PROJECT INTERNSHIP REPORT

Dated 26-06-2024

Project Details

Project Name	Technology
Identity Verification at Onboarding	Frontend : HTML, CSS, Javascript Backend: Python (FastAPI)

Project Description :

This project aims to revolutionise identity verification processes by integrating advanced deep learning and computer vision technologies. Through real-time facial recognition, eye detection, and automated document comparison, we streamline customer onboarding while ensuring compliance with KYC regulations. This innovative solution not only enhances security and accuracy but also prioritises user experience by minimising manual intervention and optimising verification efficiency.

Prepared by	Harsh Kumar Jain
Project Coordinator	Visalakshi M R Varshini Valli K
Mentored by	Pranov Kumaran
Reviewed by	Arun R S

Table of Contents

1. ABSTRACT 3
2. INTRODUCTION 4
3. CURRENT STATE AND CONTEXT 6
4. MOTIVATION 8
5. REQUIREMENTS 9
    5.1 Functional Requirements 9
    5.2 Non-Functional Requirements 10
    5.3 Technical Requirements 10
        5.3.1 Hardware Requirements 10
        5.3.2 Software Requirements 10
6. POTENTIAL SOLUTIONS 12
    6.1 Using cv2 Module in JavaScript with Backend API Integration 12
    6.2 Native JavaScript Webcam Access with Backend API Integration 13
7. PROPOSED SOLUTION 14
    7.1 Modules and Steps 14
    7.2 Final Workflow 18
8. RESULTS AND DISCUSSIONS 21
9. FUTURE ENHANCEMENTS 28
10. REFERENCES 29

ABSTRACT

The onboarding process for new customers often grapples with a delicate balance: ensuring robust identity verification while delivering a smooth user experience. Traditional methods which relied on manual document checks were susceptible to human error and delays (averaging at least 3 days). This can frustrate users and hinder business efficiency.

This project proposes a groundbreaking solution that transcends these limitations. We have developed a system that leverages cutting-edge deep learning technology for a streamlined onboarding experience. The system facilitates automatic selfie capture, verifies user liveness, and compares the captured selfie against government-issued IDs (e.g., passport, driver's licence), eliminating the need for manual document checks. This innovative approach is designed to significantly reduce onboarding time by 90%, minimise errors, and comply with industry regulations such as Know Your Customer (KYC).

INTRODUCTION

In the fast-paced digital landscape, the imperative to onboard customers swiftly and securely is more pressing than ever. However, the traditional methods of identity verification, which relies on manual document checks, often fall short due to inherent human error and lengthy processing times.

This report explores an innovative approach developed as part of our project, aimed at overcoming these challenges through advanced technology integration. Our solution leverages FastAPI for robust backend operations and integrates a WebSocket server for real-time communication. This dual-component system is designed to streamline identity verification during customer onboarding through a sequence of intelligent processes:

1. **Face Detection:** Utilising computer vision techniques to detect and validate the presence of a user's face.

2. **Eye Detection and Blink Monitoring:** Employing advanced algorithms to monitor eye movements and detect natural blinks, ensuring the user's active presence.

3. **Stable Conformation and Automated Capture**: Verifying user stability over a defined period (e.g., 2 seconds) before automatically capturing a photo for further verification steps.

4. **Face Verify:** Utilising facial recognition algorithms to assess the similarity percentage between the captured image and the user's government-issued document photo for identity verification.

These steps collectively aim to enhance the efficiency, accuracy, and security of the onboarding process while mitigating the complexities and delays associated with traditional KYC methods.

Throughout this report, we will delve into the rationale behind adopting this approach, its technical implementation, and the anticipated benefits for businesses and end-users alike. By

exploring these aspects, we aim to demonstrate how our innovative use of FastAPI and WebSocket technology represents a significant advancement in modern identity verification practices for customer onboarding.

CURRENT STATE AND CONTEXT

In the current landscape of identity verification for customer onboarding, we rely on third-party vendors like HyperVerge to provide solutions that ensure compliance, security, and efficiency. These vendors specialise in offering robust technology frameworks that leverage AI and machine learning to automate the verification process, reducing manual intervention and enhancing accuracy.

Vendor Utilisation: HyperVerge

HyperVerge is currently utilised to deploy solutions capabilities such as facial recognition, document verification, etc. Their expertise in AI-driven identity verification allows businesses to streamline onboarding processes, adhere to regulatory requirements like KYC, and mitigate the risks associated with manual verification methods. However, while vendor solutions like HyperVerge provide valuable off-the-shelf functionalities, there are compelling reasons to consider developing an in-house solution:

1. **Customization and Flexibility:** An in-house solution allows for tailor-made features and workflows that specifically meet the unique needs and operational nuances of the business. This flexibility can be crucial in adapting to evolving regulatory requirements or customer expectations.

2. **Cost Efficiency:** While initial development costs may be higher, an in-house solution can potentially offer long-term cost savings compared to ongoing vendor licensing fees and service agreements. It also eliminates dependency on external vendors for critical operations.

3. **Data Security and Control:** Direct control over the entire identity verification process ensures better management of sensitive customer data, enhancing security and compliance with data protection regulations.

4. **Integration with Existing Systems:** An in-house solution can be seamlessly integrated with existing backend systems, CRM platforms, and customer databases,

optimising workflow efficiency and data management.

5. **Scalability and Innovation:** Internal development enables continuous improvement and innovation, allowing the organisation to stay ahead of industry trends and customer expectations.

MOTIVATION

1. **Customization and Adaptability**: While vendor solutions like HyperVerge offer robust functionalities, they are designed to cater to a broad spectrum of businesses. An in-house solution allows customization to meet specific organisational needs, workflows, and compliance requirements more precisely. This tailored approach ensures that the identity verification process aligns seamlessly with internal operations and customer service strategies.

2. **Cost Efficiency and Long-term Sustainability**: Developing an in-house solution involves upfront investment in technology and expertise. However, over time, it can lead to significant cost savings compared to recurring vendor fees and licensing agreements. This financial prudence is augmented by the ability to scale and innovate without dependency on external vendors, thereby fostering long-term sustainability.

3. **Enhanced Data Security and Compliance**: Direct control over the entire identity verification process enhances data security and regulatory compliance. Organisations can implement stringent data protection measures and protocols, ensuring confidentiality and trustworthiness in handling sensitive customer information. This control mitigates risks associated with outsourcing critical operations to third-party vendors.

4. **Integration with Existing Systems**: An in-house solution facilitates seamless integration with existing backend systems, CRM platforms, and customer databases. This integration optimises operational efficiency, reduces redundancies, and improves the overall customer experience by providing unified and consistent service delivery.

5. **Agility and Innovation**: Internal development empowers organisations to innovate continuously, staying ahead of market trends and technological advancements. It enables agile responses to regulatory changes and customer expectations, fostering a culture of innovation and responsiveness within the organisation.

REQUIREMENTS

Functional Requirements:

- **Face Detection and Validation:** The system must accurately detect and validate the presence of a user's face during the onboarding process. Utilise computer vision techniques to ensure reliable face detection under various lighting conditions and angles.

- **Eye Detection and Blink Monitoring:** Implement algorithms to monitor eye movements and detect natural blinks as indicators of user presence. Ensure real-time monitoring to prevent spoofing attempts.

- **Automated Selfie Capture:** Automatically capture a selfie of the user after verifying stability and active presence. Define a stable period (e.g., 2 seconds) before triggering the capture process.

- **Document Verification Integration:** Integrate with government-issued ID verification systems (e.g., passport, driver's licence) for automatic comparison with the captured selfie. Ensure compliance with KYC (Know Your Customer) and other regulatory requirements.

- **Backend System Integration:** Integrate with FastAPI for robust backend operations to manage user data and verification processes securely. Implement WebSocket communication for real-time updates and interactions during the onboarding process.

- **Data Security and Privacy:** Implement robust data security measures to protect sensitive customer information throughout the verification process. Ensure compliance with data protection regulations (e.g, DPDPA) and internal data handling policies.

- **Scalability and Performance:** Design the system to handle concurrent user verifications efficiently, supporting scalability as the customer base grows. Optimise

performance to deliver fast response times during peak usage periods.

Non-Functional Requirements:

- **Accuracy and Reliability:** Achieve high accuracy in face detection and verification processes to minimise false positives and negatives. Ensure reliable performance across different devices and network conditions.

- **Usability and User Experience:** Provide a seamless and intuitive user experience during the onboarding process. Minimise user effort required for identity verification while maintaining security standards.

- **Compliance and Auditability:** Enable auditing capabilities to track and monitor verification activities for compliance purposes. Facilitate easy retrieval of audit logs and reports as required by regulatory authorities.

- **Adaptability and Flexibility:** Design the system with modular components to facilitate future updates and enhancements. Ensure compatibility with evolving regulatory requirements and technological advancements.

Technical Requirements

Hardware Requirements:

- **Processing Power:** A minimum of 8gb of RAM is required to process a frame under 100 ms to attain 10fps.

Software Requirements:

- **Operating System Compatibility:** The service should be compatible with android and iOS for user facing applications. Compatibility with backend server operating systems (e.g, Linux distributions) for hosting the verification system.

- **Development Frameworks and Libraries:** Utilising FastAPI for developing robust backend APIs that manage user data and verification processes securely. Incorporating deep learning frameworks (e.g., TensorFlow, PyTorch) for implementation of face detection, eye monitoring, and image comparison algorithms.

- **Real-time Communication Protocol:** Implementing WebSocket protocol for establishing real-time communication between the user application and backend servers.

- **Database Management System:** The project will use MongoDB for storing user profiles, verification results, and warning logs.

POTENTIAL SOLUTIONS

Approach 1: Using cv2 Module in JavaScript with Backend API Integration

This approach leverages the cv2 module in JavaScript (OpenCV.js) to handle webcam access on the client-side and send frame data to a backend API for processing and verification.

Steps Involved:

1. **<u>Client-side Implementation</u>:**
    a. Webcam Access: Use libraries like `cv2` (OpenCV.js) to access the user's webcam in the browser.
    b. Frame Capture: Continuously capture frames from the webcam. Convert these frames into a suitable format for processing, such as Base64 encoding or Blob objects.
    c. Request Sending: Send each frame to the backend API as a POST request, typically in JSON format or as form data.

2. **<u>Backend API Integration</u>:**
    a. Endpoint Setup: Create API endpoints on the backend (implemented with FastAPI, for instance) to receive incoming requests containing webcam frame data.
    b. Image Processing: Use server-side image processing libraries (like OpenCV in Python) to analyse and verify the captured frames.
    c. Response Handling: Process the frame data to detect faces, monitor eye movements, and verify user presence. Generate a response indicating the message (e.g., multiple faces detected or ensure your eyes are visible to the camera). Yield the response to the frontend which will display the message to the end user.

Approach 2: Native JavaScript Webcam Access with Backend API Integration

This approach relies on native JavaScript capabilities to access the user's webcam directly from the browser and send frames to the backend API for processing and verification.

Steps Involved:

1. **Client-side Implementation:**
    a. Webcam Access: Use JavaScript's getUserMedia API or modern browser APIs (**navigator.mediaDevices.getUserMedia**) to access the user's webcam.
    b. Frame Capture: Continuously capture webcam frames using JavaScript's `Canvas` API or similar methods. Convert frames into a suitable format (e.g., Base64 encoding).
    c. Request Sending: Send each captured frame to the backend API endpoint using asynchronous HTTP requests (e.g., Fetch API or XMLHttpRequest).

2. **Backend API Integration:**
    a. Endpoint Setup: Set up endpoints on the backend server to receive incoming requests containing webcam frame data.
    b. Image Processing: Utilise backend-side image processing libraries (e.g., OpenCV with Python) to analyse and verify the captured frames for face detection, eye movement monitoring, and user presence.
    c. Response Handling: Process the received frames, perform identity verification tasks, and generate a response indicating the verification result (e.g., authenticated or not).

PROPOSED SOLUTION

The proposed solution aims to modernise the customer onboarding process by integrating advanced technologies and robust backend infrastructure. This section outlines the key modules and steps involved in achieving efficient and secure identity verification.

Modules and Steps:

1. **Face Detection:**
    a. Objective: Ensure real-time detection and validation of the user's face in the frame.
    b. Technologies: Utilising face-recognition library which can recognize faces with an accuracy of 99.38%. It is built using dlib's state of the art face recognition model.
    c. Steps:
        i. Integrated the model for face detection under varying lighting conditions and angles.
        ii. This module will return the coordinates of the face in the frame if only one face is detected.
        iii. Different scenarios where the output will vary is "Multiple faces detected" or "Ensure your Face is Inside the Region of Interest".

2. **Eye Detection**:
    a. Objective: Ensure real-time detection and monitoring of the user's eyes in the frame to verify their active presence and reduce instances of obstruction (e.g., glasses or goggles).
    b. Technologies: Utilising the Haar Cascade Classifier model for eye detection, known for its reliability in detecting eyes in images and video streams.
    c. Steps:
        i. Implemented Haar Cascade Classifier model for eye detection, leveraging its ability to detect eyes based on specific features such as shape and intensity patterns. Integrated the model into the identity

verification system to complement face detection and enhance the overall verification process.

ii. Handling Different Scenarios: Provide feedback based on detection outcomes:

- **Single Eye Detected**: Prompt the user to ensure both eyes are visible.
- **No Eyes Detected**: Display a message instructing the user to position themselves appropriately relative to the camera.
- **Multiple Eyes Detected**: Alert the user about the presence of multiple faces or incorrect positioning.

d. Enhancing User Experience and Security:

i. By ensuring proper eye visibility, the module reduces instances where users wearing glasses or goggles may hinder accurate identity verification.

ii. Promotes a streamlined user experience by providing clear feedback and instructions based on real-time detection results.

3. <u>Blink Detection</u>:

a. Objective: Monitor eye movements in real-time to detect natural blinks during the identity verification process, ensuring user activity and preventing spoofing attempts.

b. Technologies: We are utilising the dlib library for eye coordinate detection and calculating the eye aspect ratio, a reliable method in computer vision for blink detection.

c. Steps:

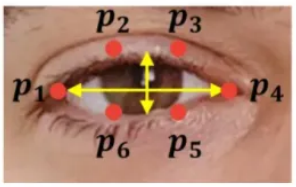
i. Eye Coordinate Detection:

- Utilise the dlib library to accurately detect and track the coordinates of the user's eyes within the captured frame.
- This step involves identifying the landmarks corresponding to the eyes using dlib's pre-trained facial landmark detector.

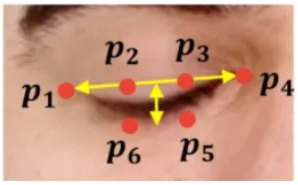
- Calculation of Eye Aspect Ratio:Compute the eye aspect ratio (EAR) using the formula:

$$\text{EAR} = \frac{\|P2 - P6\| + \|P3 - P5\|}{2 \times \|P1 - P4\|}$$

where P1,P2, P3, P4, P5 and P6 are specific landmarks on the eye detected by dlib.



Open eye will have more EAR



Closed eye will have less EAR

- The EAR provides a numerical measure of eye openness, which decreases during a blink due to eyelid closure.

d. Blink Detection Algorithm:
   i. Set a threshold EAR value (e.g., 0.25) below which a blink is detected.
   ii. Continuously monitor the computed EAR values in real-time.
   iii. Increment a blink counter each time the EAR falls below the threshold, indicating a blink has occurred.

e. Enhancing User Experience and Security:
   i. By incorporating blink detection, the module enhances security by confirming the user's active presence.
   ii. Reduces vulnerabilities to spoofing attempts by detecting natural eye movements, improving overall system reliability.

4. <u>Automatic Selfie Capture and Comparison</u>:

a. Objective: Streamline the capture and comparison process to enhance user experience by automating the selfie capture and comparing it against pre-registered government-issued IDs, ensuring compliance with KYC regulations.

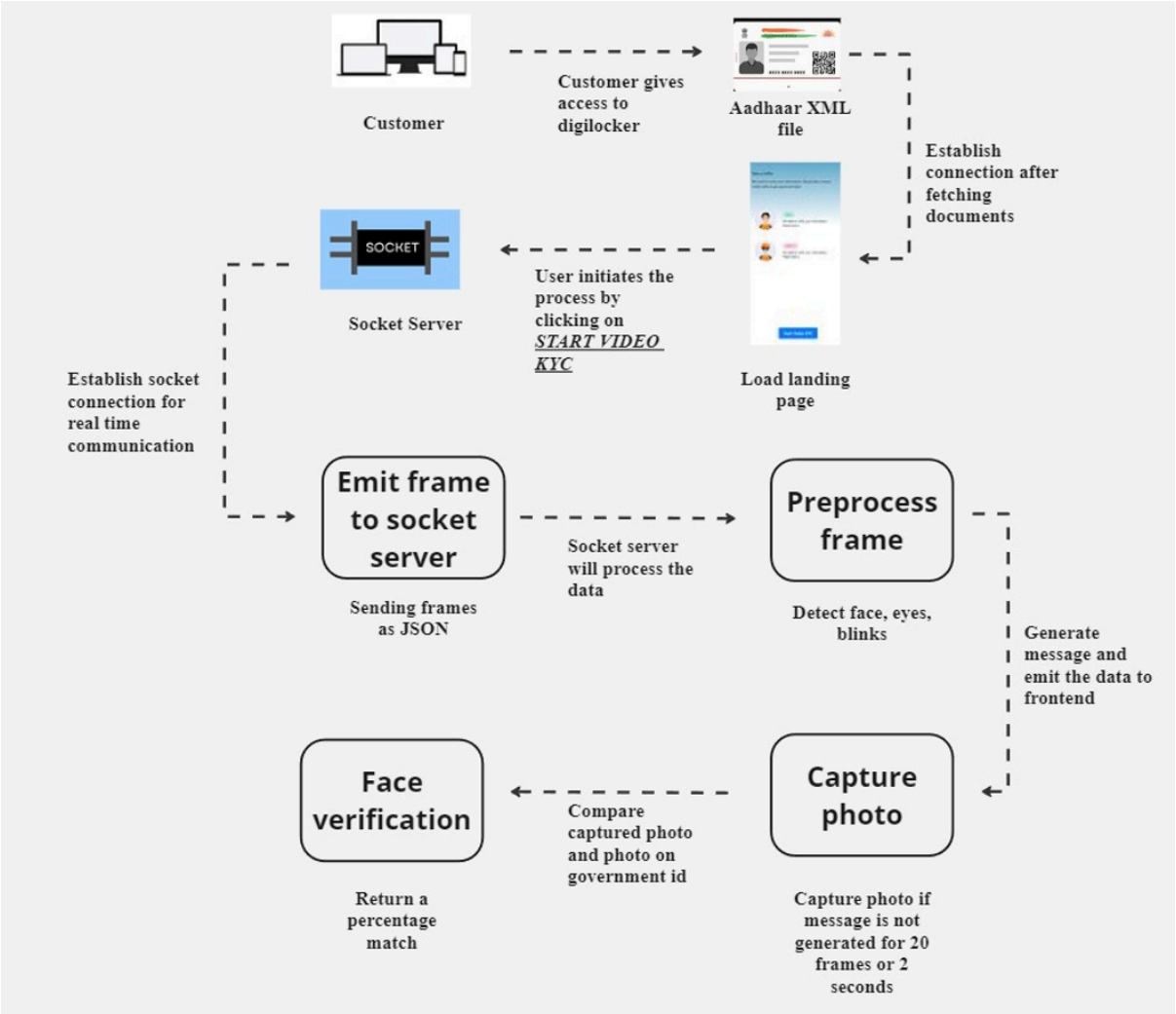
b. Technologies: Utilising image processing techniques, deepface module and face-recognition module to convert the image of the face into vector encodings and calculate the percentage similarity between the captured image and photo on government document.

c. Steps:
    i. Automatic Triggering of Selfie Capture:
        - Initiate the selfie capture process after confirming user stability and active presence.
        - Monitor the frame stream for 20 consecutive frames where the user's face and eyes are clearly visible without any warning messages (e.g., "Multiple faces detected", "Eyes not detected").
    ii. Frame Monitoring and Warning Handling:
        - Continuously monitor frames for the presence of the user's face and eyes.
        - If a warning message occurs during this monitoring period, reset the frame count back to zero and restart the monitoring process.
        - Ensure 20 consecutive frames pass without any warnings to proceed to the next step.
    iii. Capture and Feature Extraction:
        - At the 21st frame where all conditions are met (stable face and visible eyes), automatically capture the user's selfie from the frame.
        - Extract facial features and characteristics from the captured selfie using deep learning algorithms.
    iv. Comparison with Pre-Registered ID Documents:
        - Compare the extracted features from the selfie with pre-registered ID documents (e.g., passport, driver's licence, aadhaar card).
        - Implement facial recognition algorithms to ensure accurate matching and compliance with KYC regulations.
    v. Enhancing User Experience and Security:

- By automating the selfie capture based on stable face and visible eyes criteria, the module reduces user effort and enhances verification efficiency.
- Ensures compliance with KYC regulations by accurately comparing user-provided selfies with government-issued ID documents.

Final Workflow:

The final flow for the identity verification process integrated with automatic selfie capture:



1. **User Interaction Initiation:**
    a. User provides access to Digilocker or uploads government-issued documents for identity verification.
    b. Interface displays illustrations and a "Start Video KYC" button.

2. **Video KYC Initiation:**
    a. Upon clicking "Start Video KYC", the frontend initiates a request to the backend route `capture_image`.

3. **Socket Connection Establishment:**
    a. Frontend establishes a socket connection with the socket server to facilitate real-time communication for frame analysis and capture.

4. **Webcam Activation:**
    a. Frontend activates the user's webcam, capturing frames at a rate of 10 frames per second (fps).

5. **Frame Analysis and Monitoring:**
    Backend begins analysing each frame received from the webcam.
    a. Face Detection Module:
        i. Utilizes the face-recognition library (built with dlib) to detect and validate the user's face in real-time.
        ii. Provides feedback on the presence and position of the face, handling scenarios like multiple faces or improper positioning.
    b. Eye Detection Module:
        i. Implements the Haar Cascade Classifier for eye detection, ensuring the user's eyes are visible and monitoring their presence throughout the process.
        ii. Alerts users if eyes are not detected or if multiple eyes are detected, guiding proper positioning.
    c. Blink Detection Module:
        i. Utilises the dlib library to track eye coordinates and calculate the Eye Aspect Ratio (EAR) to detect natural blinks.

ii. Increments a blink counter upon detecting blinks, enhancing security by confirming user activity.

6. **<u>Warning Handling</u>:**
    a. Continuously monitors frames for 20 consecutive frames where face and eyes are clearly visible without any warnings (e.g., multiple faces detected, eyes not detected).

7. **<u>Automatic Selfie Capture</u>:**
    a. Once 20 consecutive frames meet the criteria (stable face and visible eyes), the backend automatically triggers the capture of the user's selfie from the webcam feed.
    b. Extracts facial features and characteristics using image processing techniques and deep learning algorithms (e.g., deepface module, face-recognition module).

8. **<u>Comparison with Pre-Registered ID Documents</u>:**
    a. Compare the extracted features from the captured selfie with pre-registered government-issued ID documents (e.g., passport, driver's license, Aadhaar card).
    b. Implements facial recognition algorithms to ensure accurate matching and compliance with KYC regulations.

9. **<u>Verification Result Display</u>:**
    a. Backend sends verification results to the frontend, displaying whether the identity verification was successful.
    b. Provides feedback to the user based on the comparison outcome (e.g., successful verification, additional verification required).

RESULTS AND DISCUSSION

In implementing the identity verification system outlined in this project, significant advancements have been made towards achieving efficient frame processing times, crucial for maintaining a seamless user experience and robust security standards. The system's achievement of efficient frame processing times, ranging from 1 second to 60 milliseconds, is a testament to the efficacy of leveraging **asynchronous programming** in optimising real-time data processing.

Advantages of Asynchronous Programming

- Concurrency: Asynchronous operations allow the system to perform multiple tasks simultaneously, enhancing overall throughput and responsiveness.

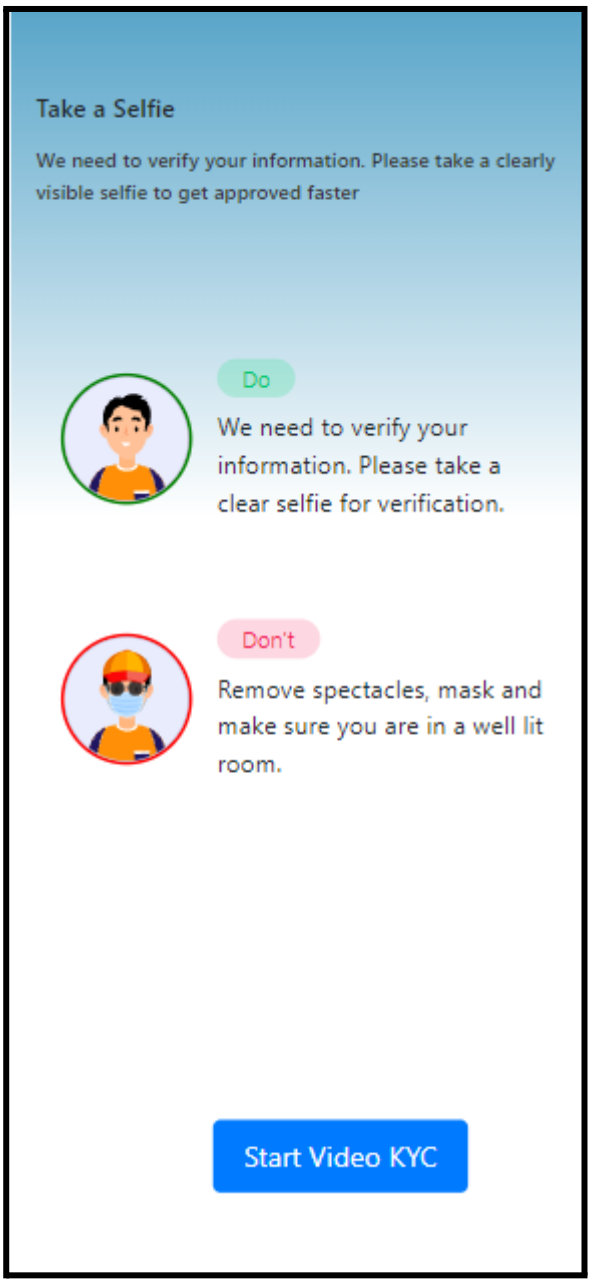
- Efficiency: By minimising blocking operations and maximising CPU utilisation, asynchronous programming reduces latency and improves the system's responsiveness to user interactions.

- Scalability: The asynchronous model supports scalability by efficiently managing resources and accommodating increasing demands without compromising performance.

OUTPUT:

1. **LANDING PAGE**

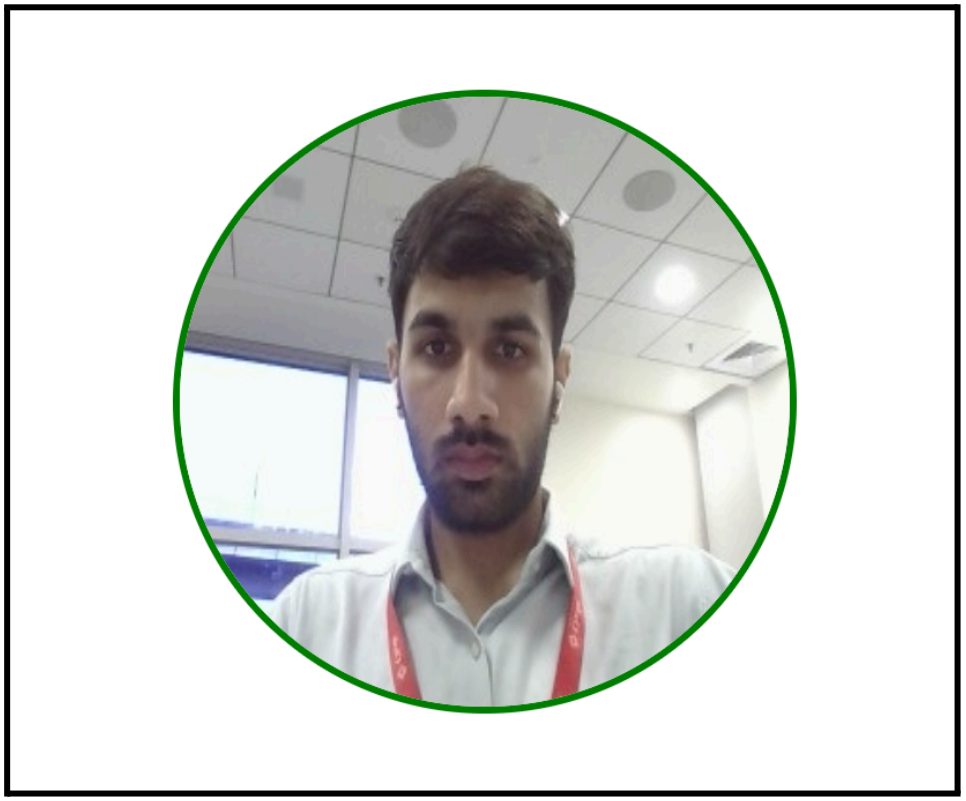
The landing page of the application features a visual guide outlining essential instructions for users, designed to ensure smooth and effective identity verification. The image below provides clear "do's and don'ts," helping users understand how to position themselves correctly during the verification process. Key instructions include removing spectacles and masks, and ensuring presence in a well-lit room.



2. CAPTURE IMAGE PAGE

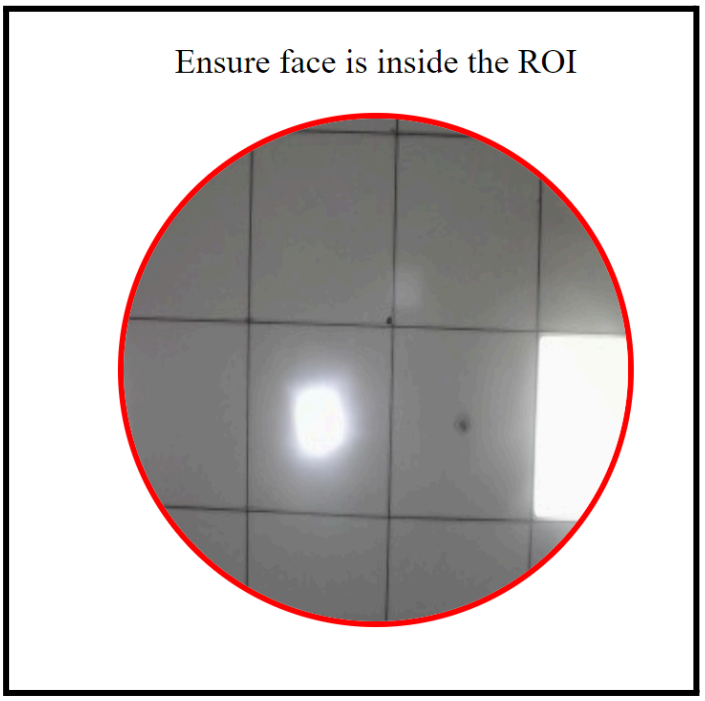
The "Capture Image" interface displays a circular area with an initial border color coded in red. This visual cue indicates to the user that the system is awaiting confirmation of optimal conditions for capturing the image.

- Initial State: Upon initialization, the circular area surrounding the capture button is bordered in red, signalling that the system is not yet ready to capture an image.
- Conditional Colour Change: As the user interacts with the system, real-time monitoring begins. If the system detects 20 consecutive frames without any warning messages (e.g., "Multiple faces detected" or "Eyes not detected"), the border colour changes to green. This change indicates that conditions for capturing the image, such as a stable face and visible eyes, have been met.
- User Feedback: If a warning message occurs during monitoring, indicating improper positioning or obstruction (e.g., glasses or inadequate lighting), the border colour remains red. This visual feedback prompts the user to adjust their position or conditions to meet verification requirements.



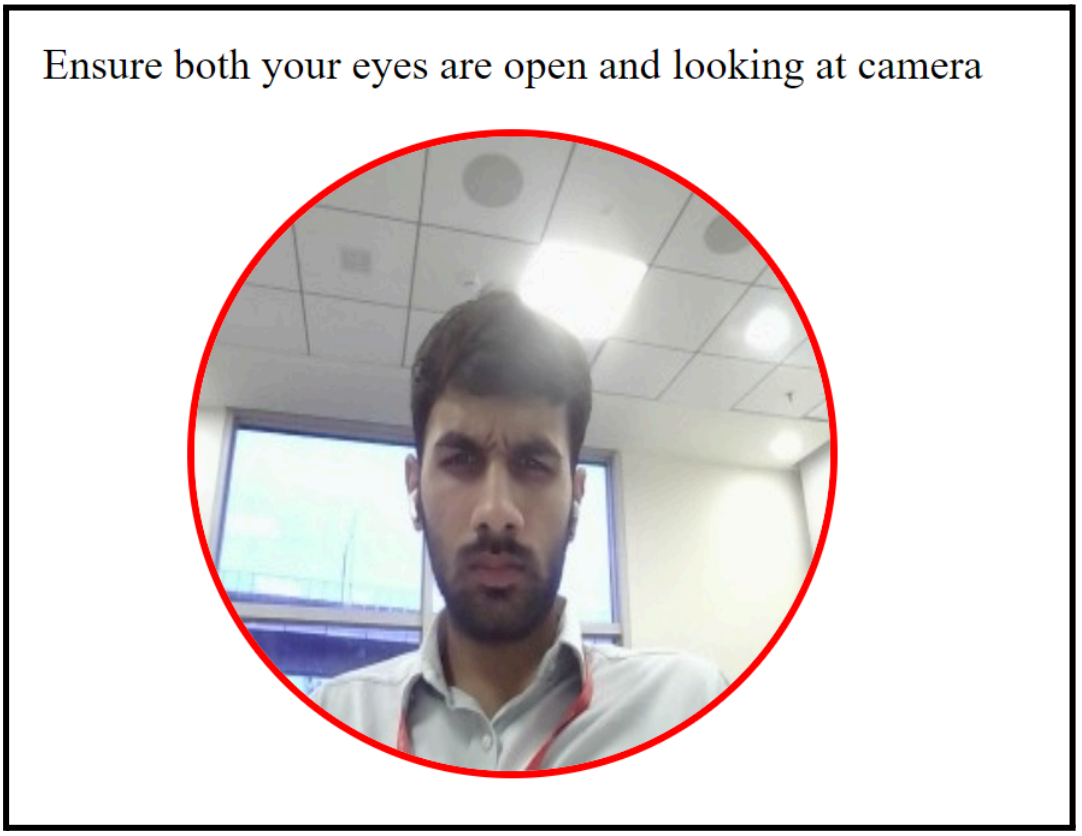
3. Ensure Face is inside the Region of Interest

The image illustrates a scenario where the user's face is not fully visible within the designated region of interest (ROI). The interface displays a red border around the ROI, indicating that the system has not detected the user's face as expected. In this image, the interface shows a situation where only a portion of the user's face is visible within the ROI.

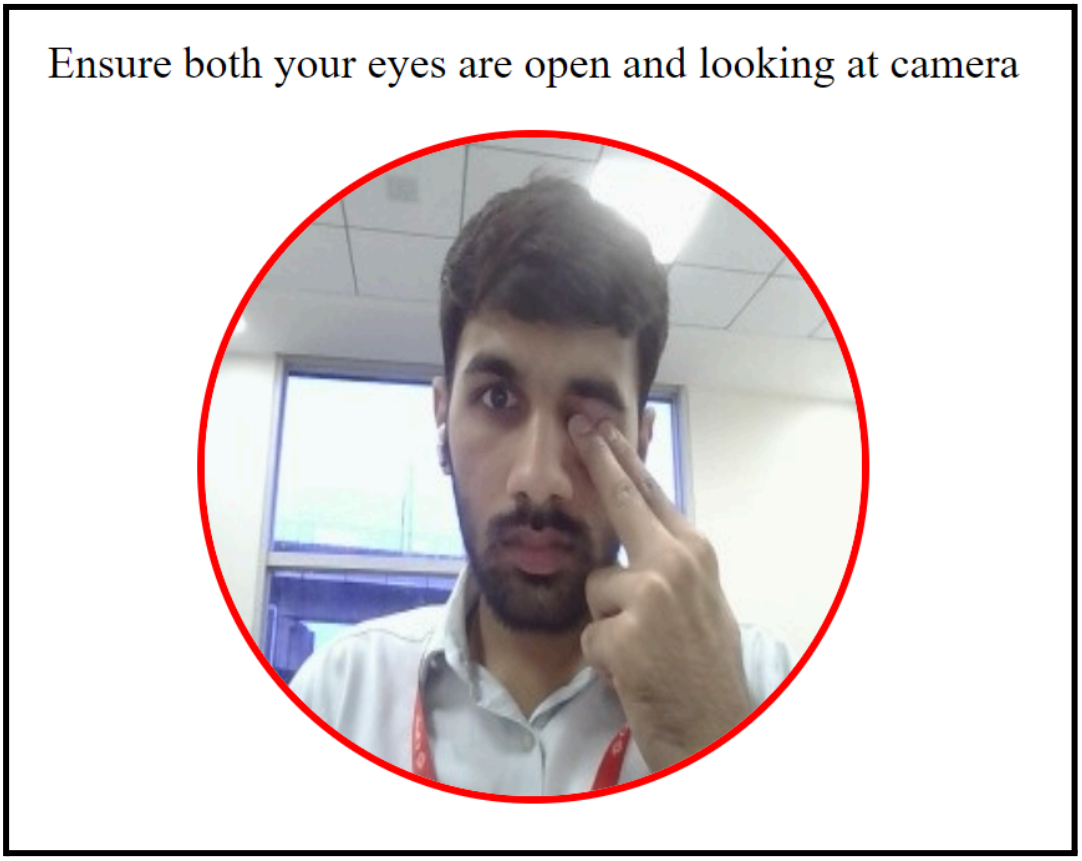


4. **<u>Ensure Both your Eyes are Looking at the Camera</u>:**

In this scenario, the interface detects that one or both of the user's eyes are not fully open while attempting identity verification. The system emphasises the need for both eyes to be fully visible and open for accurate verification.

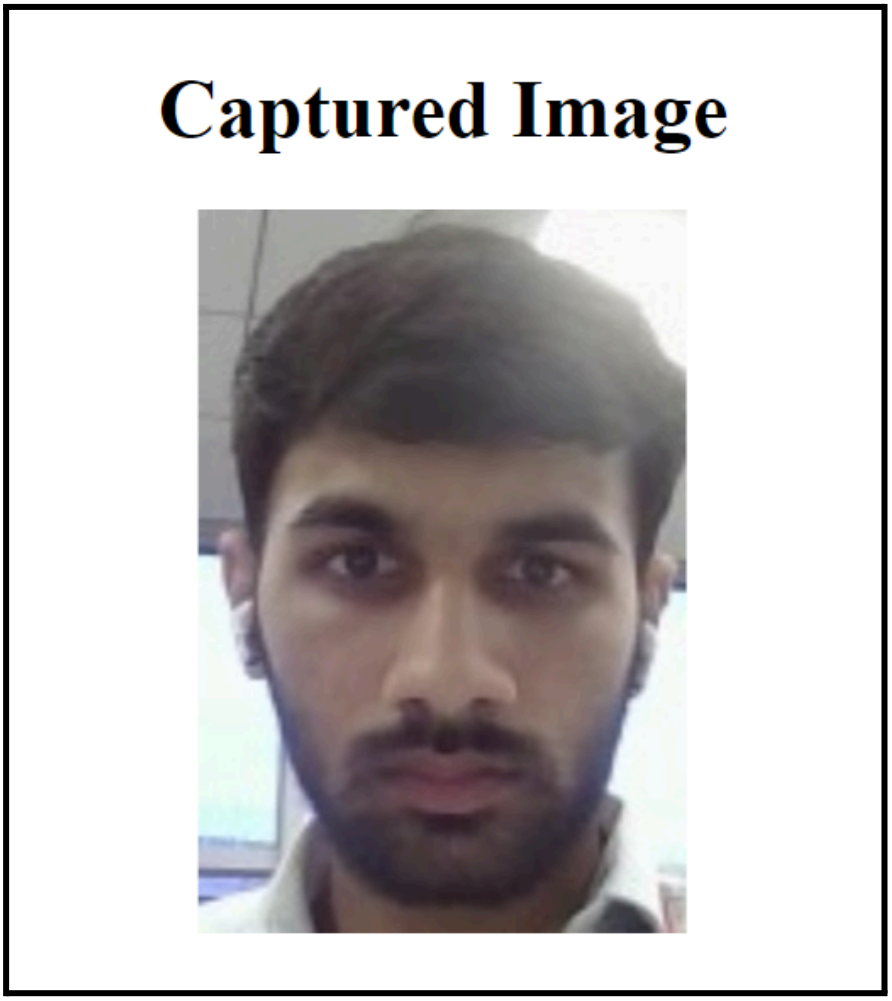


In this case, the interface detects that one or both of the user's eyes are not visible due to obstructions, such as glasses, hair, face mask, hand, etc. Proper visibility of both eyes is crucial for accurate identity verification.



5. **Captured Image:**

Upon successful completion of the identity verification process, the Captured Image page presents the final result: the user's captured image.



FUTURE ENHANCEMENTS

1. **Handling Deepfakes and Identity Spoofing:**
    a. Implement advanced AI algorithms to detect and prevent deepfake attacks during the identity verification process.
    b. Integrate deep learning models that can distinguish between genuine user interactions and synthetic or altered content.
    c. Continuously update detection techniques to stay ahead of emerging deepfake technologies.

2. **Building an SDK around this Service:**
    a. Develop a software development kit (SDK) to enable seamless integration of the identity verification service into Chola One.
    b. Provide comprehensive documentation, sample code, and libraries to facilitate easy adoption by developers.
    c. Support multiple programming languages and platforms to ensure broad compatibility and usability.

3. **Reducing False Negatives with Google's MediaPipe Solution:**
    a. Integrate Google's MediaPipe framework to improve the accuracy of eye and face detection.
    b. Leverage MediaPipe's robust features for real-time perception tasks, enhancing the system's ability to detect eyes and faces reliably.
    c. Utilise MediaPipe's capabilities to handle varying lighting conditions, occlusions, and pose variations effectively.

4. **Persistent Message Display Mechanism**:
    a. Implement a mechanism where warning messages during the verification process persist for a short duration, allowing users adequate time to read and respond.
    b. Ensure that messages remain visible on the screen until a user action acknowledges or resolves the issue.
    c. Enhance user experience by providing clear and persistent feedback, reducing confusion and improving the overall verification process efficiency.

REFERENCES

1. https://www.raconteur.net/sponsored/how-automating-kyc-can-reduce-regulatory-risk-and-boost-productivity
2. https://thefinancialbrand.com/news/bank-onboarding/why-banks-need-to-fix-poor-digital-onboarding-experience-147074/
3. https://www.globalprivacyblog.com/2023/12/indias-digital-personal-data-protection-act-2023-vs-the-gdpr-a-comparison/#:~:text=the%20GDPR%3A%20A%20Comparison,-Posted%20on%20December&text=Companies%20subject%20to%20India's%20new%20data%20protection%20law%20should%20assess%20practical%20implications.&text=The%20Indian%20parliament%20enacted%20India's,Act%202023%20(the%20DPDPA).
4. https://karza.in/products/TotalKyc
5. https://hyperverge.co/solutions/face-auth/
6. https://hyperverge.co/blog/what-is-digital-identity-verification/
7. https://hyperverge.co/blog/facial-recognition-api-vs-sdk/
8. https://kycaml.guide/blog/understanding-the-different-types-of-generative-ai-deepfake-attacks/
9. https://facia.ai/blog/liveness-detection/
10. https://ai.google.dev/edge/mediapipe/solutions/vision/face_landmarker