 2019. https://www.geeksforgeeks.org/alpha-testing-software-testing/

[10] Google, "Classification: accuracy | machine learning crash course," Google Developers, 2019. https://developers.google.com/machine-learning/crash-course/classification/accuracy