Online Voting System using Blockchain

Presented By:
Shivani Kolanu
Department of Computer Science

Georgia State University

Agenda

- Introduction
- Motivation
- Proposed System
- Methodology
  - Face Recognition Implementation
  - Blockchain Implementation
  - UI Implementation
  - Backend Implementation
- Results of Implementation
- Demo
- Conclusion

Introduction

The Online Voting System utilizing Blockchain offers a user-friendly platform for casting votes. It allows voters to securely and conveniently submit their votes from any location using any internet-enabled device.

To ensure the security of votes, this system utilizes blockchain technology, guaranteeing that votes are stored securely and transparently, resistant to tampering or alteration.

To verify the legitimacy of voters, a two-step authentication process is implemented. This includes email verification as the initial step, followed by a facial recognition system. Only after successful completion of both authentication steps can a voter proceed to cast their vote.

This two-step authentication mechanism guarantees that votes are cast by legitimate individuals, thereby ensuring the integrity of the process. It ensures that votes are accurately cast, securely recorded, and properly counted, maintaining the integrity of the electoral process.

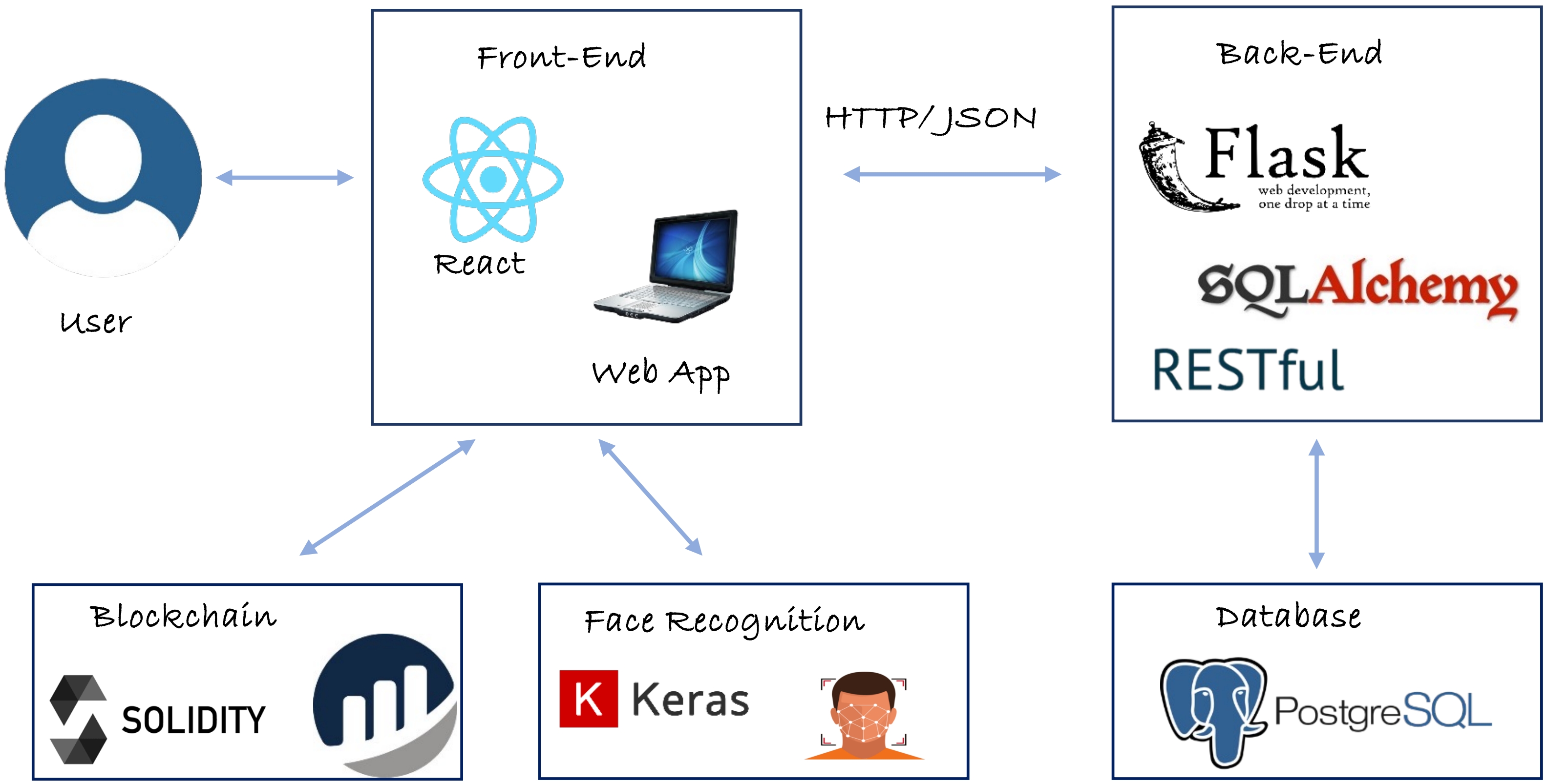
Motivation

In contrast to traditional in-person voting methods, this online voting system offers convenience by allowing individuals to vote remotely.

What distinguishes this system is its emphasis on security and authentication. By storing votes on blockchain, we ensure tamper-proof records. Moreover, our integration of facial recognition technology verifies the identity of voters, enhancing the integrity of the process.

By implementing blockchain technology and robust authentication measures, our online voting system instills confidence in the integrity of electoral processes. This trust encourages greater voter participation and ensures that election outcomes accurately reflect the will of the people.

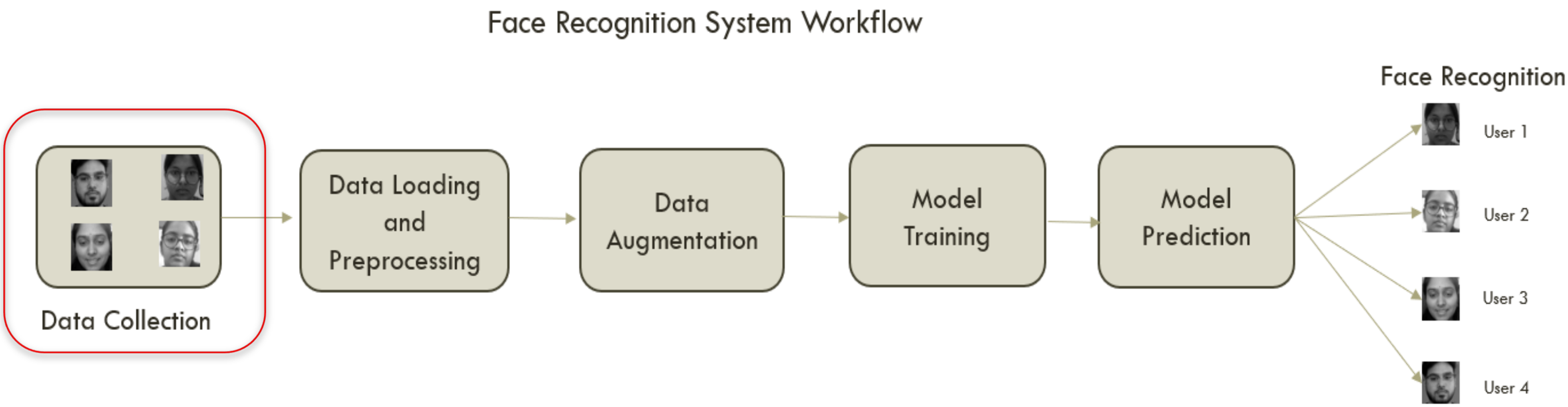
Proposed System



Architecture of the Project

Methodology - Face Recognition Implementation

Face Recognition

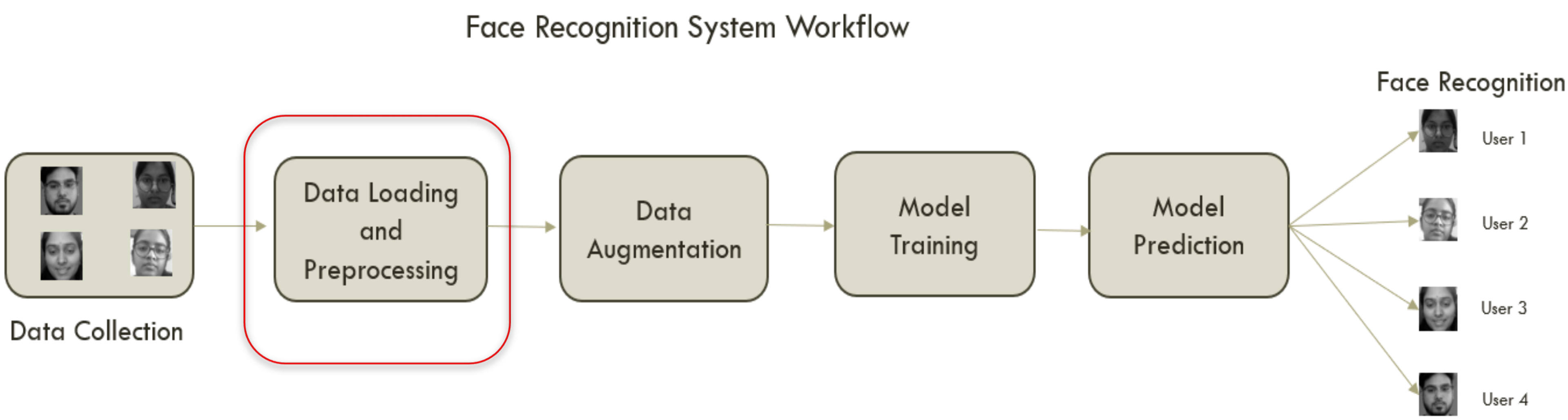


Data Collection

- *In this step, the program captures images from a video source (presumably a webcam) using OpenCV.*
- *It detects faces in each frame using a Haar cascade classifier and saves the detected face regions as images.*
- *This process continues until a certain number of images per user is collected.*
- *Uses functions like VideoCapture, CascadeClassifier and Imwrite for capturing, face detection, and image saving.*

Methodology

Face Recognition

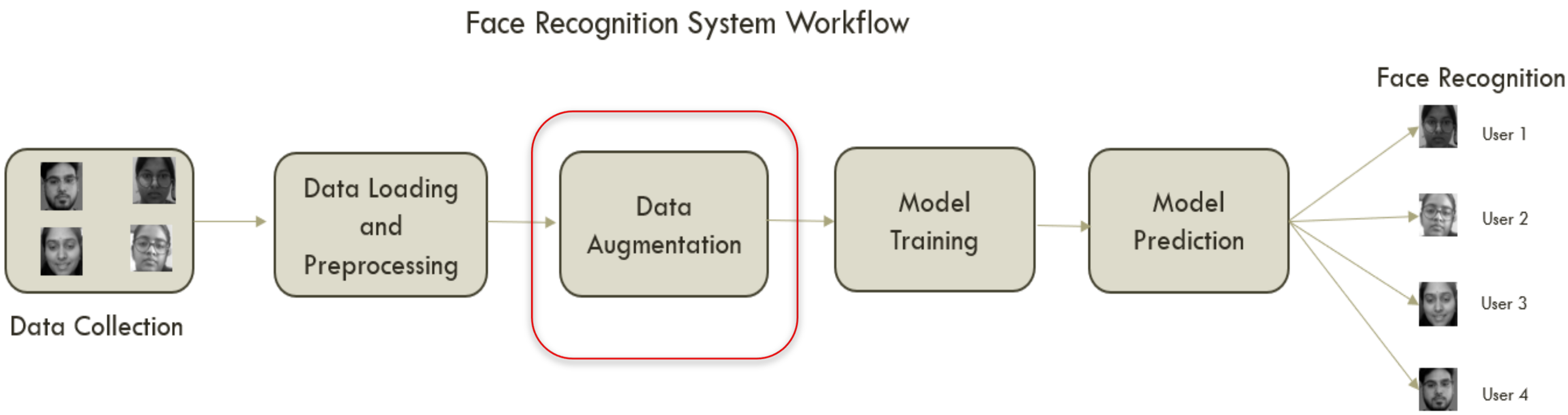


Data Loading and Preprocessing

- *After data collection, the images are stored in a directory. During preprocessing, the images are loaded, converted to grayscale, resized to a fixed dimension (100x100 pixels), and normalized to enhance contrast and make them suitable for training.*
- *Uses imread, cvtColor, and resize functions for image loading and preprocessing techniques like resizing and converting to grayscale.*

Methodology

Face Recognition

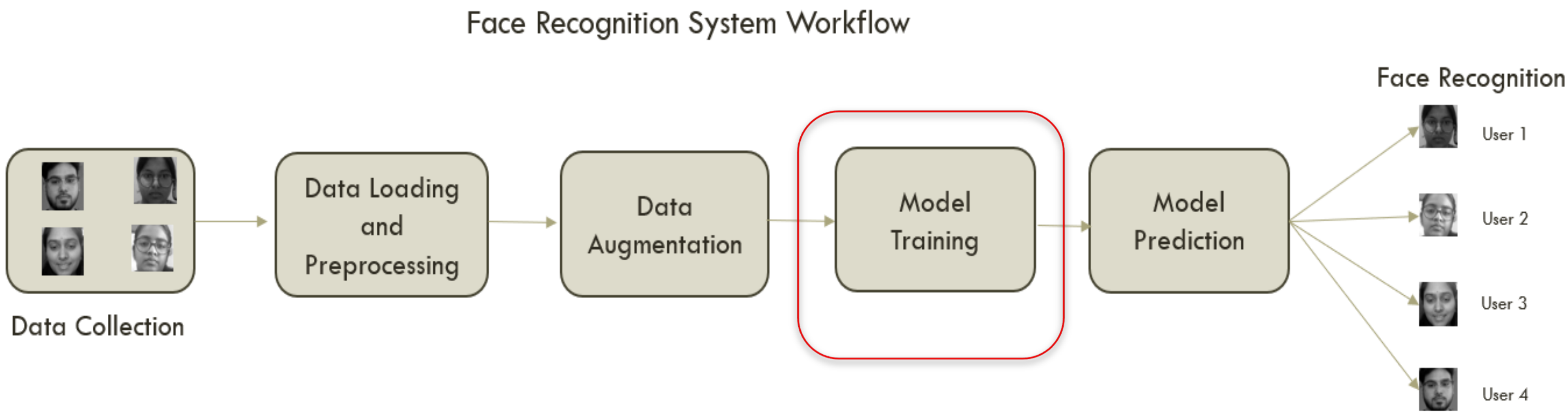


Data Augmentation

- *Data augmentation is a technique used to artificially increase the diversity of the training set by applying various transformations such as rotation, shifting, shearing, zooming, and flipping to the images.*
- *This helps in improving the model's generalization and robustness.*

Methodology

Face Recognition

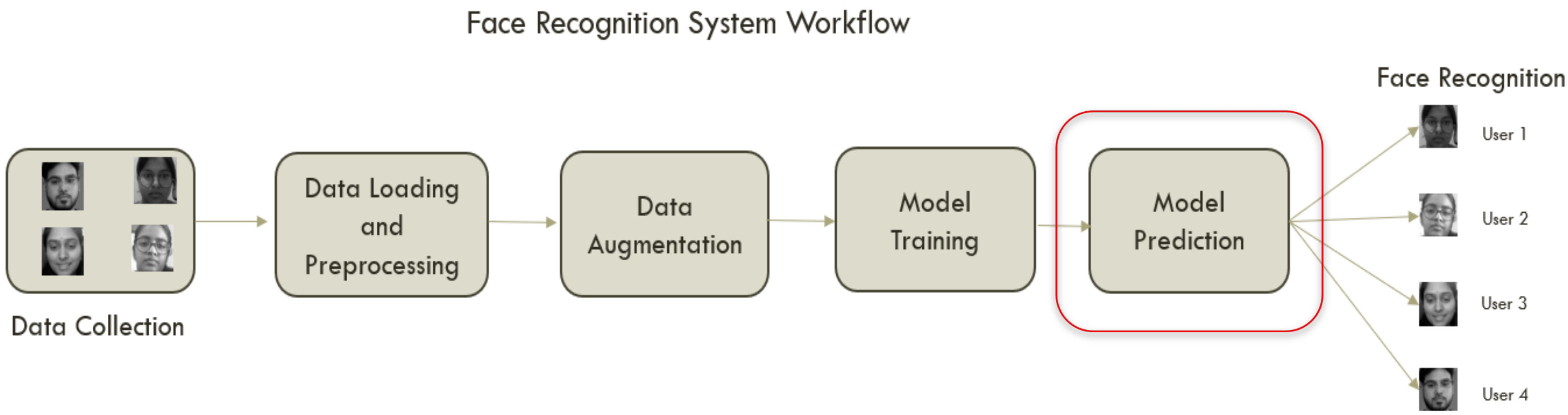


Model Training

- *In this step, a convolutional neural network (CNN) model is defined. The model consists of several convolutional layers followed by max-pooling layers for feature extraction, dropout layers for regularization, and fully connected layers for classification.*
- *The model is compiled with an optimizer (Adam) and a loss function (sparse categorical crossentropy) and then trained using the augmented training data.*

Methodology

Face Recognition

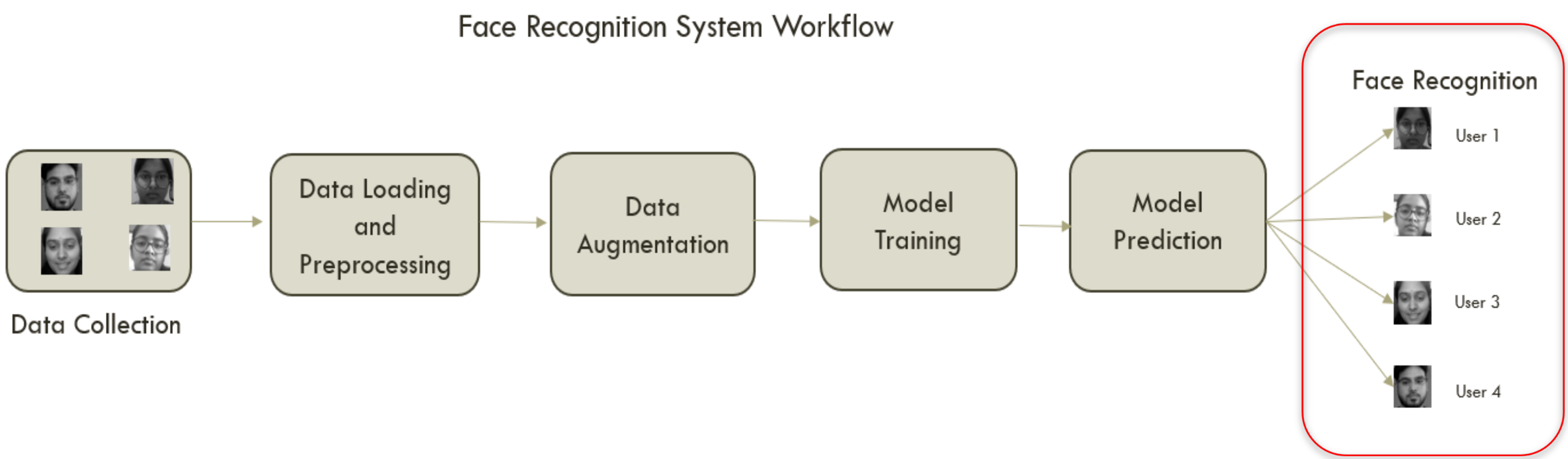


Model Prediction

- *After training, the model is used to make predictions on the test set. The trained model takes input images and outputs the predicted class labels.*
- *The predicted labels are compared with the ground truth labels to evaluate the model's performance.*

Methodology

Face Recognition



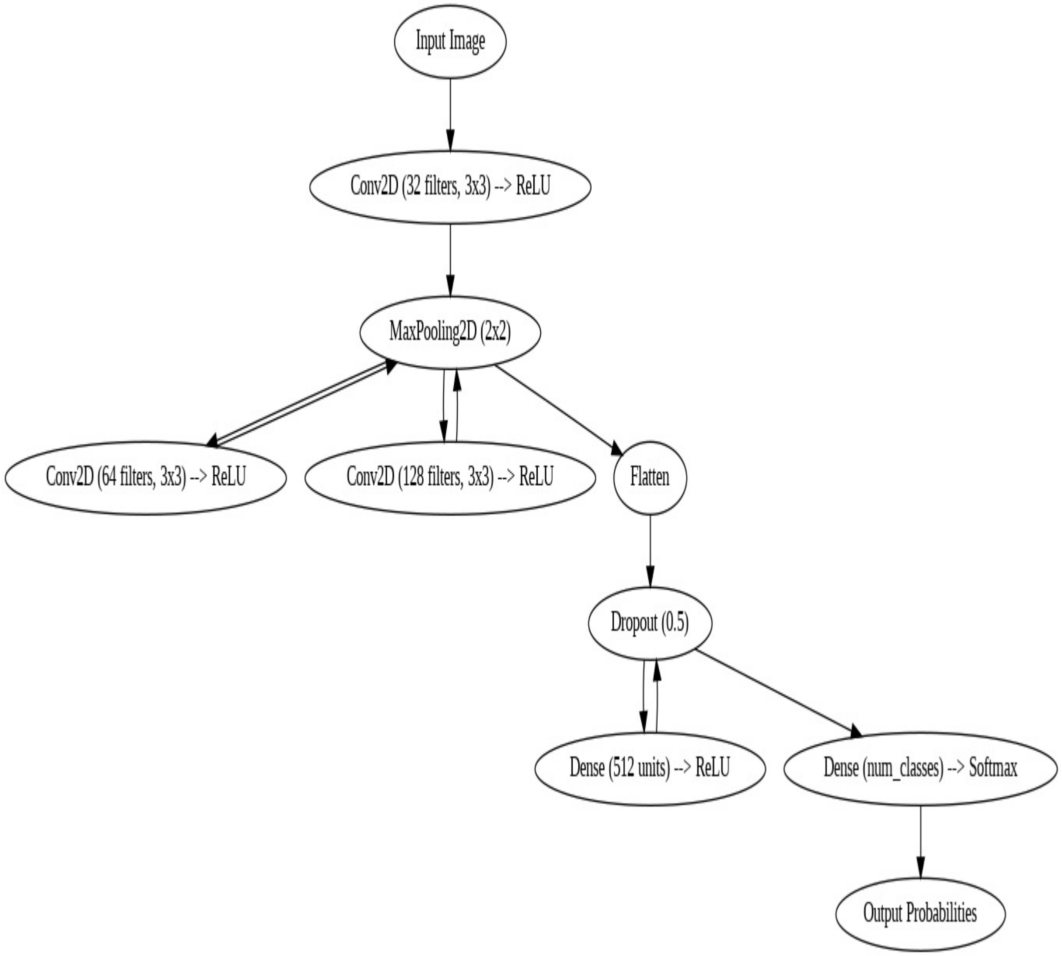
Face Recognition

- *This step involves using the trained model for real-time face recognition.*
- *The program captures video frames, detects faces using the Haar cascade classifier, preprocesses the detected face regions, and feeds them to the trained model for prediction.*
- *If the predicted user ID matches the ID of the registered user, the face is recognized as belonging to that user. Otherwise, it is classified as unknown.*

Methodology

Face Recognition Model

```
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(100, 100, 1)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dropout(0.5),
    Dense(512, activation='relu'),
    Dropout(0.5),
    Dense(num_classes, activation='softmax')
])
```



Methodology

Face Recognition Model

, rebuild TensorFlow with the appropriate compiler flags.
Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 98, 98, 32)	320
max_pooling2d (MaxPooling2D)	(None, 49, 49, 32)	0
conv2d_1 (Conv2D)	(None, 47, 47, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 23, 23, 64)	0
conv2d_2 (Conv2D)	(None, 21, 21, 128)	73856
max_pooling2d_2 (MaxPooling2D)	(None, 10, 10, 128)	0
flatten (Flatten)	(None, 12800)	0
dropout (Dropout)	(None, 12800)	0
dense (Dense)	(None, 512)	6554112
dropout_1 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 5)	2565

$$\text{output\_size} = \left\lfloor \frac{\text{input\_size} - \text{filter\_size} + 2 \times \text{padding}}{\text{stride}} \right\rfloor + 1.$$

Conv2D Layers:

Trainable Parameters = No. of Filters x (Filter Height x Filter Width x (Depth of I/o +1))

Conv2D_1 = 32 x (3x3x (1+1)) = 320 trainable params.

Conv2D_2 = 64 x (3x3x (32+1)) = 18,496 trainable params.

Conv2D_3 = 128 x (3x3x (64+1)) = 73,856 trainable params.

MaxPooling2D Layers:

MaxPooling2D_1
MaxPooling2D_2
MaxPooling2D_3

Max Pooling layers do not learn any parameters during training.

Flatten Layer:

Flatten layers do not learn any parameters during training.

Dense Layers:

Trainable Parameters: = (Input Units x Output Units) + Output Units

Dense_1 = (12800 x 512) + 512 = 6,554,112 trainable params.

Dense_2 = (512 x 5) + 5 = 2565 trainable params.

Face Recognition - Results

Face Recognition

Dataset Specifications and Results

Dataset Specifications

- Dataset Path: final_project/backend/datasets
- Total number of images: 2,100 images
- Images for training: 1,680 images (80%)
- Images for testing: 420 images (20%)

With Data Augmentation

- *Epochs: 5*
- *Optimizer: Adam Optimizer*
- *Loss Function: Sparse Categorical Entropy*
- *Batch Size: 32*
- *The Training Accuracy right now is 97%.*
- *The Testing Accuracy right now is 97.38%*
- *Loss is 0.14*

Without Data Augmentation

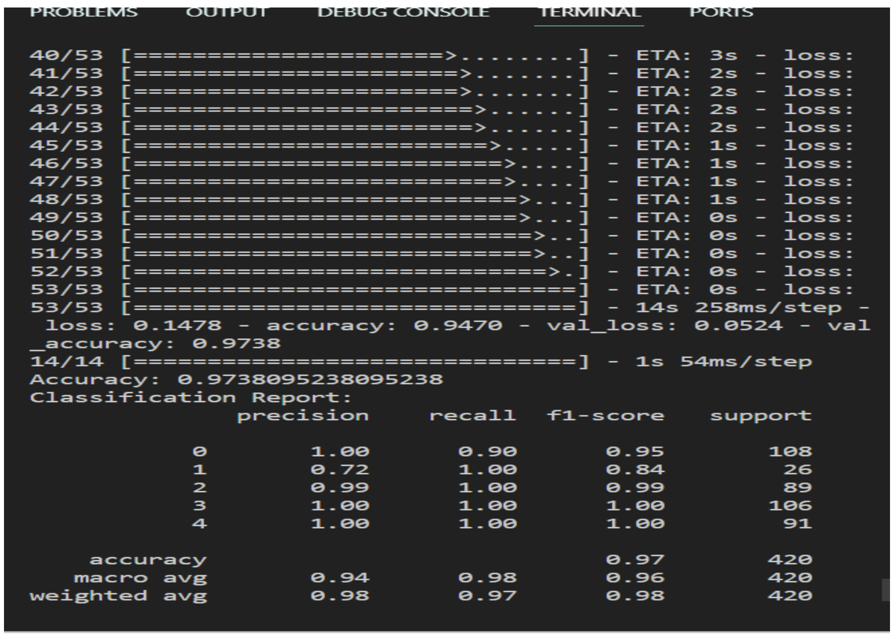
- ➢ *Epochs: 5*
- *Optimizer: Adam Optimizer*
- *Loss Function: Sparse Categorical Entropy*
- *Batch Size: 32*
- *The Training Accuracy right now is 96%.*
- *The Testing Accuracy right now is 96%*
- *Loss is 0.18*

Face Recognition - Results

Face Recognition

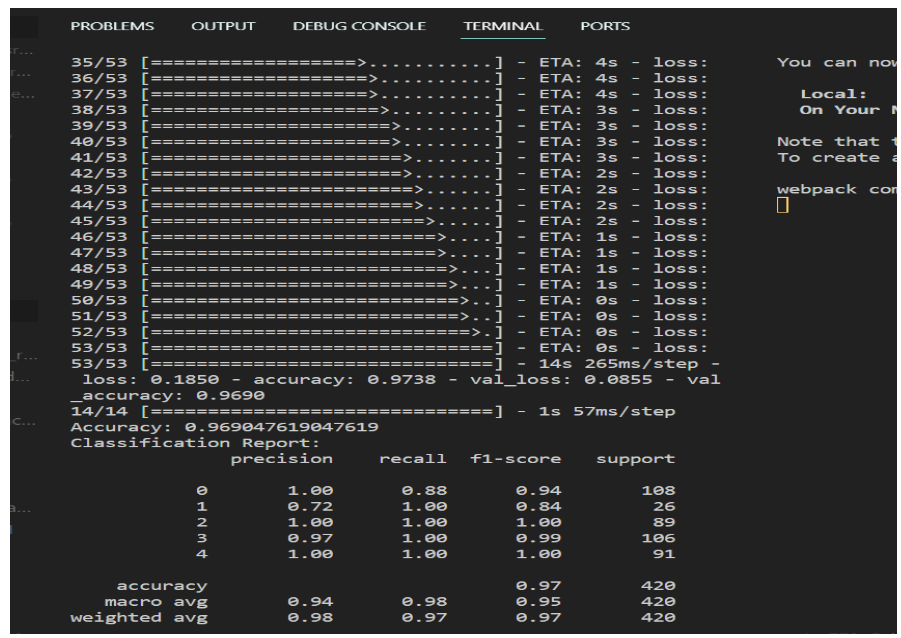
Results – Visualizations and Graphs

With Augmentation



```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

40/53 [====================>........] - ETA: 3s - loss:
41/53 [=====================>.......] - ETA: 2s - loss:
42/53 [=====================>.......] - ETA: 2s - loss:
43/53 [======================>......] - ETA: 2s - loss:
44/53 [======================>......] - ETA: 2s - loss:
45/53 [=======================>......] - ETA: 1s - loss:
46/53 [========================>....] - ETA: 1s - loss:
47/53 [========================>....] - ETA: 1s - loss:
48/53 [=========================>...] - ETA: 1s - loss:
49/53 [=========================>...] - ETA: 0s - loss:
50/53 [==========================>..] - ETA: 0s - loss:
51/53 [==========================>..] - ETA: 0s - loss:
52/53 [===========================>.] - ETA: 0s - loss:
53/53 [============================] - ETA: 0s - loss:
53/53 [============================] - 14s 258ms/step -
 loss: 0.1478 - accuracy: 0.9470 - val_loss: 0.0524 - val
_accuracy: 0.9738
14/14 [============================] - 1s 54ms/step
Accuracy: 0.9738095238095238
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.90      0.95       108
           1       0.72      1.00      0.84        26
           2       0.99      1.00      0.99        89
           3       1.00      1.00      1.00       106
           4       1.00      1.00      1.00        91

    accuracy                           0.97       420
   macro avg       0.94      0.98      0.96       420
weighted avg       0.98      0.97      0.98       420
```

Without Augmentation

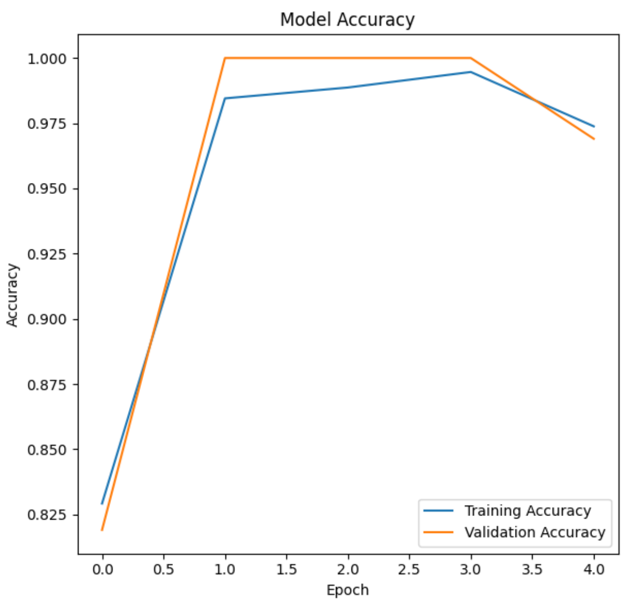


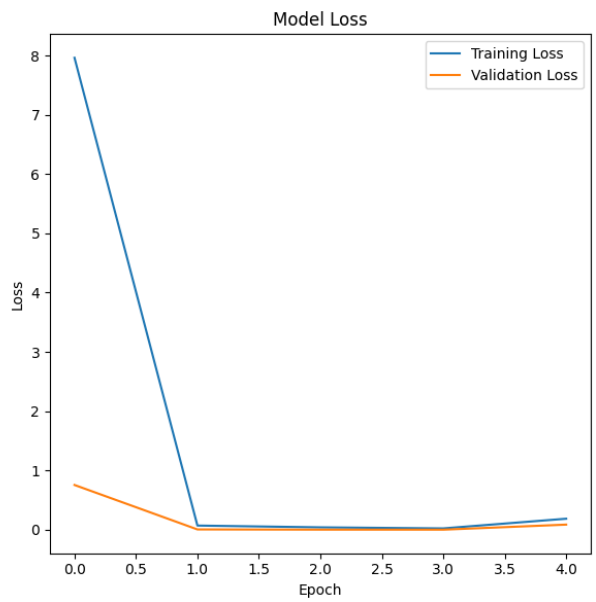
```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

35/53 [=================>...........] - ETA: 4s - loss:     You can no
36/53 [=================>...........] - ETA: 4s - loss:
37/53 [==================>..........] - ETA: 4s - loss:       Local:
38/53 [===================>.........] - ETA: 3s - loss:       On Your N
39/53 [===================>.........] - ETA: 3s - loss:
40/53 [====================>........] - ETA: 3s - loss:     Note that t
41/53 [=====================>.......] - ETA: 3s - loss:     To create a
42/53 [=====================>.......] - ETA: 2s - loss:
43/53 [======================>......] - ETA: 2s - loss:     webpack co
44/53 [======================>......] - ETA: 2s - loss:
45/53 [=======================>.....] - ETA: 2s - loss:
46/53 [========================>....] - ETA: 1s - loss:
47/53 [========================>....] - ETA: 1s - loss:
48/53 [=========================>...] - ETA: 1s - loss:
49/53 [=========================>...] - ETA: 1s - loss:
50/53 [==========================>..] - ETA: 0s - loss:
51/53 [==========================>..] - ETA: 0s - loss:
52/53 [===========================>.] - ETA: 0s - loss:
53/53 [============================] - ETA: 0s - loss:
53/53 [============================] - 14s 265ms/step -
 loss: 0.1850 - accuracy: 0.9738 - val_loss: 0.0855 - val
_accuracy: 0.9690
14/14 [============================] - 1s 57ms/step
Accuracy: 0.969047619047619
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.88      0.94       108
           1       0.72      1.00      0.84        26
           2       1.00      1.00      1.00        89
           3       0.97      1.00      0.99       106
           4       1.00      1.00      1.00        91

    accuracy                           0.97       420
   macro avg       0.94      0.98      0.95       420
weighted avg       0.98      0.97      0.97       420
```

Face Recognition - Results

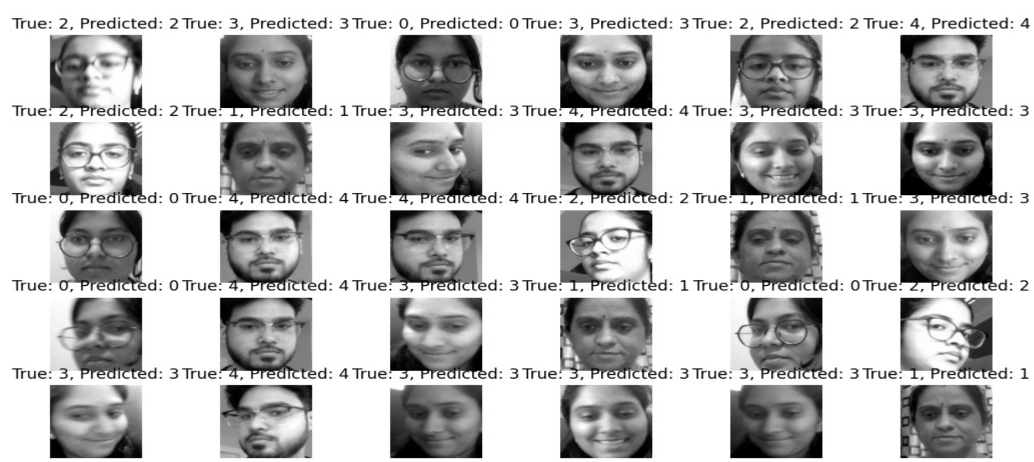
Face Recognition

Results – Visualizations and Graphs





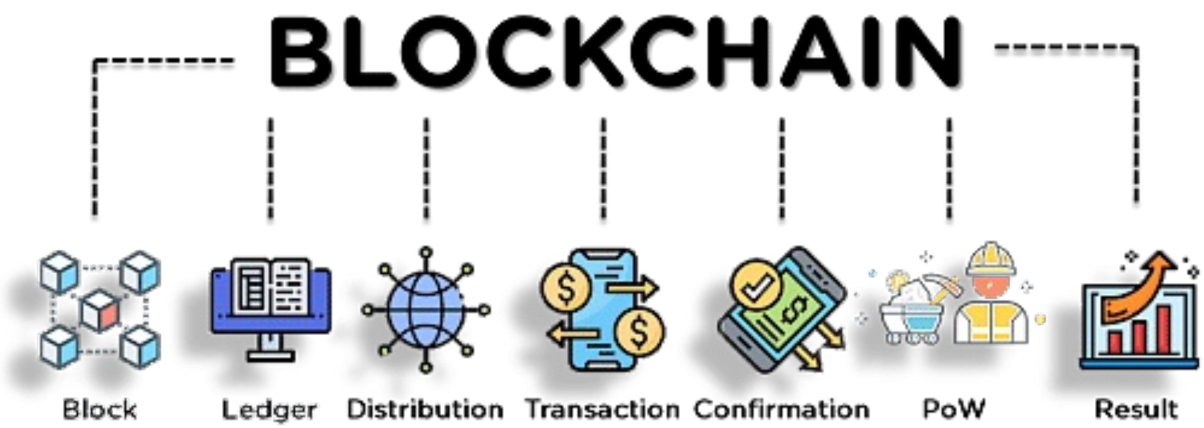
Graph explaining the Model Accuracy and Model Loss performance throughout its 5 epochs.



Model Predictions of the Test Dataset.
'True' is the actual ID number of the person.
'Predicted' is the ID which the trained model predicted.

Methodology – Blockchain Implementation

Intro to Blockchain



- ✓ Blockchain technology is an advanced database mechanism that allows transparent information sharing within a business network. A blockchain stores data in blocks that are linked together in a chain.
- ✓ Data is divided into common blocks linked together using cryptographic hashes as unique IDs.
- ✓ Data integrity is ensured via Blockchain, which uses a single source of truth to eliminate data duplication and increase security.

Methodology

Blockchain

Chosen to utilize **Ethereum**, a public blockchain, for its decentralized nature, ensuring that no single entity holds control over the network.

This decentralization enhances security, promotes censorship resistance, and ensures transparency within the system.

Although Ethereum's main network, known as **mainnet**, is the primary platform, I will be utilizing a test network for our initial transactions and contract deployments.

The test network, **Sepolia**, functions similarly to the **mainnet** but serves as a sandbox environment for testing purposes.

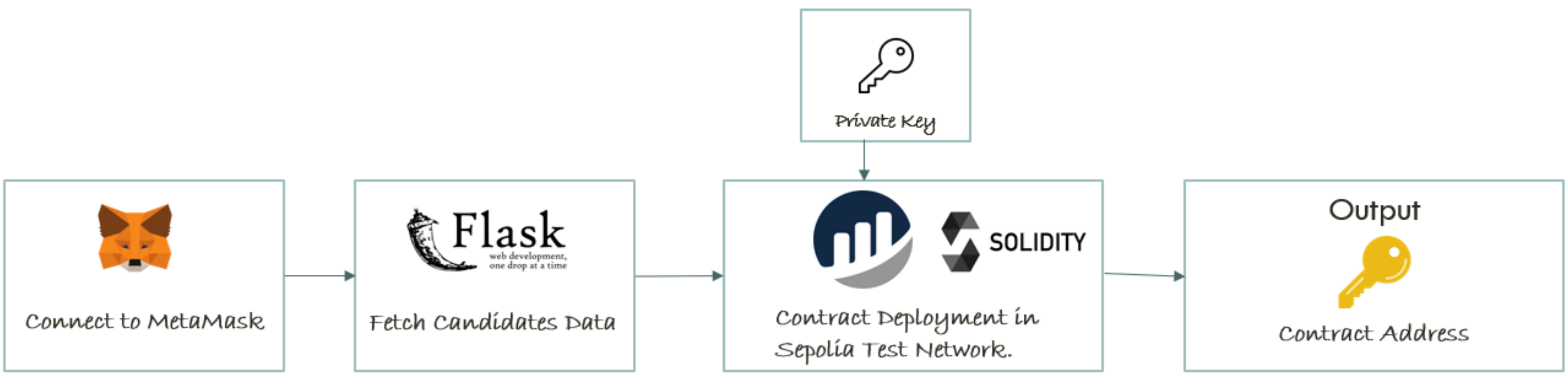
Methodology

Blockchain – Smart Contract

What is a Smart Contract?

A smart contract is a self-executing contract with the terms of the agreement directly written into code.

Once deployed to the blockchain, smart contracts automatically execute when predefined conditions are met, without the need for intermediaries.



Contract Deployment Process

Methodology

Blockchain – Smart Contract in Solidity

Data Structure in Smart Contract

Candidate:

- id
- voteCount

SOLIDITY

addCandidate()

- Adds a new candidate.
- Pushes it's id and voteCount as 0.

SOLIDITY

Used to add/deploy new Candidates into the Blockchain.

Used in VotingPage and called when selected a candidate and clicked on 'Vote'.

vote()

- Checks the ID received and will increment the voteCount.

SOLIDITY

getAllVotesOfCandidates()

- Returns all the candidates struct.
- This contains Candidate id and voteCount.

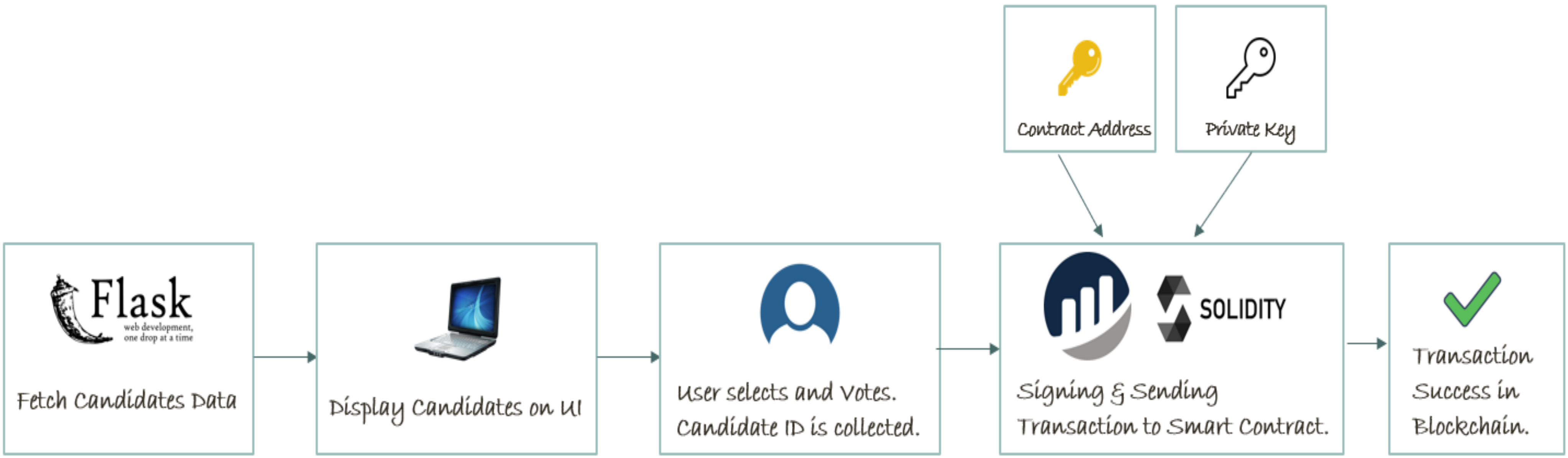
SOLIDITY

Used in the Results Page to retrieve the count of votes per each candidate.

Functions of Smart Contract

Methodology

Blockchain – Each Vote Flow

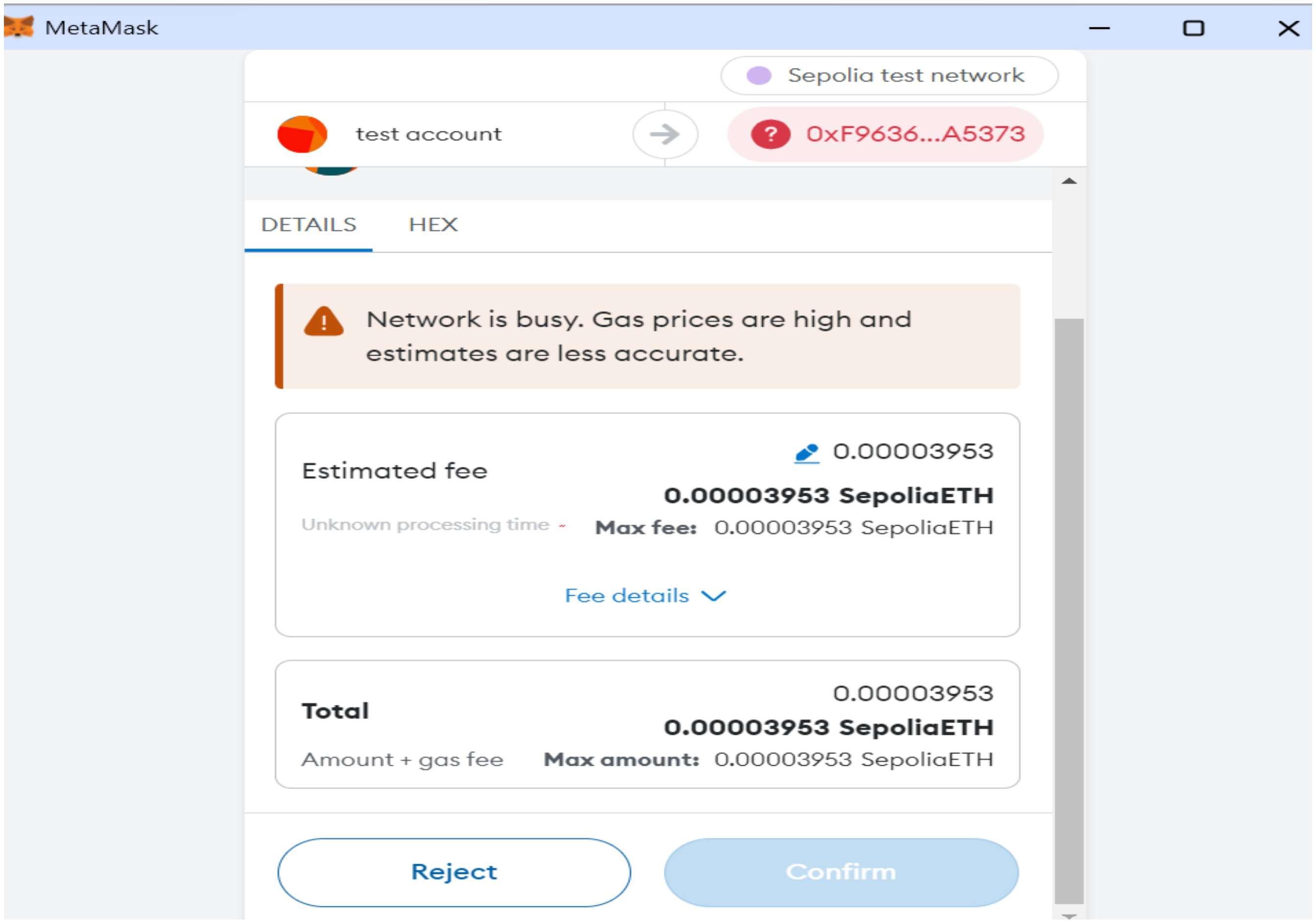


Vote Workflow into Blockchain

Blockchain - Results

Blockchain – Results

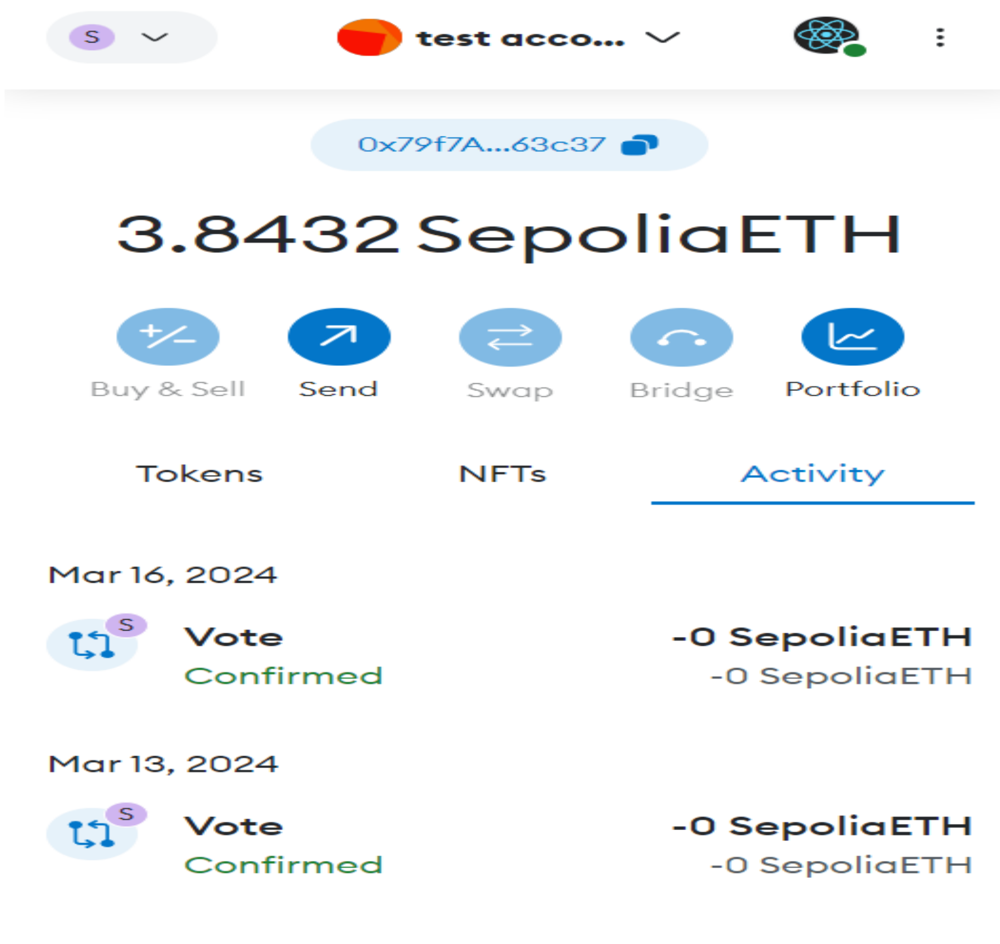
Sending the Transaction to MetaMask after Voting



Blockchain - Results

Blockchain – Results

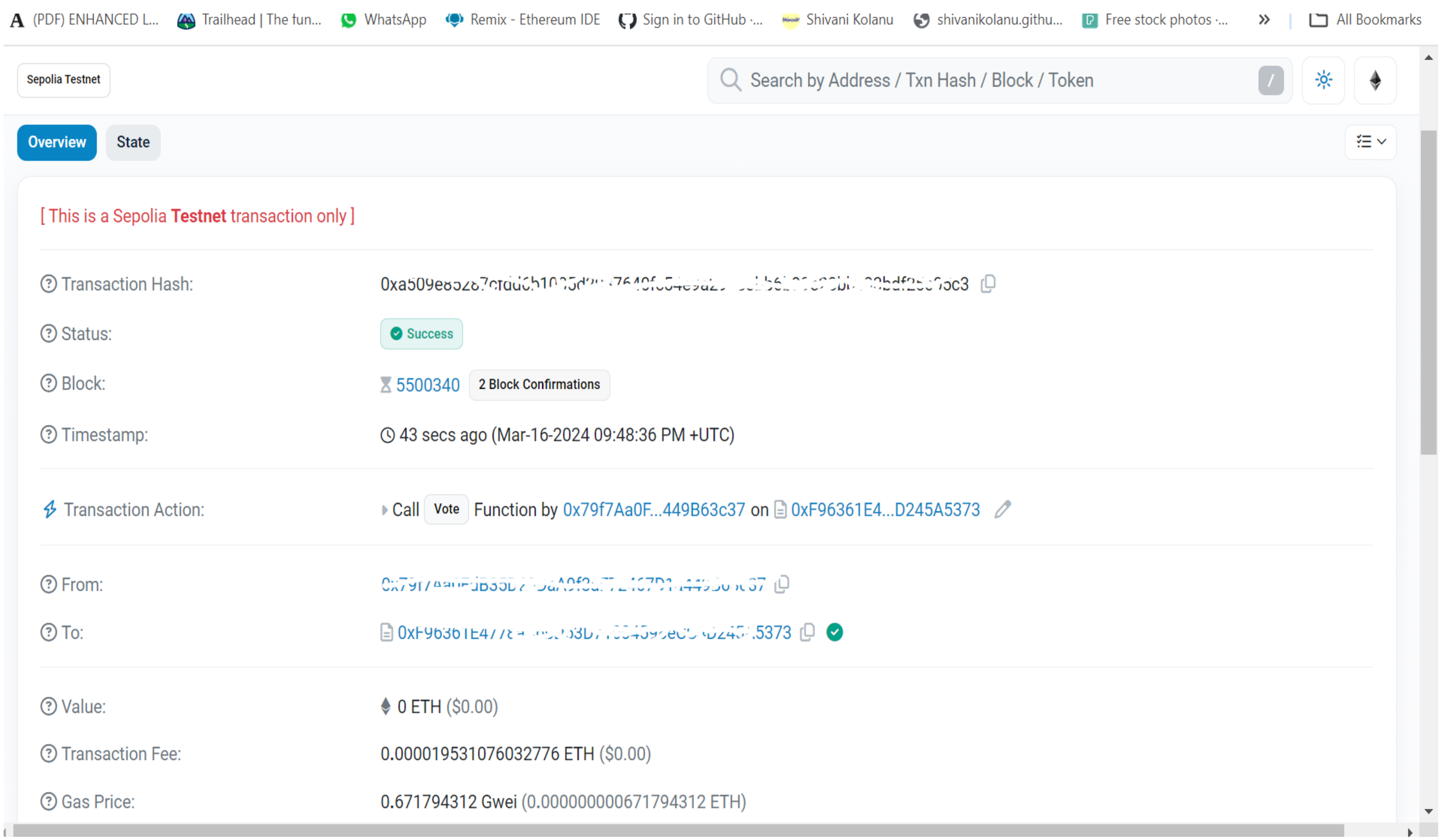
The Transaction is confirmed



Blockchain - Results

Blockchain – Results

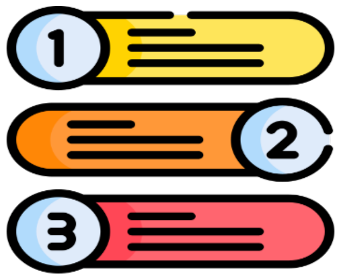
The Block created in Sepolia Test Network for the transaction data.



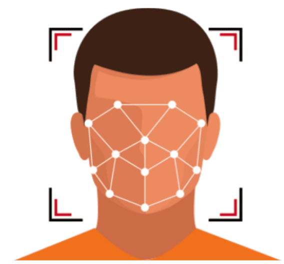
UI Methodology

UI – Technical Implementation Overview

React and Material-UI Integration



State Management for Multi-Step Process



Integration with Facial Recognition

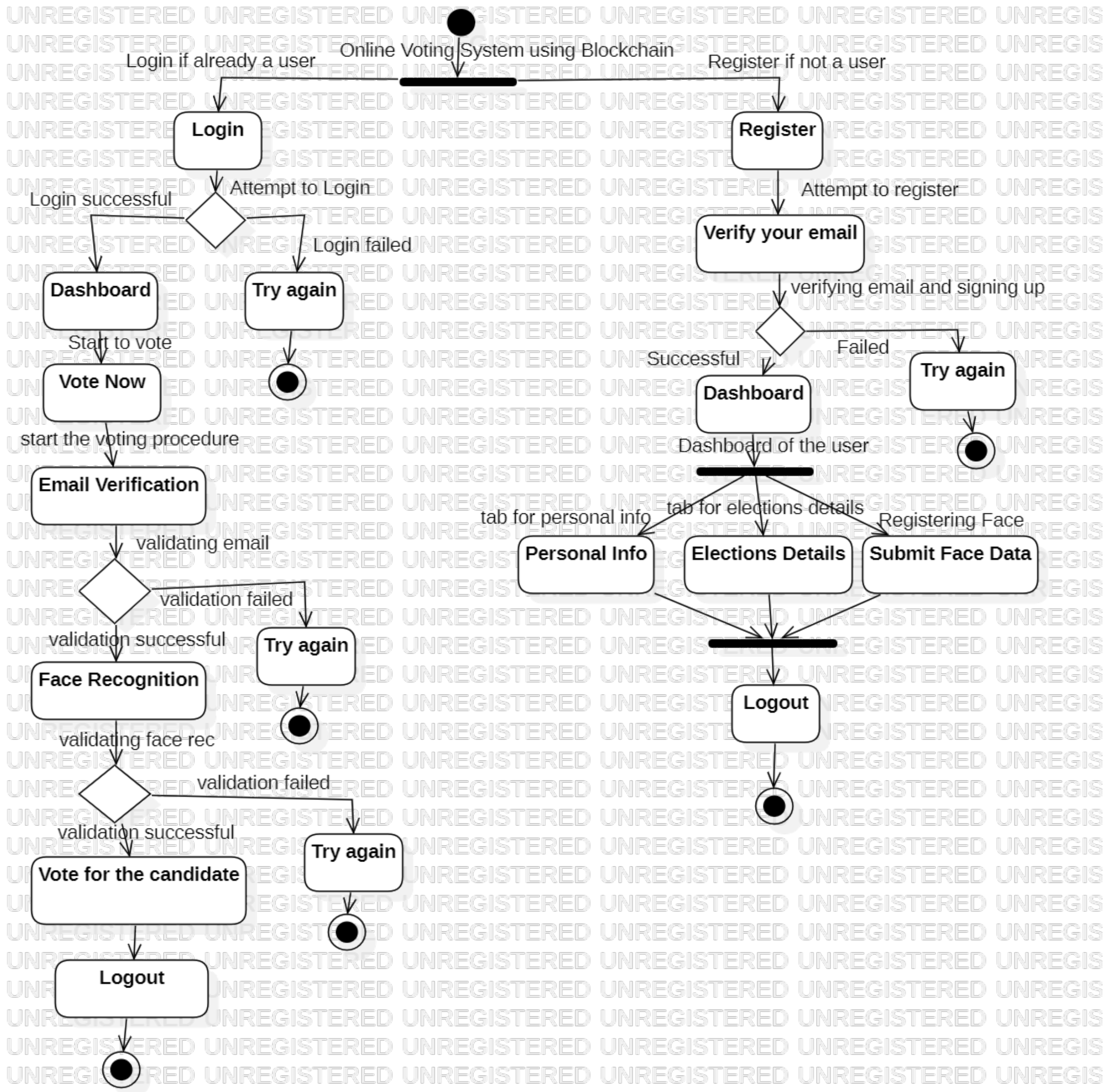
AXIOS

Handling Asynchronous Operations with Axios

User Feedback and Error Handling

Methodology

Activity Diagram for the UI Flow



Backend Methodology

Backend – Technical Implementation Overview

Flask Framework Integration

Database and Blockchain Integration

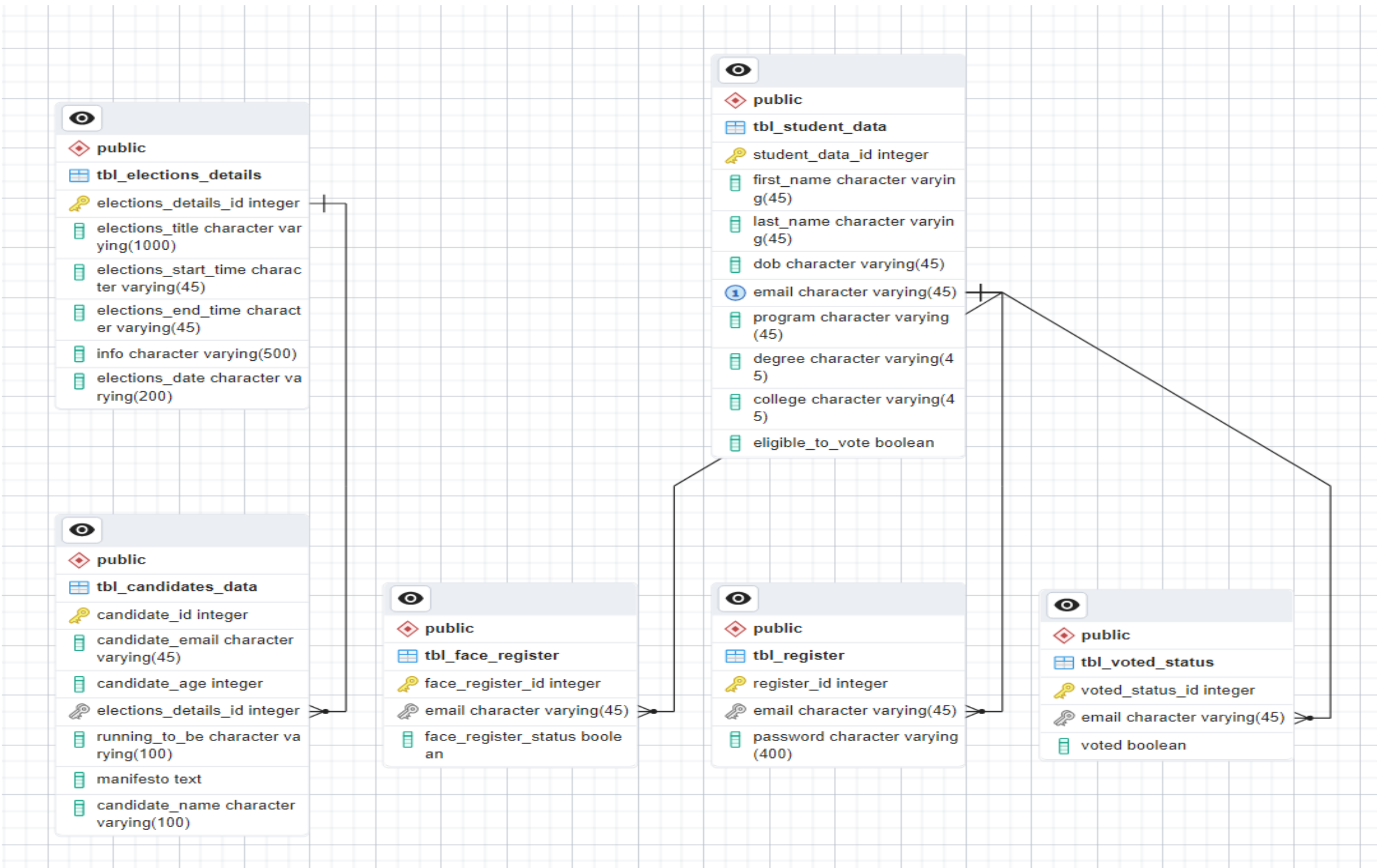
Email Verification System

Security and Session Management

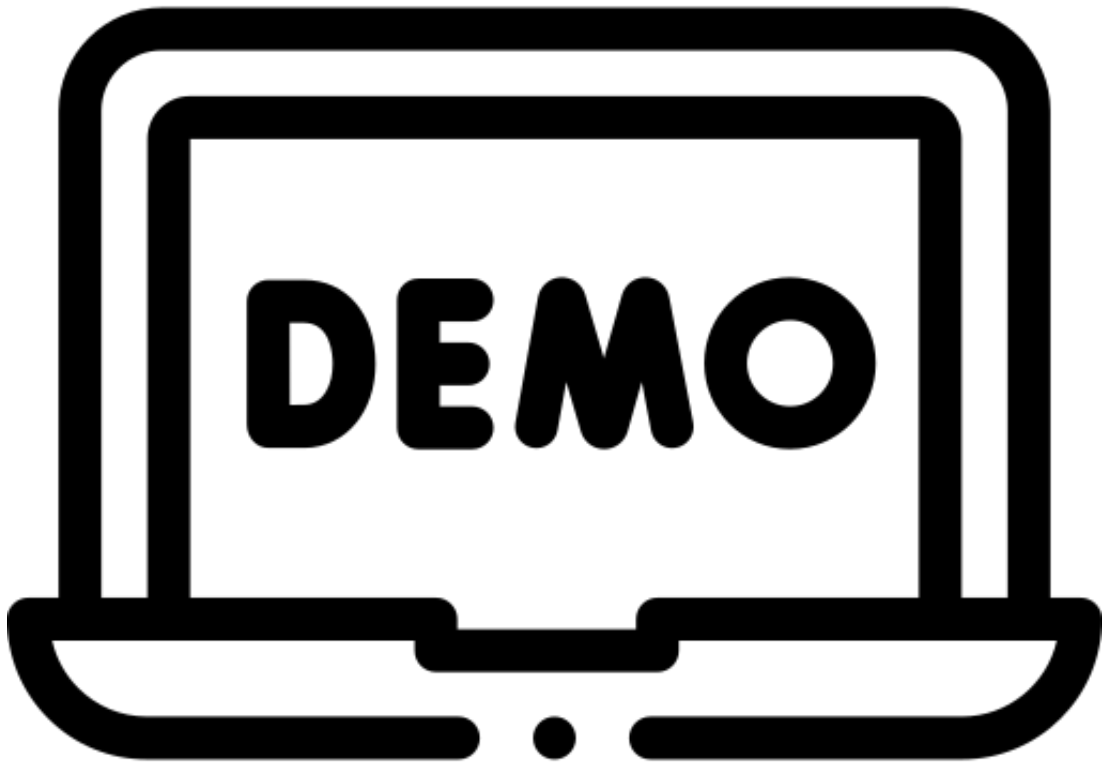
Machine Learning Integration

Methodology

Backend – E-R Diagram of the Database



Demo of the Project



Conclusion

- Online voting system leveraging blockchain technology revolutionize the electoral process, offering convenient and accessible voting options for citizens from any location with internet access.
- The integration of blockchain ensures security and transparency, safeguarding the integrity of the voting process by providing tamper-proof storage of votes.
- Robust authentication mechanisms, such as two-step verification and face recognition, enhance the system's ability to verify voter identities, fostering trust and confidence in the electoral process.
- Continuous innovation and adaptation to technological advancements are essential for the ongoing improvement and relevance of online voting systems in modern democratic societies.

Thank You!