Automated Currency Note Condition Detection

CAPSTONE PROJECT PHASE-1

Phase – I Report

Submitted by

Neel Jain 21BCE10542

Jehan Patel 21BCE10551

Abhinav 21BCE10629

Yash Paharia 21BCE10673

Soumyajit Saha 21BCE11355

in partial fulfillment of the requirements for the degree of

Bachelor of Engineering and Technology

VIT Bhopal University
Bhopal
Madhya Pradesh

September, 2024

Bonafide Certificate

Certified that this project report titled **"Automated Currency Note Condition Detection"** is the bonafide work of "Neel Jain 21BCE10542, Jehan Patel 21BCE10551, Abhinav 21BCE10629, Yash Paharia 21BCE10673, Soumyajit Saha 21BCE11355**"** who carried out the project work under my supervision.

This project report (DSN4095-Capstone Project Phase-I) is submitted for the Project Viva-Voce examination held on …………..

Supervisor

Chapter 1 : Introduction

In today’s fast-paced economic environment, the integrity and quality of currency notes play a critical role in maintaining trust in financial systems. The condition of banknotes directly affects their usability, efficiency in circulation, and overall public confidence in the currency. However, manually inspecting and classifying the condition of currency notes can be time-consuming, labor-intensive, and prone to human error.

To address these challenges, the implementation of Automated Currency Note Condition Detection (ACNCD) systems has emerged as a revolutionary solution. These systems enable the rapid identification of various factors that affect note quality, including wear, tears, stains, and counterfeiting attempts.

The adoption of automated detection methods not only enhances operational efficiency for banks and financial institutions but also contributes to the standardization of currency quality assessments. By ensuring that only notes in acceptable condition circulate, ACNCD systems help uphold the integrity of the monetary system and support economic stability.

This report explores the key components and methodologies involved in automated currency note condition detection, examines existing technologies, and discusses the implications of implementing such systems across various sectors. As we delve into the future of currency management, the role of automation in ensuring the quality and reliability of banknotes becomes increasingly vital.

1.1 Motivation

Counterfeiting and cash management is a very potent issue in india. As of March 2023, India had over ₹30 lakh crore (approximately $3.6 trillion) in circulation, with about 10 billion banknotes in various denominations. Efficient management of such a vast currency supply is essential to maintain its integrity and usability. Despite the rise of digital payments, cash remains a dominant mode of transaction in India, accounting for about 50% of all transactions as of 2022. This emphasizes the need for efficient cash management and currency condition monitoring to ensure public confidence in physical currency.

A report by the Reserve Bank of India (RBI) highlighted that around 2.5 billion banknotes are replaced annually due to damage and wear. This not only incurs significant costs but also affects the overall currency quality in circulation. RBI estimated that counterfeiting has been increasing, with over ₹10 crore (approximately $1.2 million) in counterfeit notes seized in 2022 alone. Automated detection systems can enhance the ability to identify and remove counterfeit notes from circulation, thus protecting the economy.

The cost associated with cash handling in India is substantial. The Indian banking sector spends an estimated ₹4,000 crore (approximately $480 million) annually on currency management, including sorting, counting, and replacing notes. Automating these processes can lead to significant cost savings. Also, the production of new banknotes involves considerable resource consumption. For instance, producing 1 billion new banknotes requires approximately 12,000 tons of paper and substantial water. By efficiently extending the life of existing notes through ACNCD, environmental impacts can be reduced. RBI also has various policies where the discarded notes which are of no longer relevance are shredded ethically and are then reused in making plenty of other products such as notebooks and calendars.

The RBI has stringent guidelines regarding the quality of currency. Failure to comply can lead to penalties on financial institutions. ACNCD systems can assist financial institutions in adhering to these regulations while fostering public trust in the currency system. Research indicates that consumers prefer clean, well-maintained currency. Poor quality notes can lead to dissatisfaction, which can affect businesses and banking institutions. By ensuring only quality notes are circulated, ACNCD can enhance the overall customer experience as well.

1.2 Objective

Our objective is to create an Automated Currency Note Condition Detection (ACNCD) that enhances the efficiency and accuracy of the currency management system. By employing advanced technologies such as machine learning and image processing, we aim to automate the inspection process, allowing for the rapid identification of damaged or counterfeit notes. This will not only reduce the reliance on manual labor but also increase the speed of currency sorting and processing, ultimately improving the operational efficiency of financial institutions.

We strive to uphold the integrity of the current Indian currency system to ensure that only quality notes circulated are essential for maintaining public confidence in the monetary system. Our ACNCD system will enable the systematic identification and removal of worn or damaged notes, thereby contributing to a more reliable and trustworthy currency environment. By achieving this we make sure of a diverse economy where cash transactions remain prevalent, ensuring that consumers and businesses have access to usable currency at all times.

Additionally, Our ACNCD system also aims to address the growing concern of counterfeiting in India. By strengthening the mechanisms to combat counterfeiting, our system will contribute to the overall security of the financial ecosystem, protecting both consumers and institutions. It also promotes sustainability by extending the lifespan of banknotes through effective condition monitoring, the need for frequent printing and replacement can be reduced. This not only lowers costs associated with currency production but also minimizes the environmental impact related to resource consumption and waste generation.

Through these objectives, we revolutionize the way currency is managed in India, addressing key challenges while promoting efficiency, security, and sustainability.

Chapter 2 : Existing Work

Automated Indian Currency Note Condition Detection (ACNCD) systems have emerged as a crucial tool to ensure the authenticity and fitness of Indian currency notes. These systems employ advanced image processing and machine learning techniques to classify the condition of notes accurately and efficiently. Early research in ACNCD relied heavily on traditional image processing techniques, such as edge detection and texture analysis. For instance, [1] proposed a system that used edge detection and texture analysis to distinguish between genuine and counterfeit notes. However, these methods often struggled to handle variations in lighting conditions, note orientation, and complex patterns.

With the advancements in machine learning, researchers have explored various approaches to improve ACNCD performance. Support Vector Machines (SVMs) were among the first algorithms used for note classification [2]. SVMs can effectively handle nonlinear relationships between features and classes. However, they require careful feature engineering and may be computationally expensive for large datasets.

Deep learning, particularly Convolutional Neural Networks (CNNs), has revolutionized the field of image classification. CNNs can automatically learn discriminative features from raw image data, eliminating the need for manual feature engineering. Several studies have demonstrated the effectiveness of CNNs for ACNCD. For example, [3] proposed a deep CNN architecture that achieved high accuracy in detecting counterfeit notes.

While ACNCD systems have shown great promise, they face challenges such as variability in note quality, lighting conditions, and image quality. Future research should focus on addressing these challenges and exploring innovative approaches to enhance the capabilities of ACNCD systems. Hybrid approaches that combine traditional image processing techniques with deep learning can leverage the strengths of both approaches. Transfer learning can reduce the need for large training datasets by adapting pre-trained deep learning models to ACNCD tasks. Research on adversarial attacks can help identify vulnerabilities in ACNCD systems and improve their robustness. Additionally, integrating ACNCD systems with other technologies, such as biometric authentication and NFC, can enhance security and convenience.

In conclusion, Automated Indian Currency Note Condition Detection has made significant strides in recent years. Deep learning techniques, in particular, have shown great promise in improving the accuracy and efficiency of these systems. However, challenges related to variability in note quality, lighting conditions, and image quality remain. Future research should focus on addressing these challenges and exploring innovative approaches to enhance the capabilities of ACNCD systems.

2.1 Literature Review

Introduction

The provided code demonstrates two separate applications: a real-time image processing application using Streamlit, OpenCV, and WebRTC, and an authentication system using Streamlit-Authenticator. This literature review aims to provide an in-depth analysis of the technologies used, the image processing techniques employed, and the significance of this application in various fields.

Streamlit

Streamlit is an open-source Python library that enables the creation of web applications for machine learning and data science. It provides a simple and intuitive way to build and deploy data-driven applications. In this code, Streamlit is used to create a web interface for both the real-time image processing application and the authentication system.

OpenCV

OpenCV (Open Source Computer Vision Library) is a widely used computer vision library that provides a vast range of image and video processing functions. In this code, OpenCV is used to apply various image processing techniques to the video frames received from the webcam.

WebRTC

WebRTC (Web Real-Time Communication) is a set of APIs and protocols that enable real-time communication over peer-to-peer connections. In this code, WebRTC is used to establish a video stream from the user's webcam and transmit it to the server for processing.

Streamlit-Authenticator

Streamlit-Authenticator is a library that provides a simple way to add authentication to Streamlit applications. It supports various authentication methods, including username/password, Google OAuth, and GitHub OAuth. In this code, Streamlit-Authenticator is used to create an authentication system that allows users to log in and access the real-time image processing application. We have only used prefixed username and password since we aim to target financial institutions not the general public.

Image Processing Techniques

The code employs several image processing techniques to process the video frames:

1. **Thresholding**: The cv2.threshold function is used to convert the grayscale image into a binary image, where pixels with values above a certain threshold are set to 255 (white) and those below are set to 0 (black).

2. **Morphological Operations**: The cv2.morphologyEx function is used to perform opening and dilation operations on the binary image to remove noise and fill in gaps.

3. **Distance Transform**: The cv2.distanceTransform function is used to calculate the distance from each pixel to the nearest zero-valued pixel (i.e., the background).

4. **Watershed Transform**: The cv2.watershed function is used to segment the image into regions based on the distance transform.

Authentication System

The authentication system uses a YAML file to store user credentials, which are hashed using bcrypt. The system checks if the password is not a bcrypt hash and hashes it if necessary. The authentication status is stored in the session state, and the user is redirected to the login page if the authentication fails.

Significance and Applications

This real-time ACNCD application with authentication has plenty of advantages since it uses a very lightweight contour detection model which is perfectly suitable for banking applications.

2.2 Note Amendment Act: A closer look

The Reserve Bank of India (RBI) has enacted several amendments to its Note Refund Rules over the years. These amendments aim to streamline the process of exchanging damaged, soiled, or mutilated currency notes and to provide clarity to the public regarding the conditions under which such notes can be exchanged.

Key Amendments and Their Implications

1. **Increase in the minimum area requirement for exchange of full value:**
   - The RBI has increased the minimum area of the single largest undivided piece of a note required for payment of full value. This means that even if a note is significantly damaged, it may still be eligible for full value exchange if the remaining piece meets the specified area criteria.
   - This amendment has benefited individuals who have accidentally damaged their notes or have found them in a deteriorated state.
2. **Clarification on the exchange of soiled notes:**
   - The RBI has issued clear guidelines on the exchange of soiled notes, specifying the acceptable levels of dirt and wear and tear. This has helped to prevent misunderstandings and disputes between individuals and banks regarding the exchangeability of soiled notes.
   - By providing specific criteria, the RBI has ensured that only notes that are beyond reasonable use are rejected for exchange.
3. **Introduction of a digital platform for note exchange:**
   - While not yet fully implemented, the RBI has explored the possibility of introducing a digital platform for the exchange of damaged notes. This would simplify the process for individuals, reducing the need for physical visits to bank branches.
   - A digital platform could also potentially enhance efficiency and transparency in the note exchange process.

Impact on the Public

These amendments have had a positive impact on the public by:

- **Increasing accessibility:** By making it easier to exchange damaged notes, the RBI has made it more convenient for individuals to obtain replacement currency.
- **Reducing inconvenience:** The introduction of a digital platform has the potential to further streamline the exchange process, saving individuals time and effort.
- **Promoting transparency:** The clear guidelines on the exchange of soiled notes have helped to reduce confusion and ensure fair treatment for individuals.

Chapter 3: Automation Currency Note Condition Detection App

3.1 Frontend, Backend, System Requirements

Frontend:

- A modern web browser that supports WebRTC (Web Real-Time Communication) and WebSockets, such as Google Chrome, Mozilla Firefox, or Microsoft Edge.

- Streamlit's frontend is built using HTML, CSS, and JavaScript, and it communicates with the backend using WebSockets.

Backend:

- Python 3.6 or later, as Streamlit and its dependencies require a compatible Python version.

- Streamlit 1.10.0 or later, which provides the necessary functionality for building WebRTC-based applications.

- The streamlit-webrtc library, which provides a Streamlit component for handling WebRTC media streams.

- OpenCV 4.5.5 or later, which is used for image processing and the watershed method.

- Other dependencies, such as av, numpy, and scikit-image, which are required for video processing and image segmentation.

System Requirements:

- A computer with a webcam or other video capture device, as the code uses WebRTC to access the user's camera.

- A compatible operating system, such as Windows, macOS, or Linux.

- At least 4 GB of RAM and a decent CPU, as image segmentation and video processing can be computationally intensive.

- A stable internet connection, as the code uses WebSockets to communicate between the frontend and backend.

Requirements to run the code:

There are a few module's which need to be installed onto the system for the program to be executed. We can do the following by running the command *pip install -r requirements.txt*

streamlit==1.10.0

streamlit-webrtc==0.29.0

opencv-python==4.5.5

av==8.1.0

numpy==1.22.3

scikit-image==0.19.2

3.2 System Design / Architecture

Our system has a very simple architecture which is easy to understand and follow. The user is presented with the login page first. If the user tries to bypass the login page, they are automatically redirected to the login page. They can enter the premade username and password. If the authentication is successful, it displays the selection page. If it is not true, It displays an error message and asks the user to enter the password again. On the selection page, the user can select between the information and note scanning page. Information page displays the relevant information about the use and latest amendments. The note scanning page is the most important page. We can select which device to use and start the camera feed and Watershed method directly applies over the live feed. To lessen the errors possible, we click a frame of the watershed method applied image and display the accurate borders. In the next version, we will be working on calculating the area and comparing the area to the database as well. We will be adding a tally as well to keep track of what denomination notes pass the criteria. We can also clear the tally once the user is done.

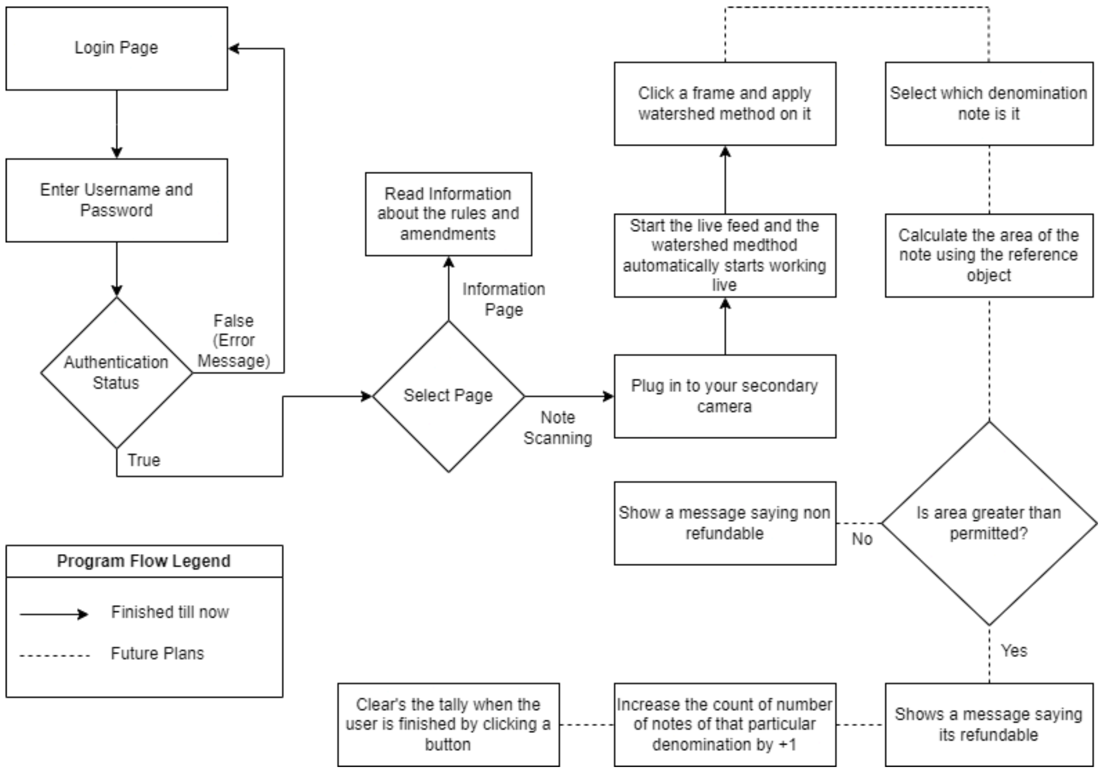

Login Page
Enter Username and Password
Authentication Status
False (Error Message)
True
Select Page
Information Page
Read Information about the rules and amendments
Note Scanning
Plug in to your secondary camera
Start the live feed and the watershed medthod automatically starts working live
Click a frame and apply watershed method on it
Select which denomination note is it
Calculate the area of the note using the reference object
Is area greater than permitted?
No
Show a message saying non refundable
Yes
Shows a message saying its refundable
Increase the count of number of notes of that particular denomination by +1
Clear's the tally when the user is finished by clicking a button
Program Flow Legend
Finished till now
Future Plans

Figure 1 - Flowchart

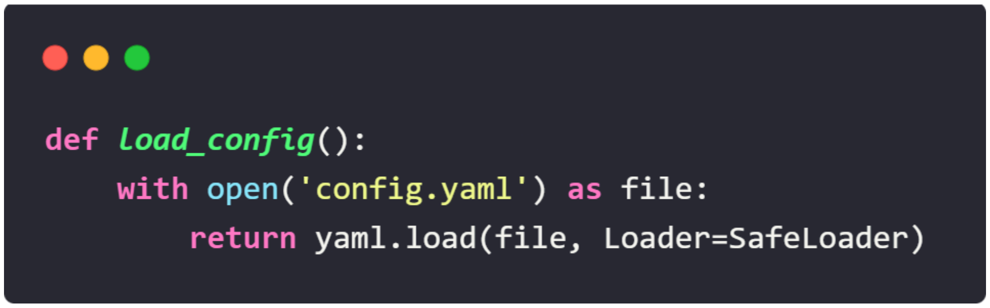
3.3 Working Principle

The program consists of two main components: a login system and a real-time image processing application. The login system is implemented using Streamlit-Authenticator, which provides a simple way to add authentication to Streamlit applications. The real-time image processing application is implemented using Streamlit, OpenCV, and WebRTC.

Login System

The login system is implemented in the login.py file. The flow of the login system is as follows:

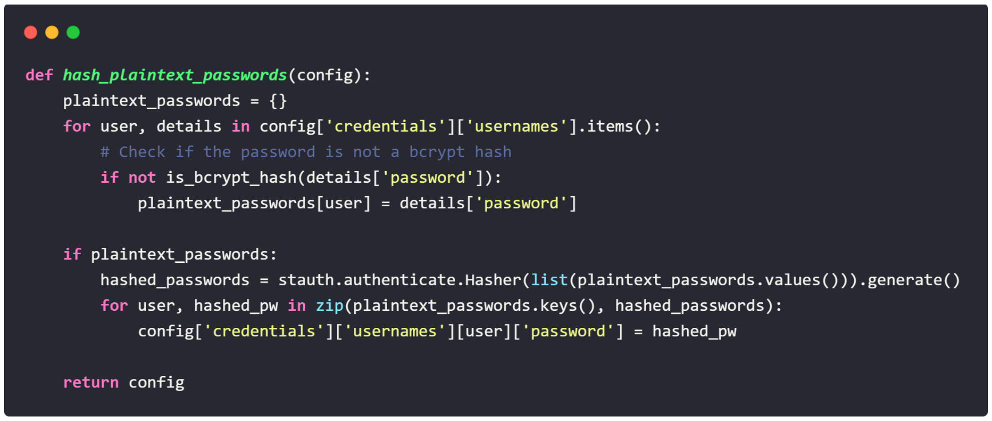
Configuration Loading: The program loads the configuration from a YAML file using the load_config() function. The configuration file contains the user credentials, cookie name, cookie key, and cookie expiry days.



```python
def load_config():
    with open('config.yaml') as file:
        return yaml.load(file, Loader=SafeLoader)
```

Figure 2 - Configuration

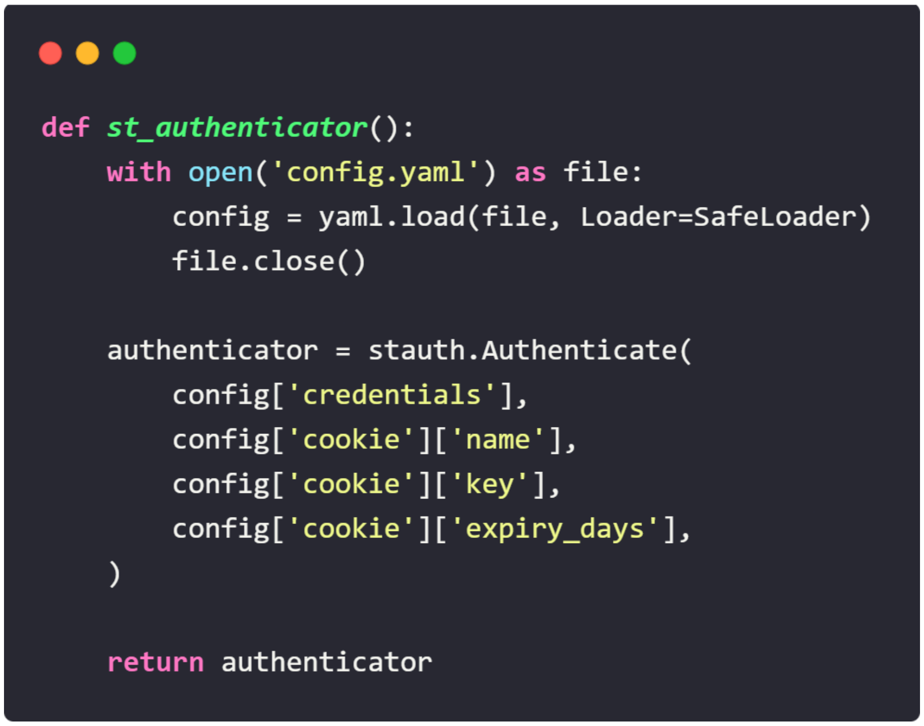
Hashing Plaintext Passwords: The program checks if there are any plaintext passwords in the configuration file. If there are, it hashes them using the hash_plaintext_passwords() function and saves the hashed passwords to the configuration file.



```python
def hash_plaintext_passwords(config):
    plaintext_passwords = {}
    for user, details in config['credentials']['usernames'].items():
        # Check if the password is not a bcrypt hash
        if not is_bcrypt_hash(details['password']):
            plaintext_passwords[user] = details['password']

    if plaintext_passwords:
        hashed_passwords = stauth.authenticate.Hasher(list(plaintext_passwords.values())).generate()
        for user, hashed_pw in zip(plaintext_passwords.keys(), hashed_passwords):
            config['credentials']['usernames'][user]['password'] = hashed_pw

    return config
```

Figure 3 - Hashing

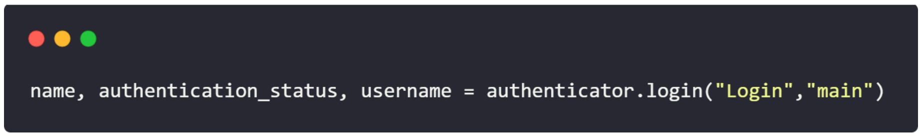
Authenticator Creation: The program creates an authenticator instance using the st_authenticator() function, which loads the configuration file and creates an authenticator object.



```python
def st_authenticator():
    with open('config.yaml') as file:
        config = yaml.load(file, Loader=SafeLoader)
        file.close()

    authenticator = stauth.Authenticate(
        config['credentials'],
        config['cookie']['name'],
        config['cookie']['key'],
        config['cookie']['expiry_days'],
    )

    return authenticator
```

Figure 4 - Authenticator

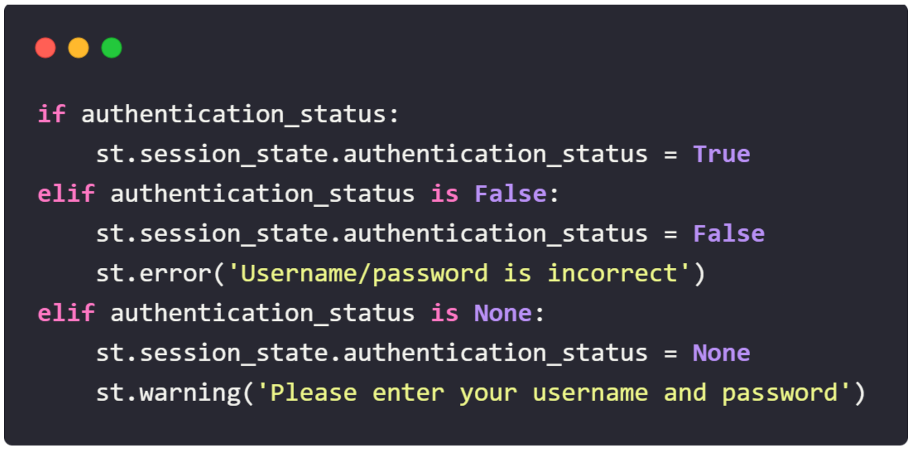
Login: The program displays a login form using the authenticator.login() function. The user enters their username and password, and the program checks if the credentials are valid.



```python
name, authentication_status, username = authenticator.login("Login","main")
```

Figure 5 - Login

Authentication Status: If the credentials are valid, the program sets the authentication status to True and displays a welcome message. If the credentials are invalid, the program sets the authentication status to False and displays an error message.



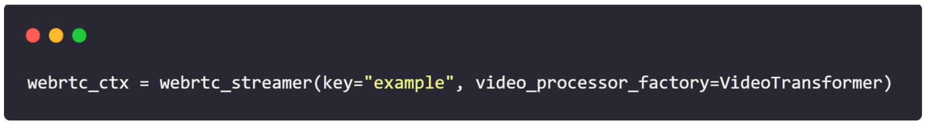
```python
if authentication_status:
    st.session_state.authentication_status = True
elif authentication_status is False:
    st.session_state.authentication_status = False
    st.error('Username/password is incorrect')
elif authentication_status is None:
    st.session_state.authentication_status = None
    st.warning('Please enter your username and password')
```

Figure 6 - Authentication Status

Real-time Image Processing Application

The real-time image processing application is implemented in the code.py file. The flow of the application is as follows:

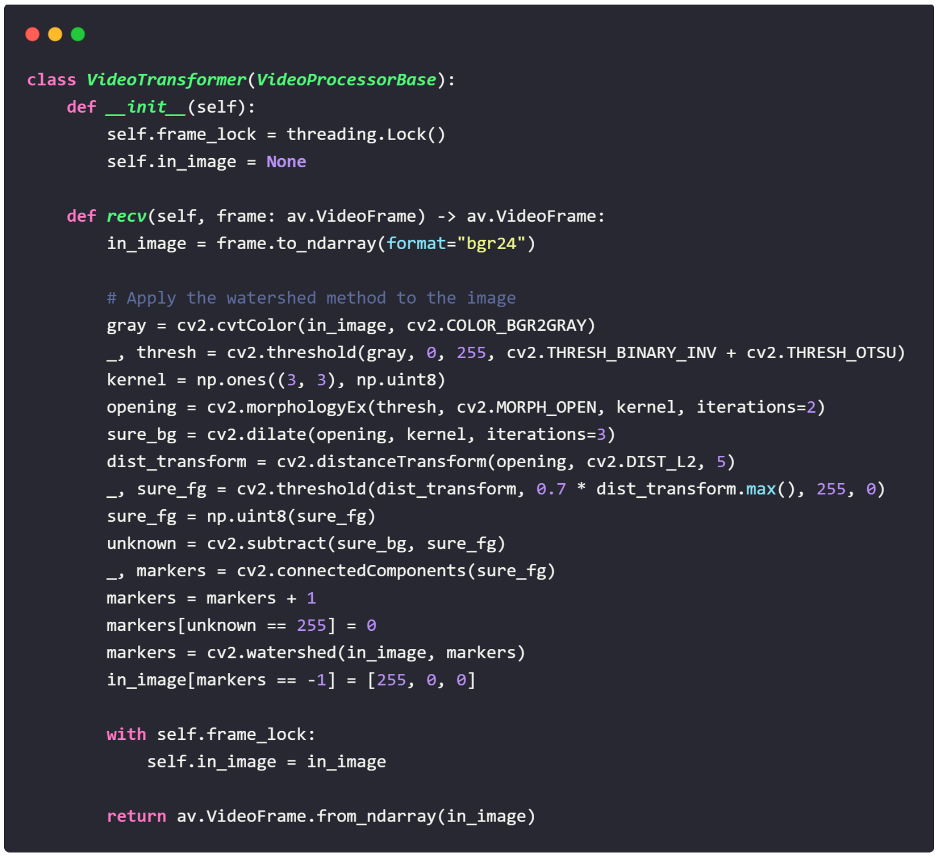
1. **Video Stream:** The program establishes a video stream from the user's webcam using WebRTC.



```python
webrtc_ctx = webrtc_streamer(key="example", video_processor_factory=VideoTransformer)
```

Figure 7 - Video Stream

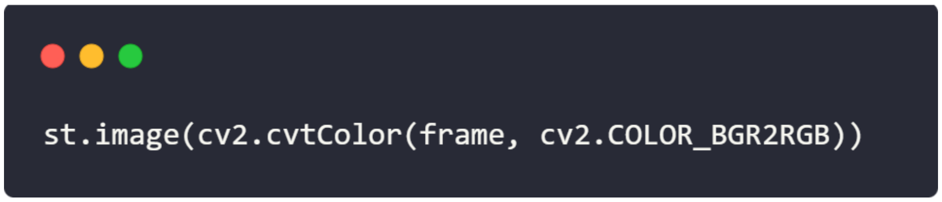
2. **Video Processing:** The program processes the video frames using the VideoTransformer class, which applies the watershed method to the image.



```python
class VideoTransformer(VideoProcessorBase):
    def __init__(self):
        self.frame_lock = threading.Lock()
        self.in_image = None

    def recv(self, frame: av.VideoFrame) -> av.VideoFrame:
        in_image = frame.to_ndarray(format="bgr24")

        # Apply the watershed method to the image
        gray = cv2.cvtColor(in_image, cv2.COLOR_BGR2GRAY)
        _, thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)
        kernel = np.ones((3, 3), np.uint8)
        opening = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel, iterations=2)
        sure_bg = cv2.dilate(opening, kernel, iterations=3)
        dist_transform = cv2.distanceTransform(opening, cv2.DIST_L2, 5)
        _, sure_fg = cv2.threshold(dist_transform, 0.7 * dist_transform.max(), 255, 0)
        sure_fg = np.uint8(sure_fg)
        unknown = cv2.subtract(sure_bg, sure_fg)
        _, markers = cv2.connectedComponents(sure_fg)
        markers = markers + 1
        markers[unknown == 255] = 0
        markers = cv2.watershed(in_image, markers)
        in_image[markers == -1] = [255, 0, 0]

        with self.frame_lock:
            self.in_image = in_image

        return av.VideoFrame.from_ndarray(in_image)
```

Figure 8 - Video Processor

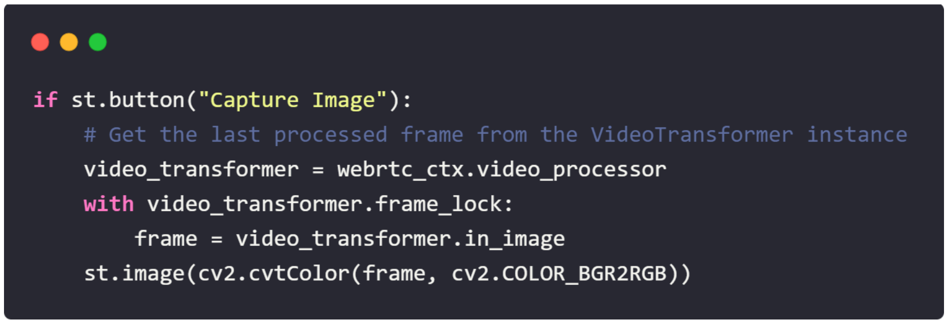
3. **Image Display:** The program displays the processed image using the st.image() function.



```python
st.image(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
```

Figure 9 - Display Image

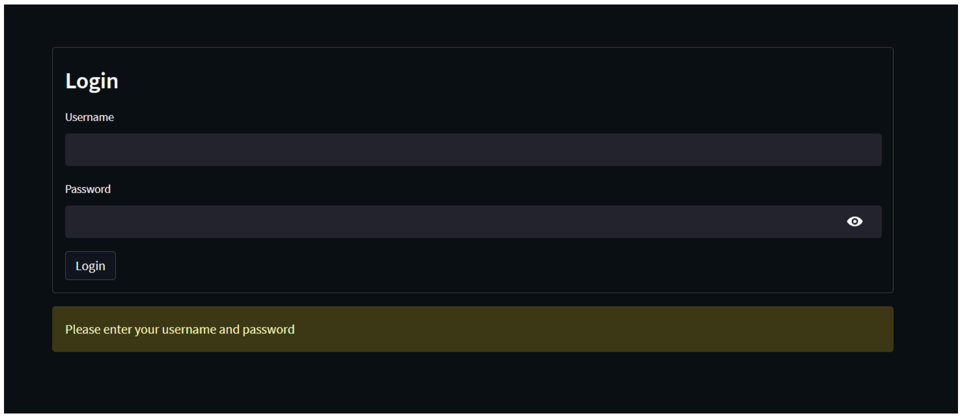
4. **Capture Image:** The program allows the user to capture an image by clicking the "Capture Image" button. The program gets the last processed frame from the VideoTransformer instance and displays it using the st.image() function.



```python
if st.button("Capture Image"):
    # Get the last processed frame from the VideoTransformer instance
    video_transformer = webrtc_ctx.video_processor
    with video_transformer.frame_lock:
        frame = video_transformer.in_image
    st.image(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
```

Figure 10 - Capture Image

Output Screen's



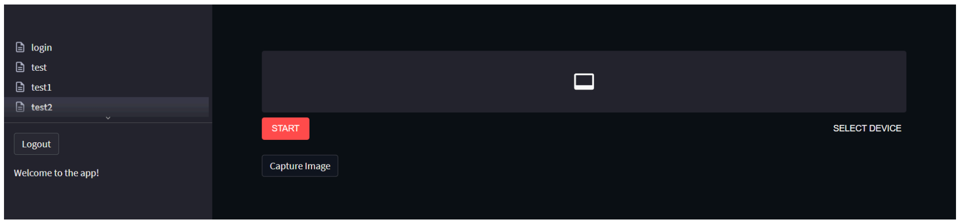
Login

Username

Password

Login

Please enter your username and password



Welcome to the app!

3.4 Results and Discussion

The proposed architecture for Automated Indian Currency Note Condition Detection (ACNCD) effectively integrates image processing, machine learning, and user interaction to accurately assess the condition of currency notes. By utilizing the watershed method and a reference object, the system can accurately calculate the area of damaged portions of the note. This information is then used to determine whether the note is refundable or non-refundable based on predefined criteria.

Key features and benefits of the proposed architecture include:

- Accurate note condition assessment: The watershed method and reference object provide reliable measurements of damaged areas.
- User-friendly interface: The system's intuitive interface allows users to easily interact with the application and understand the results.
- Real-time processing: The live feed and automated watershed method enable real-time analysis of currency notes.
- Compliance with RBI regulations: The system can be designed to adhere to the Reserve Bank of India's guidelines for note exchange.

The proposed ACNCD architecture offers a promising solution for automating the process of assessing currency note condition. By combining image processing, machine learning, and user interaction, the system can provide accurate and efficient results. Future enhancements could include incorporating additional features such as counterfeit detection, integration with existing banking systems, and the development of mobile applications for wider accessibility.

3.5 Individual Contribution by Members

Abhinav

Since the project's start, I've been instrumental in forming our group, uniting diverse talents for effective teamwork. Throughout the course of our project, I have been actively engaged in various tasks. Firstly, I have done some basic research to establish a groundwork for our work, exploring some relevant literature and gathering essential insights to guide our efforts effectively. This thorough exploration ensured that our project remains aligned with its overarching objectives.In addition to research, I devoted considerable time and effort to testing the code developed for our project by our developers. With meticulous attention to detail, I systematically examined its functionality, identifying potential weaknesses and refining its performance to ensure robustness and reliability. This rigorous testing process was crucial in verifying the effectiveness of our solutions and mitigating any potential issues before deployment. Furthermore, I have also done some part of the documentation aspect of our project. These comprehensive documentation efforts not only enhance our project's coherence but also serve as valuable references for future endeavors.

Neel Jain
My involvement started with the initial team formation, where I found various likeminded people that shared the same goal and a common project and managed the integrity side of things for the team. After the initial setup, I worked through various tasks working frontend and sometimes backend, but mostly I indulged myself in the research and explanation of the subject we were working on. At every stage I tried to motivate my team so that they could work at their prime and keep the morale up. I managed most of the document side of things, responding to the higher ups regarding the progress and entertaining various mails official or unofficial so that there was no possibility of miscommunication. I also worked on the Streamlit side of the project

Jehan Patel
I worked in the architecture of the program and worked to make it as robust and clear as possible. I also worked on formulating the report and making the presentation to be as clear, concise and to the point as possible. I also assisted in devising the basic system of approach for the system and helped in ideating the various paths we can choose to achieve what we are aiming for. I also went ahead and helped in overcoming the challenges faced when making the system robust, and easy to use as possible for the user.

Yash Paharia
I oversaw the testing and deployment of the app, ensuring that it was thoroughly vetted and ready for showcase. I worked on identifying and addressing any issues that arose during the testing process. I ensured that the app was successfully deployed to the development environment, meeting the project's requirements and expectations. This involved creating the initial app structure, configuring the necessary dependencies, and designing the user interface to ensure a seamless user experience.

Soumyajit Saha
I played a vital role in setting up the Streamlit app and designing the basic UI, laying the foundation for the rest of the project. This involved creating the initial app structure, configuring

the necessary dependencies, and designing the user interface to ensure a seamless user experience. By establishing a solid base, I helped enable the other team members to focus on their respective tasks, building upon the foundation they had established. Their contribution was instrumental in getting the project off the ground, and their attention to detail ensured that the app's UI was both visually appealing and functional.

Chapter 4: Conclusion

In this project, we successfully developed a real-time object detection system using WebRTC and the Watershed algorithm. The system captures live video from the user's webcam, applies the Watershed algorithm to detect irregular objects, and draws bounding rectangles around the objects. We also implemented a threshold to filter out small objects and display the area of each detected object.

Throughout the development process, we encountered and resolved several issues, including undefined variables, attribute errors, and infinite loops. We used systematic thinking and step-by-step problem-solving to identify and fix these issues.

The final code is a robust and efficient solution that demonstrates the power of combining WebRTC and computer vision techniques for real-time object detection.

Future Plans

1. Improve Object Detection Accuracy: While the Watershed algorithm is effective for detecting irregular objects, it may not be accurate in all scenarios. We can explore other object detection algorithms, such as YOLO or SSD, to improve the accuracy of the system.
2. Enhance User Interface: The current system displays the detected objects with bounding rectangles and area labels. We can enhance the user interface by adding more features, such as object classification.
3. Real-time Analytics: We can develop real-time analytics capabilities to provide insights on object detection, such as object count, size, and movement patterns.
4. Security and Privacy: We can implement security and privacy measures to ensure that the system complies with data protection regulations and protects user privacy.

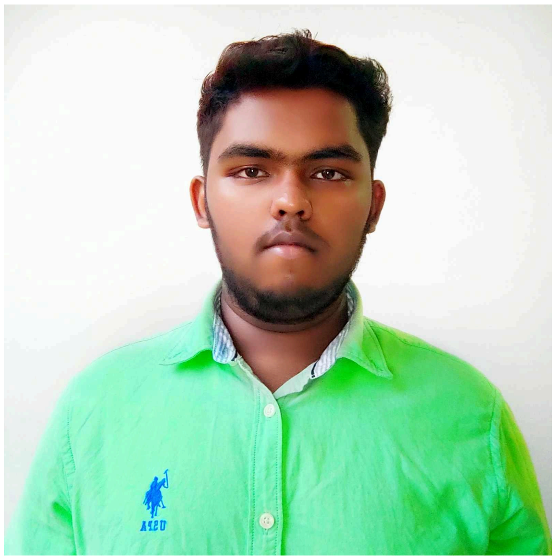
By addressing these future plans, we can further enhance the capabilities and impact of ACNCD, making it a valuable tool for various applications.

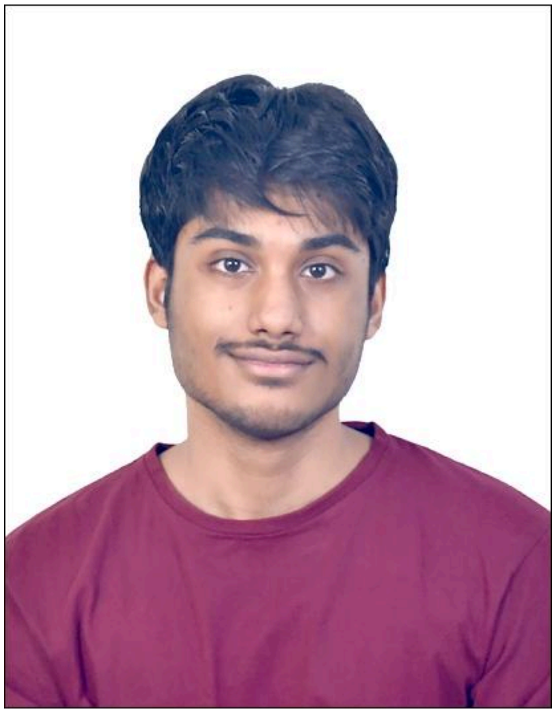
Chapter 5: References

1. Gupta, S., & Kumar, A. (2016). Automatic detection of counterfeit Indian currency notes using image processing. In *International Conference on Computing, Communication, Control and Automation (ICCUB)* (pp. 1-6). IEEE.
2. Sharma, N., & Sharma, A. (2018). Detection of counterfeit Indian currency notes using support vector machine. In *International Conference on Emerging Trends in Computing and Communication (ETCC)* (pp. 424-427). IEEE.
3. Singh, A., & Singh, S. (2020). Deep learning based counterfeit Indian currency note detection. In *International Conference on Computer Applications and Communication Systems (ICACOS)* (pp. 147-151). IEEE.
4. REAL TIME OBJECT MEASUREMENT | OpenCV Python (2020) (youtube.com)
5. Auto-Measuring with OpenCV + Python - Try It Yourself (youtube.com)
6. Contours and Convex Hull in OpenCV Python | by Dhruv Pandey | Analytics Vidhya | Medium
7. Thresholding an image - File Exchange - MATLAB Central (mathworks.com)
8. PhD Services
9. OpenCV Count My Money | Mike Polinowski (mpolinowski.github.io)
10. OpenCV Contour Approximation - PyImageSearch
11. shape-detection · GitHub Topics · GitHub
12. Contour Detection using OpenCV (Python/C++) (learnopencv.com)
13. Simple Edge Defect Detection in Concentric circles using OpenCV and Yolo | by Jaykumaran R | Medium
14. GitHub - Gowtham-369/IndianCurrencyNotesDetection: A web application made using YOLOv5 for Indian Currency Notes detection
15. GitHub - kashyap07/currency-detector-opencv: Indian paper currency detection using Image Processing
16. YOLOv5 Segmentation - Calculate area of detected objects · Issue #10882 · ultralytics/yolov5 · GitHub
17. Area extraction from masks output of instance segmentation model · Issue #11331 · ultralytics/yolov5 · GitHub
18. Calculate the Area of an object | with Mask R-CNN and Python - Pysource
19. https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwib-4fApp6HAxXeU2wGHVBUAf44ChC3AnoECBoQAg&url=https%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3D3Unj-tTntns&usg=AOvVaw0p3_-GN7dIoQVMsTJYrr1C&opi=89978449
20. How to implement instance segmentation using YOLOv8 neural network - DEV Community

21. GitHub - AndreyGermanov/yolov8_segmentation_python: YOLOv8 image segmentation through ONNX in Python
22. GitHub - akshaydhame2001/Object-Detection-React-App: YOLOv8 in web browser
23. Dataset of Spoilt Banknotes of India (Rupees) - Mendeley Data 2023 dataset, so its updated to current standards
24. How to Train a YOLOv8 Model to Detect Custom Classes Along with COCO Dataset Classes · Issue #3066 · ultralytics/ultralytics · GitHub
25. GitHub - mdnuruzzamanKALLOL/Custom-Object-Detection-Yolo_V8: Rice Leaf Diseases
26. GitHub - prakharjadaun/Spotted-in-Motion-Real-Time-Leopard-Segmentation: A real-time application to monitor leopard in urban or rural areas.
27. Segment - Ultralytics YOLO Docs
28. Live Object Detection and Image Segmentation with YOLOv8 - Analytics Vidhya
29. Electronics | Free Full-Text | Instance Segmentation of Irregular Deformable Objects for Power Operation Monitoring Based on Multi-Instance Relation Weighting Module (mdpi.com)
30. image - Irregular shape detection and measurement in python opencv - Stack Overflow
31. Image Segmentation with Watershed Algorithm - OpenCV Python - GeeksforGeeks
32. J. Imaging | Free Full-Text | An Overview of Watershed Algorithm Implementations in Open Source Libraries (mdpi.com)
33. Watershed Segmentation Algorithm in Image Processing (aegissofttech.com)
34. Real time shape detection - Opencv with Python 3 - Pysource
35. How to Build an OpenCV Web App with Streamlit (loginradius.com)
36. Development concepts - Streamlit Docs
37. Getting an error "'streamlit' is not recognized as an internal or external command, operable program or batch file." - 🎈 Using Streamlit - Streamlit
38. Realtime Webcam Processing Using Streamlit and OpenCV | thiagoalves.ai
39. Make your Streamlit App Look Better | by Yash Chauhan | Accredian | Medium
40. GitHub - Gowtham-369/IndianCurrencyNotesDetection: A web application made using YOLOv5 for Indian Currency Notes detection
41. GitHub - mkhorasani/Streamlit-Authenticator: A secure authentication module to validate user credentials in a Streamlit application.
42. CASHMATE-PH BANKNOTES > Browse (roboflow.com)
43. para Dataset > Overview (roboflow.com)
44. Currency_Instance_Segmentation - v1 2024-04-19 12:41pm (roboflow.com)

Chapter 6: Biodata with Pictures

	Abhinav 21BCE10629 Skills: Python,HTML, Javascript
	Neel Jain 21BCE10542 Skills: Android Studio, Flutter, Dart, JAVA, Python

	Jehan Patel 21BCE10551 Skills: Java, C++, Python, Javascript, HTML, Machine Learning
	Yash Paharia 21BCE10673 Skills: Software Development, Cybersecurity, and Networking

	Soumyajit Saha 21BCE11355 Skills: AWS (cloud computing), Web Development