BLOCKCHAIN BASED E-VOTING SYSTEM

SECOND SEMINAR

SUPERVISORS

Dr. Dina Elsayad

TA. Manar Sultan

TEAM MEMBERS

Suhail Mahmoud

Heba Shaaban

Abdelrahman Hamdi

Rana Ahmed

Abdelrahman Osama

Habiba A. Abdulkader

AGENDA

01 Problem Definition

02 Motivation

03 Objective

04 System Architecture

05 Data Creation

06 Face Verification

07 ID Data Extraction

08 Fake ID Detection

09 Blockchain

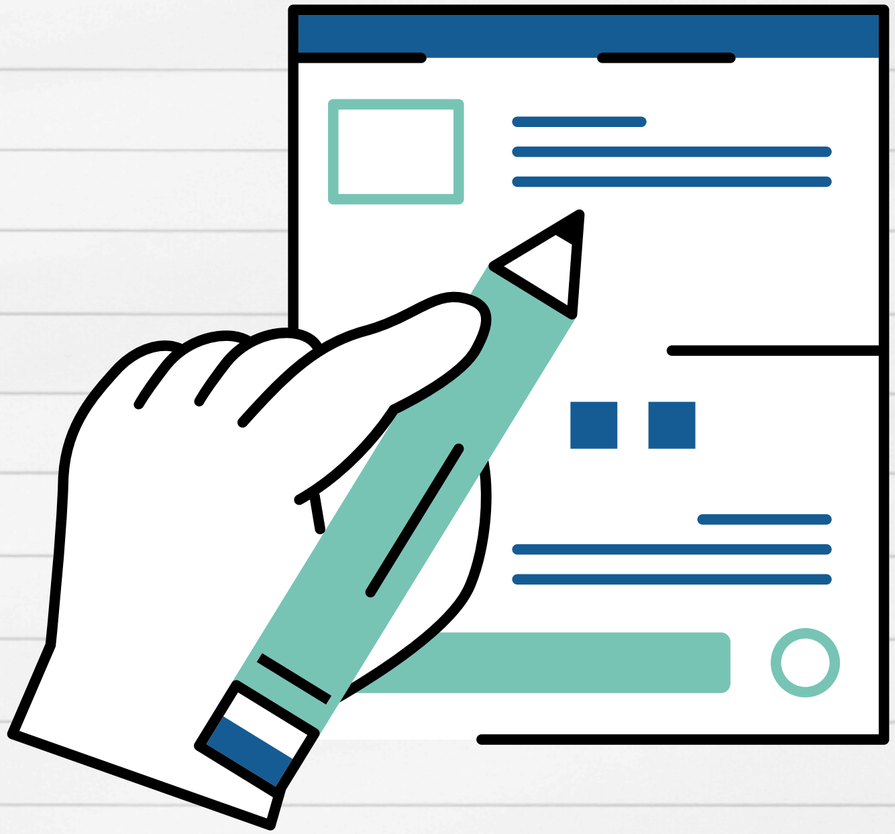
10 Frontend

11 Time Plan

12 Next Phase

01

PROBLEM DEFINITION

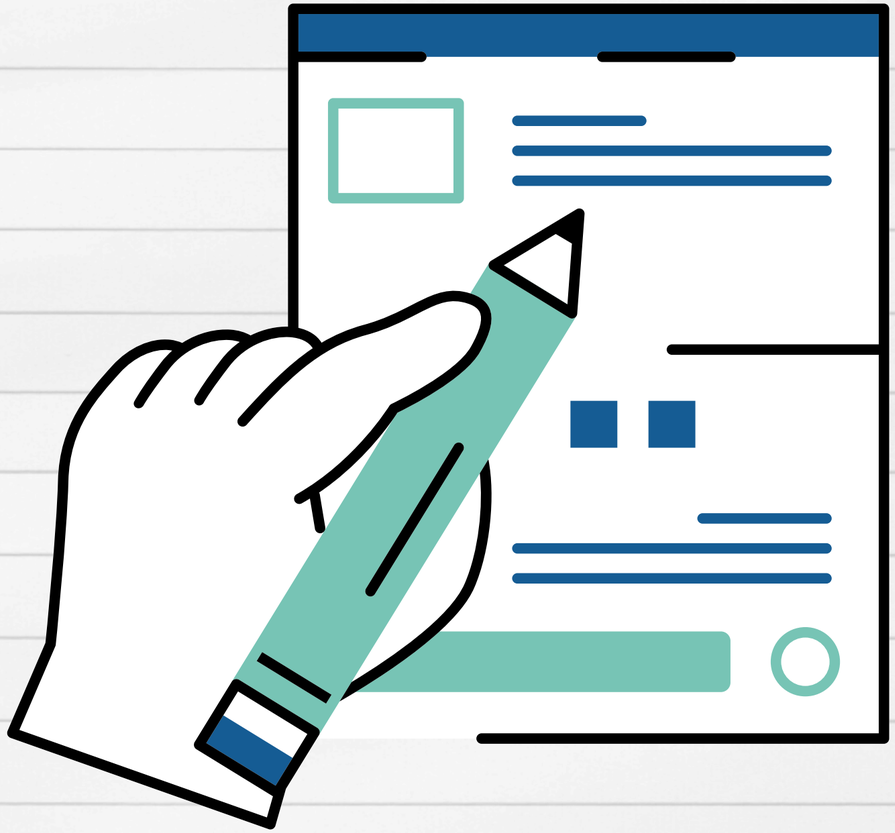


PROBLEM DEFINITION

Current voting systems face various security and transparency issues, including fraud, manipulation, and lack of trust from the public. Traditional systems make it challenging to securely identify voters while keeping their identities private. These vulnerabilities highlight the need for a reliable, decentralized solution to ensure vote integrity, transparency, and user privacy.

02

MOTIVATION

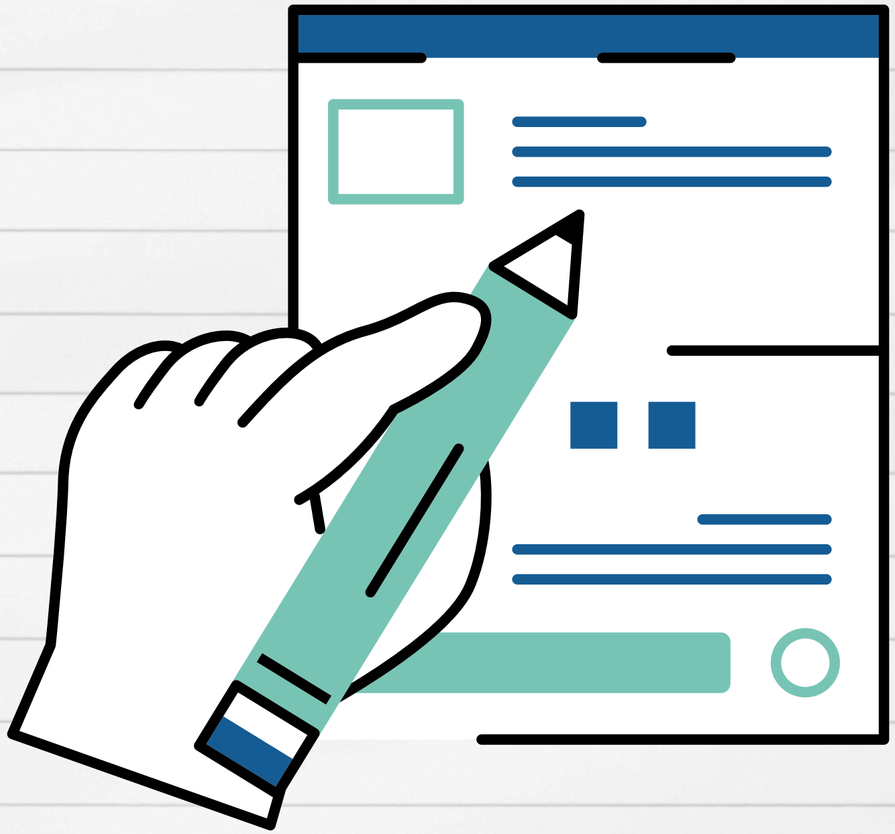


MOTIVATION

The main motivation is to build a secure and transparent e-voting platform that leverages blockchain's decentralized structure to provide an immutable record of votes. Additionally, it utilizes AI technology for user authentication, ensuring that only verified users can participate and that each vote remains anonymous and unaltered.

03

OBJECTIVES

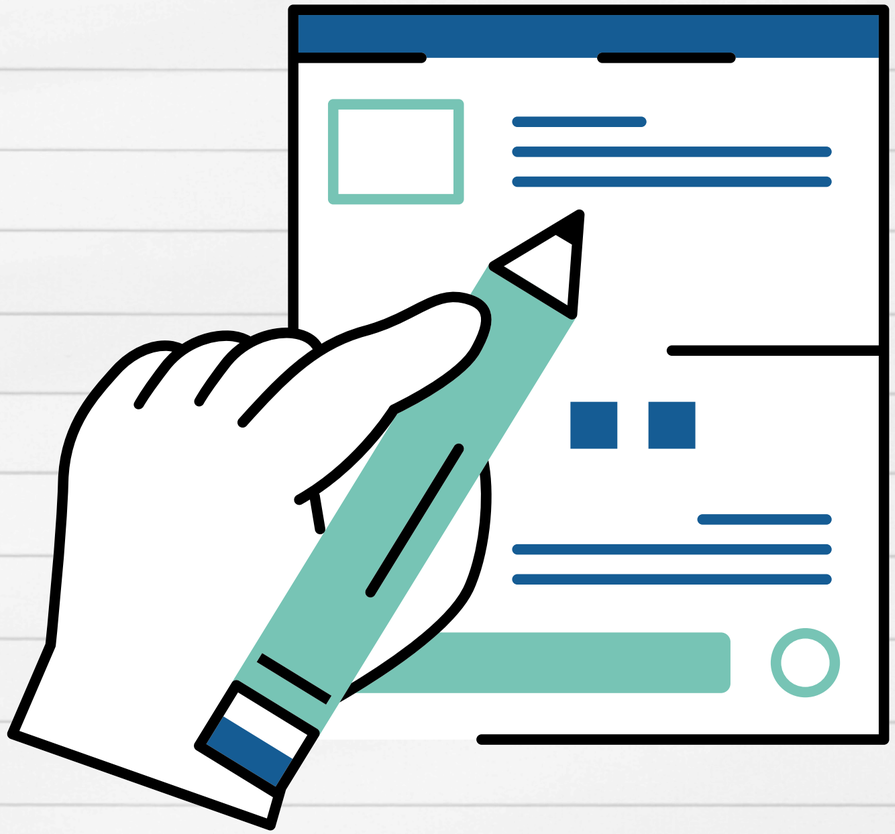


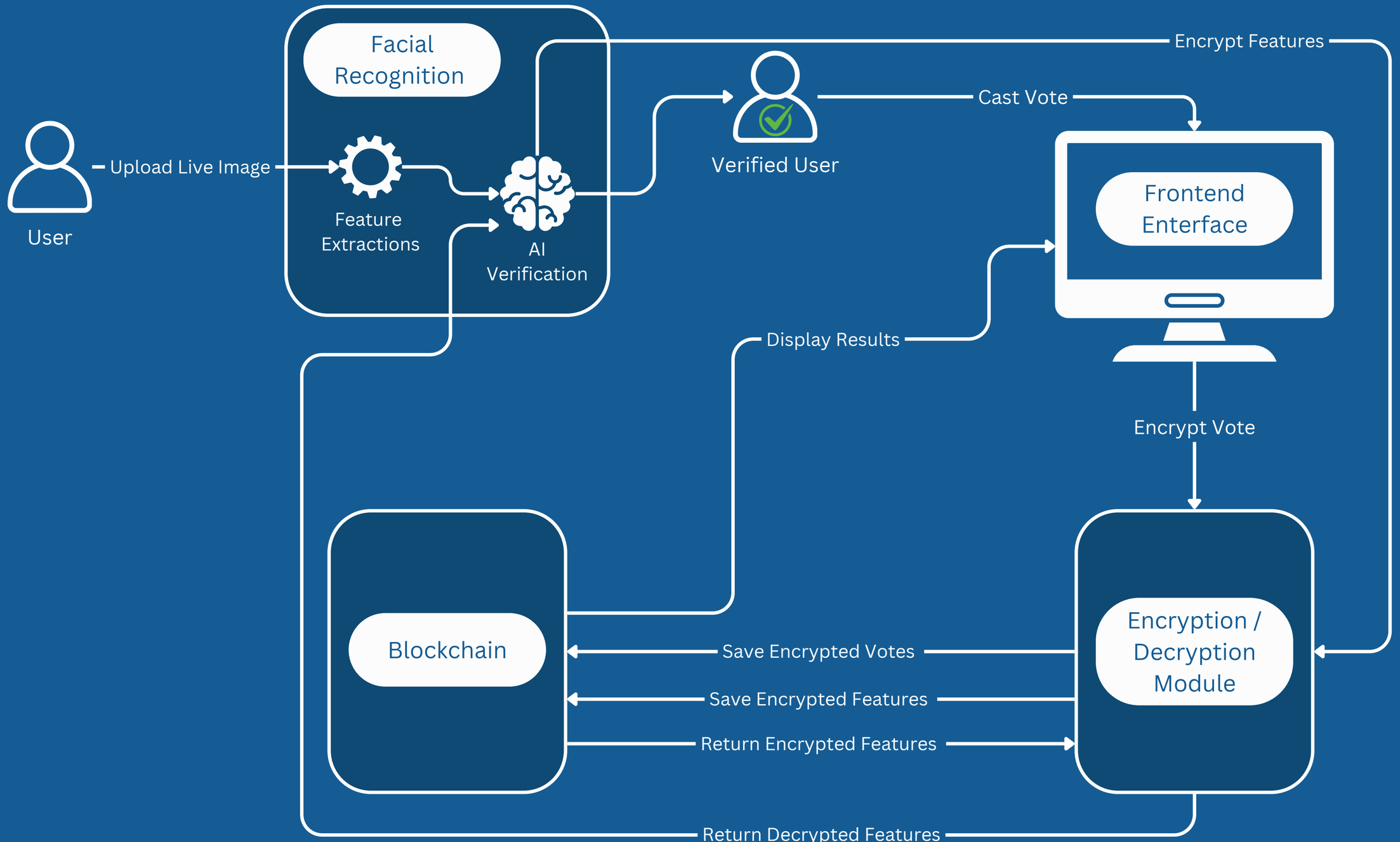
OBJECTIVE

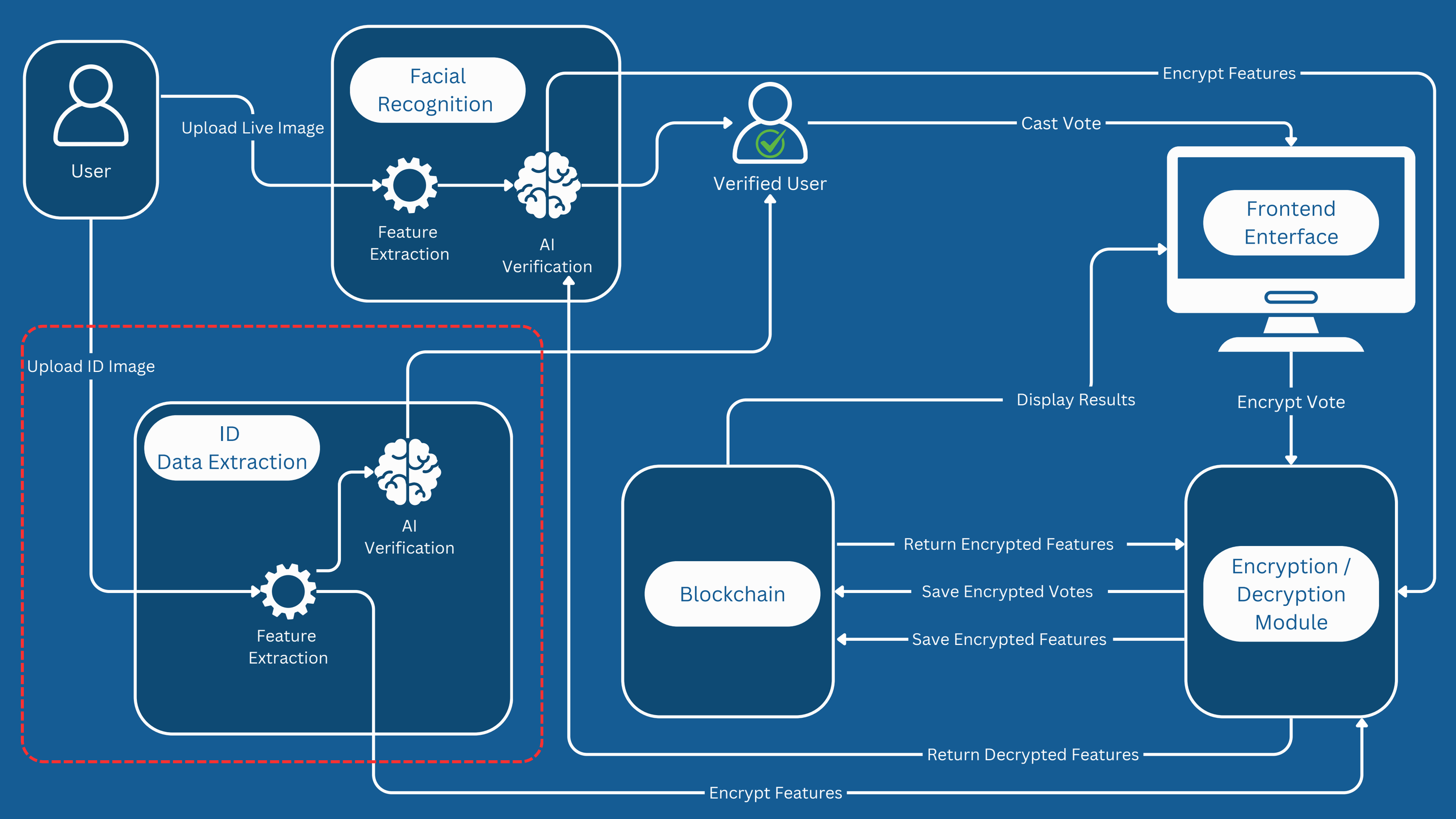
- The main goal of this project is to develop an online voting system with blockchain as its core foundation and AI-powered identity verification. This system will :
  - Ensure privacy and trust in voting processes, benefiting governments, organizations, and private entities seeking secure voting solutions.
  - Enable real-time, auditable, and transparent voting records.

04

SYSTEM ARCHITECTURE

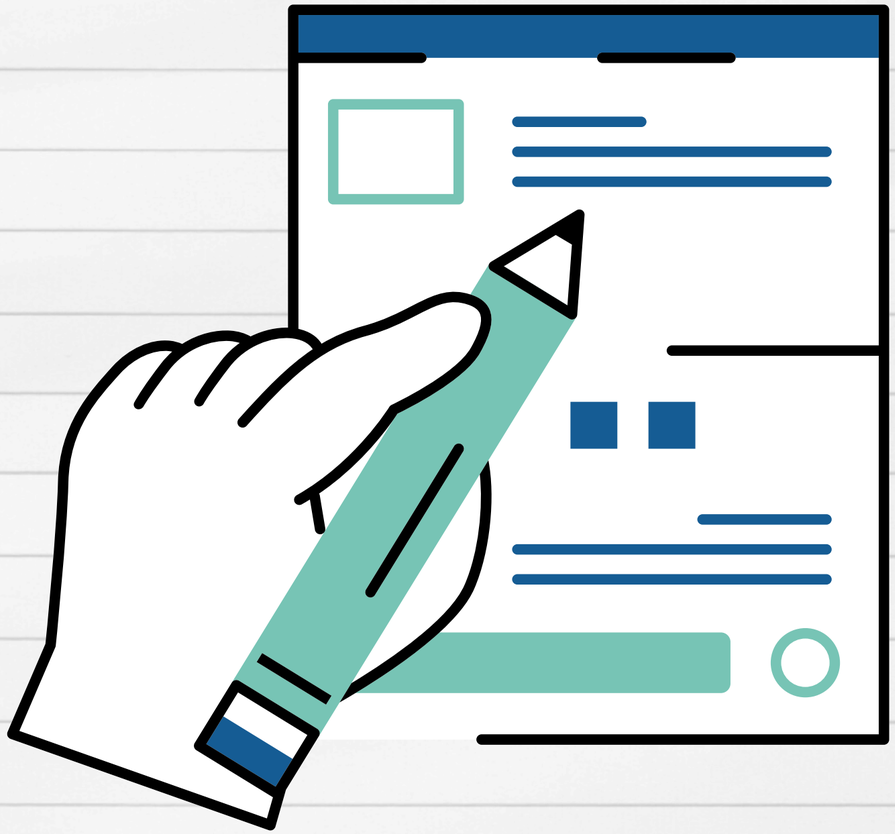



Facial Recognition
User
Upload Live Image
Feature Extractions
AI Verification
Verified User
Encrypt Features
Cast Vote
Frontend Enterface
Display Results
Encrypt Vote
Blockchain
Save Encrypted Votes
Save Encrypted Features
Return Encrypted Features
Encryption / Decryption Module
Return Decrypted Features


User
Upload Live Image
Facial Recognition
Feature Extraction
AI Verification
Verified User
Encrypt Features
Cast Vote
Frontend Enterface
Upload ID Image
ID Data Extraction
AI Verification
Feature Extraction
Display Results
Encrypt Vote
Blockchain
Return Encrypted Features
Save Encrypted Votes
Save Encrypted Features
Encryption / Decryption Module
Return Decrypted Features
Encrypt Features

05

DATA CREATION



DATA CREATION

- **WHAT IS THAT?**
  - we was need to collect and create Egyptian national ID Dataset.
- **WHY?**
  - **Security:** Verifying the user’s national id, we can prevent impersonation and unauthorized access.
  - **Integration:** AI verification integrates seamlessly with the blockchain to reducing administrative overhead.

DATA CREATION

- **HOW?**
  - First, by Collecting all data found , Then the stage of filtering start by remove all redundant data and only keep the **Unique** data
  - sites that found data on it (roboflow, facebook, google, github).
  - after collecting, we got over **32K** image.
  - after filtering, we got only **8,649** Unique Real Images (probably).

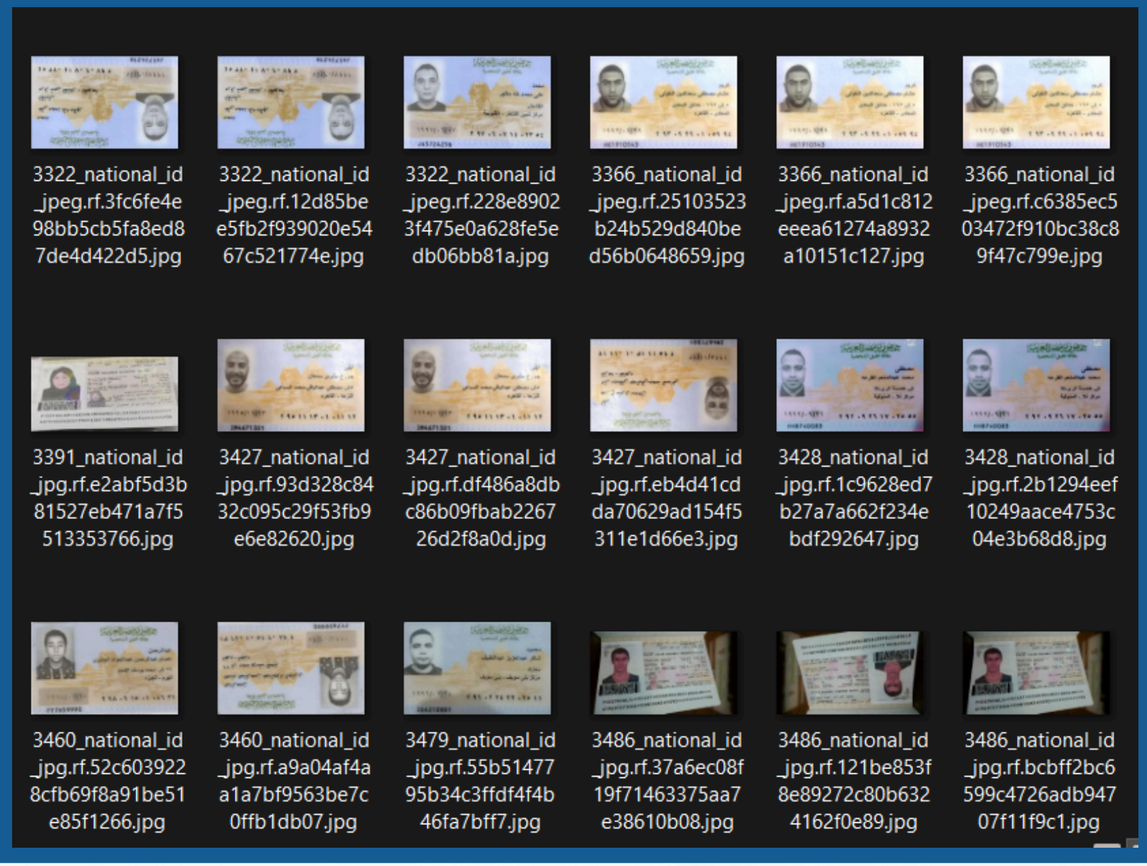
DATA CREATION

- **Filtering**
  - **First**, some data filtered **manually**.
  - **Second**, using some **regex** technique for names of images.
  - **Third**, using some some **CNN** Network like..
    - Simple CNN Network.
    - MobileNetV2.

    “of course with **manual** review”.

DATA CREATION

Before



After

DATA CREATION

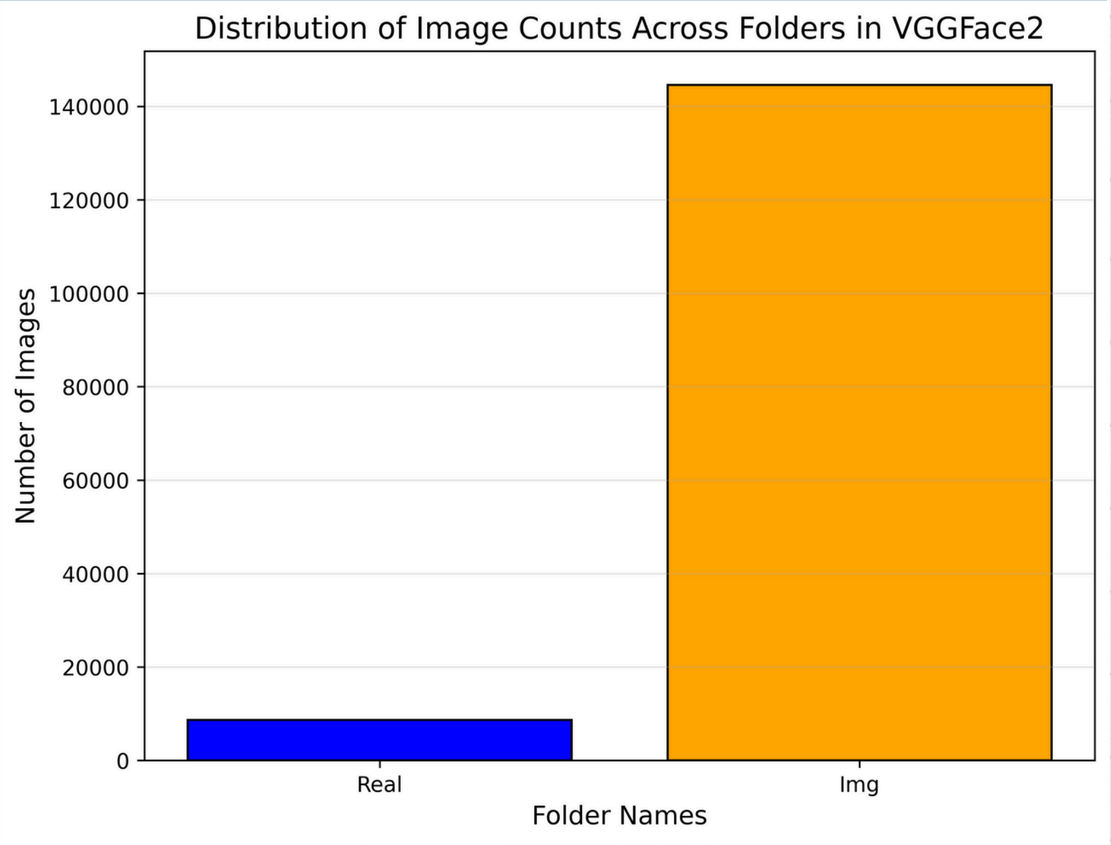
we collect the Unique identity from data

- VggFace2 (8631 identity).
- CelebA (10177 identity).
- Synthetic Faces High Quality (SFHQ) part 4 (125K identity).

DATA CREATION

DATA CREATION

- SO with **8649** real and over **144K** Img.
- we had over **153K** Egyptian ID Card.
- it is now on **Kaggle**.



DATA CREATION

Here We Go

ABDELRAHMAN HAMDI AND 3 COLLABORATORS · UPDATED A MONTH AGO · PRIVATE

0 | Code | Download

Egyptians IDs Dataset

Dataset for Egyptians ID Front Cards Real and Synthetic

Data Card | Code (0) | Discussion (0) | Suggestions (0) | Settings

Egyptians IDs DataSet (2 directories)

About this directory

Img → Synthetic Data (144568) , **Real** → Real Data (8658) (Make sense)

Img
145k files

Real
8658 files

Data Explorer

Version 1 (8.09 GB)

- Egyptians IDs DataSet
  - Img
  - Real

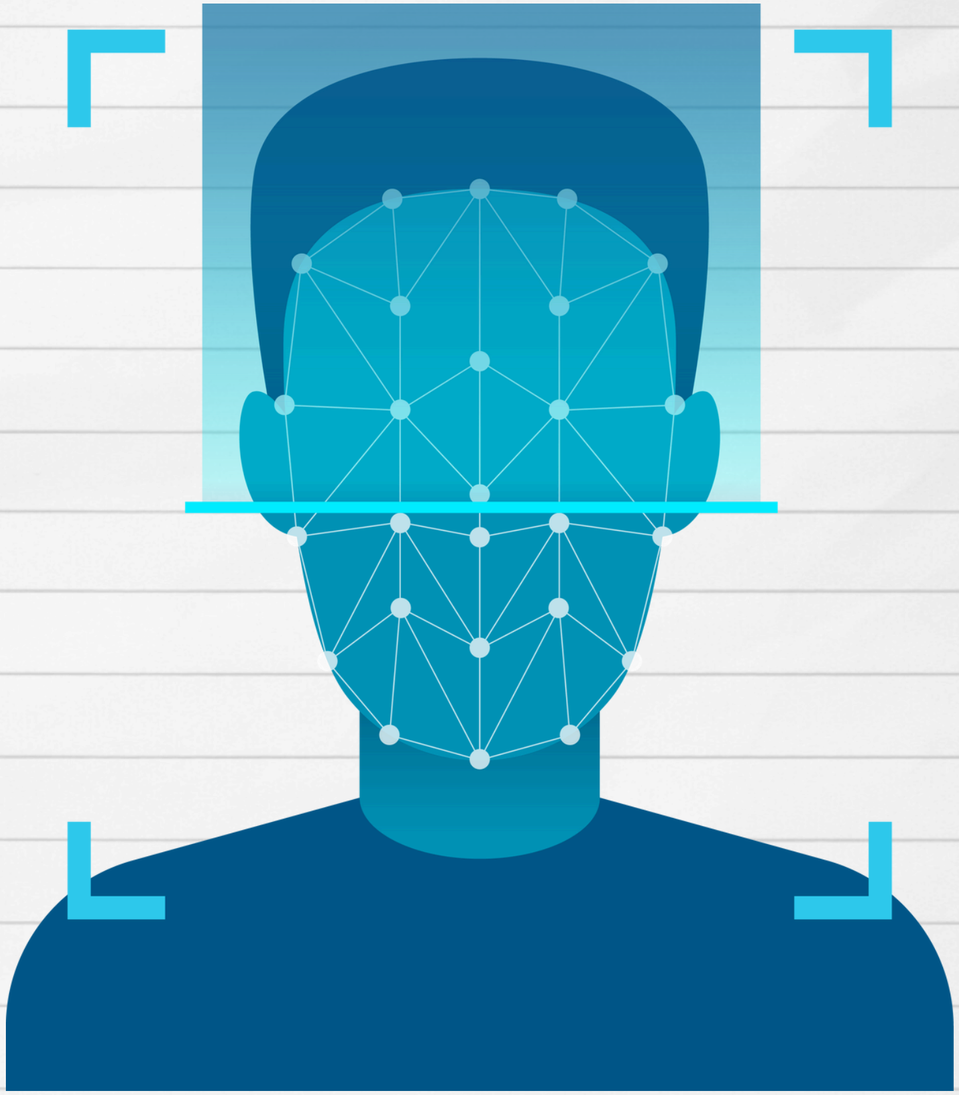
Summary

153k files
.jpg 153k

+ New Version

06

FACE VERIFICATION



FACE VERIFICATION

- **WHAT IS THAT?**
  - AI-based facial recognition module will ensure voter authentication.

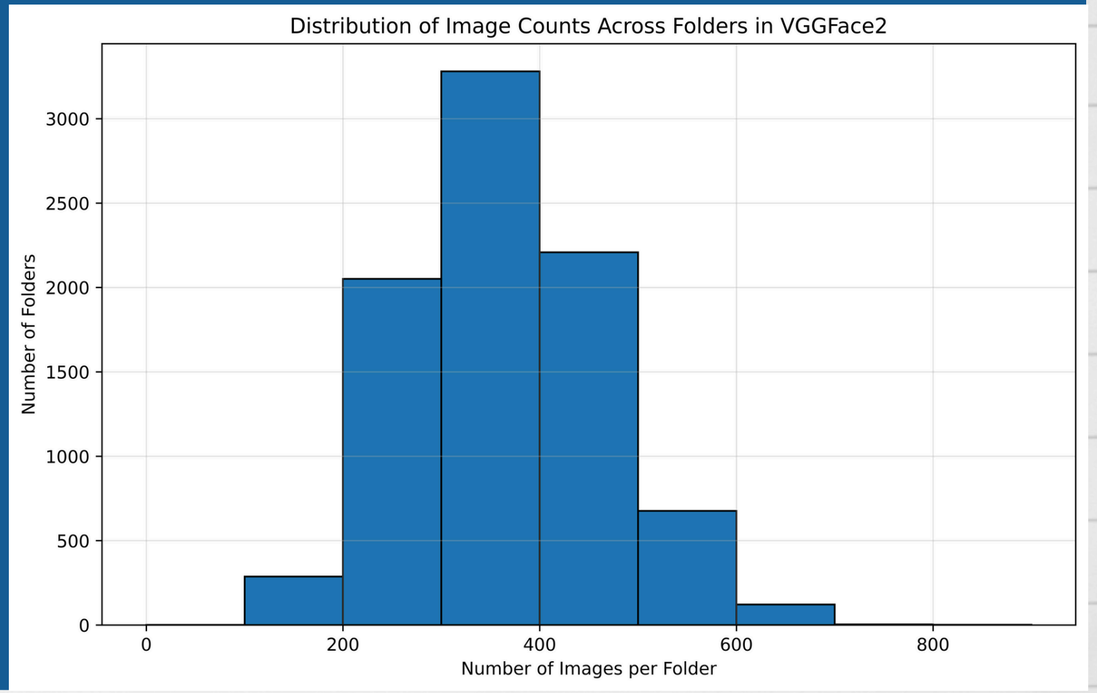
Dataset: VGGFace2

- **Overview**
  - Includes over 3.3 million images.
  - Contains over 9,000 identities.
  - Diversity in terms of pose, ethnicity, and lightning conditions.

- **Features**
  - **Pose Variation:** Includes faces captured at various angles.
  - **Expression Diversity:** Includes images with a wide range of facial expressions.
  - **Backgrounds:** Includes images with varied backgrounds to improve generalization.

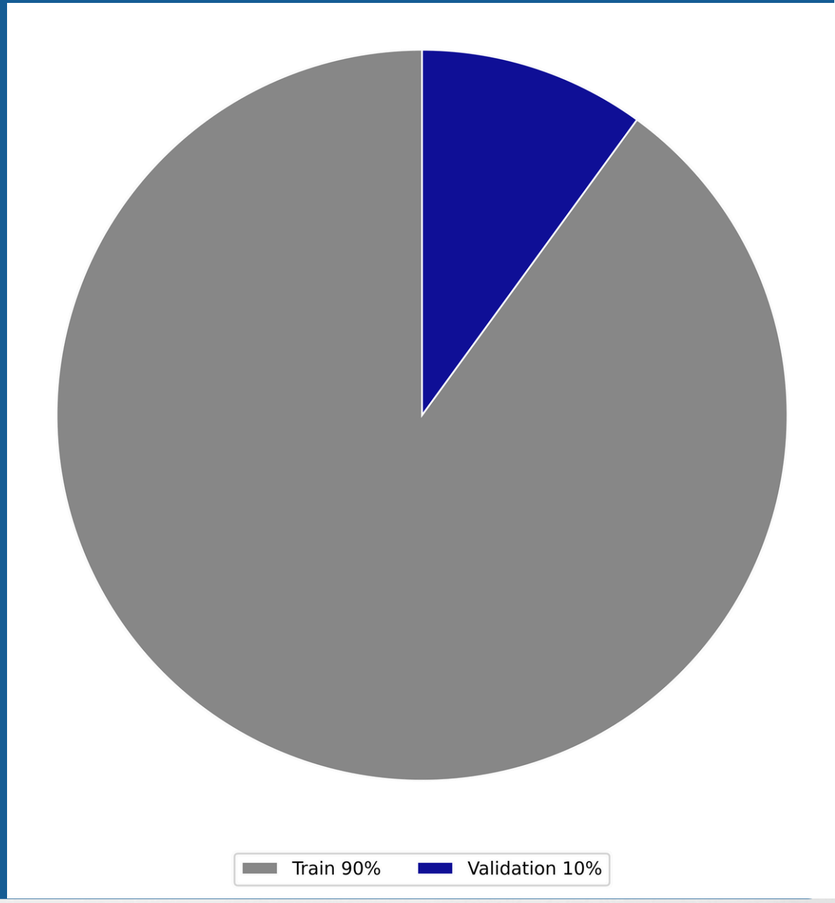
Analysis

- Total Number of Identities : 8631
- Total Number of Images : 3141890 (3.14M)
- Min images per folder : 87
- Max images per folder : 843
- Average images per folder : 364.02



Preprocessing

- Split
  - Train : 7767 (90%)
  - Validation : 864 (10%)
- Train
  - Detect Face (MTCNN)
  - Resize (160 x 160)
  - Augmentation
    - Horizontal Flip
    - Rotation
    - Color
  - Tensor
  - Generate Triplets

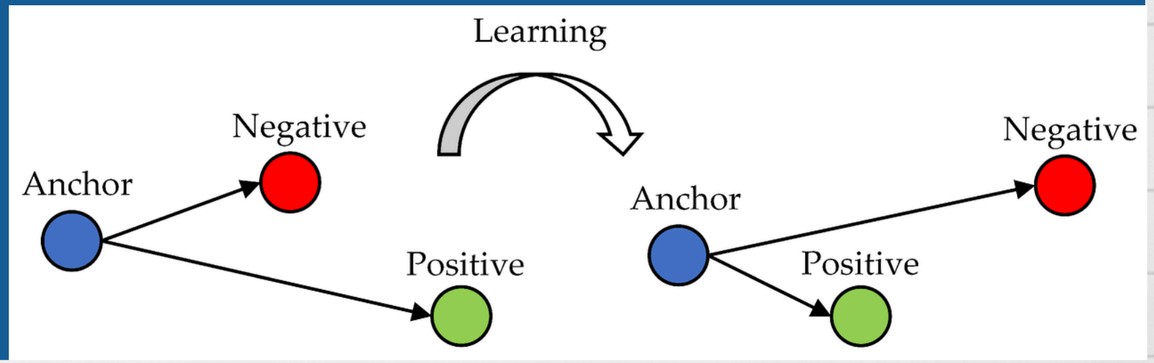


Preprocessing

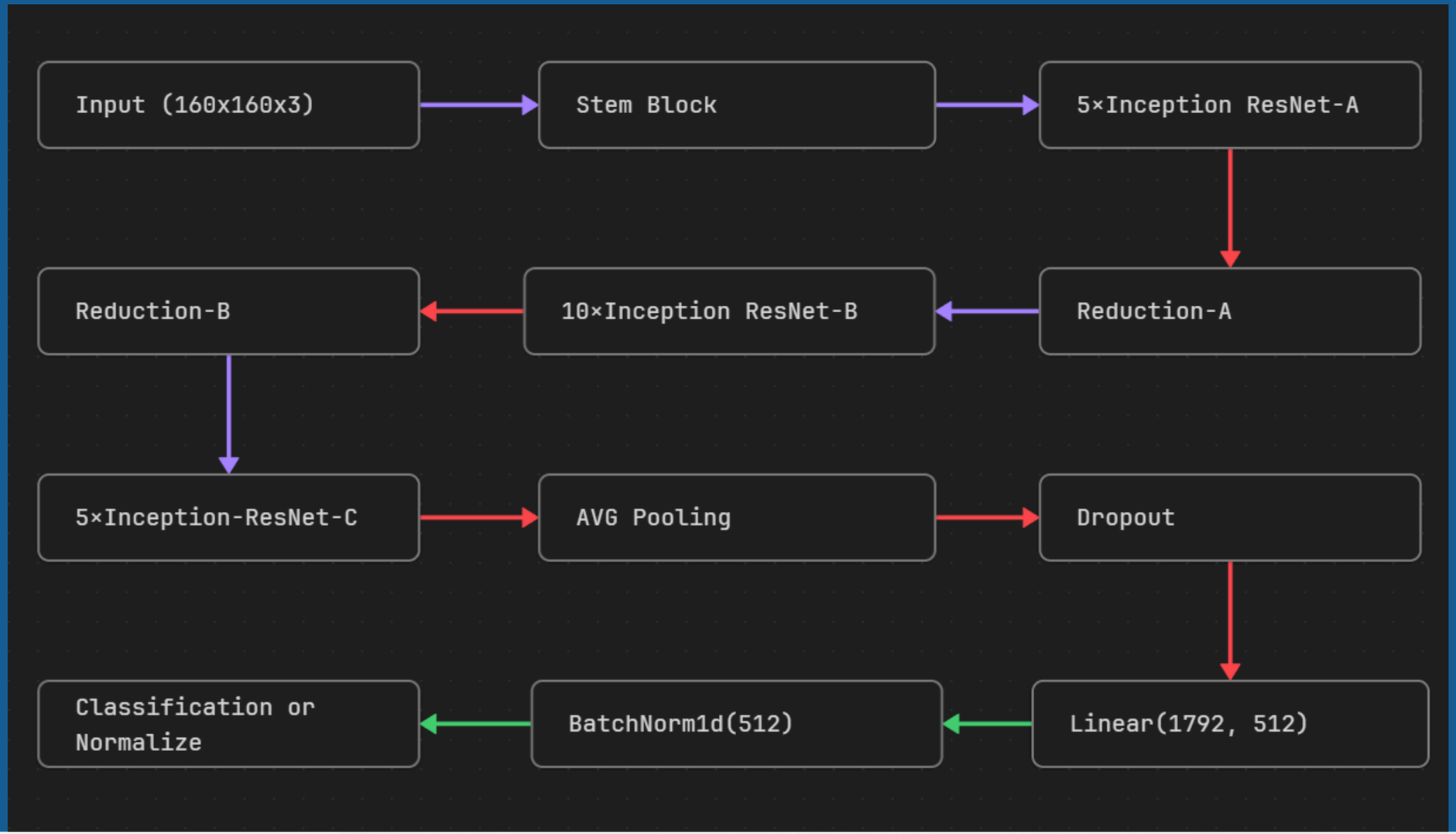
- **Test**
  - **Detect Face (MTCNN)**
  - **Resize (160 x 160)**
  - **Generate Pairs**

Triplets

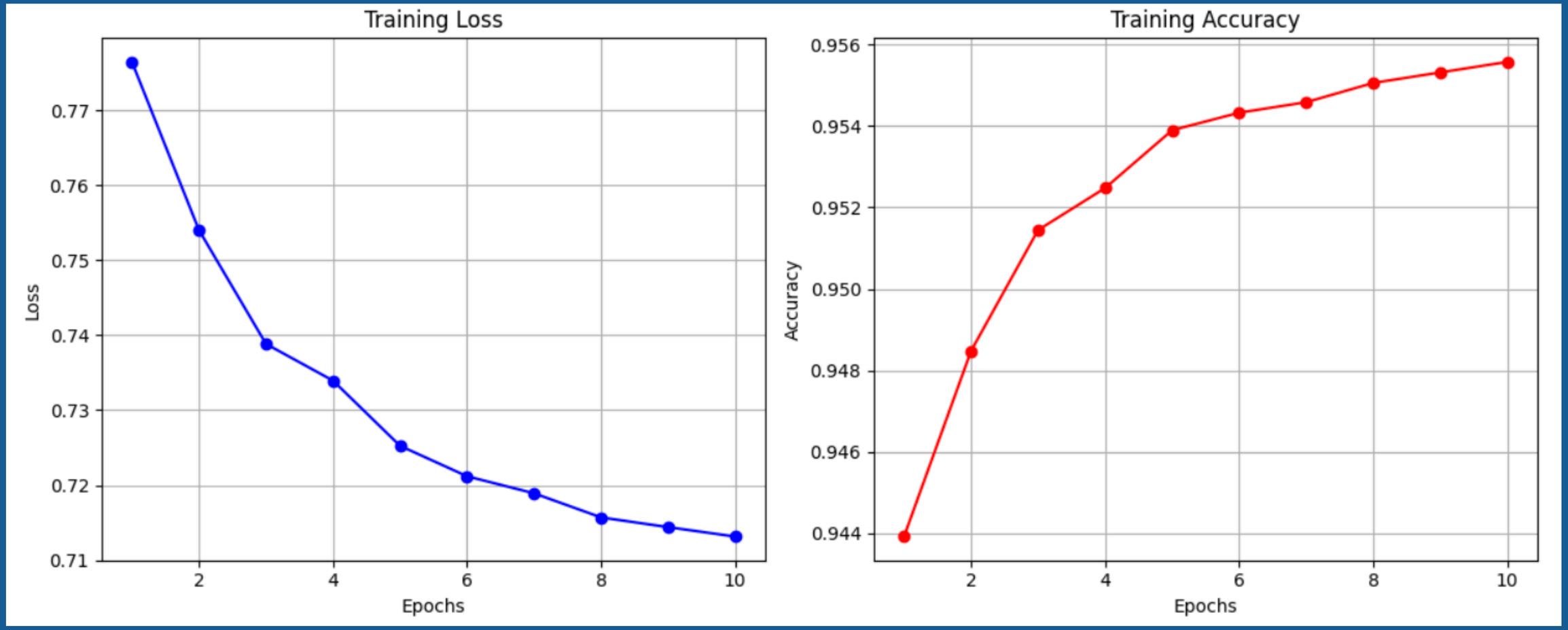
- **Anchor** , **Positive** from the **SAME** Class
- **Negative** from **different** Class



Model: InceptionResnet-V1



Result

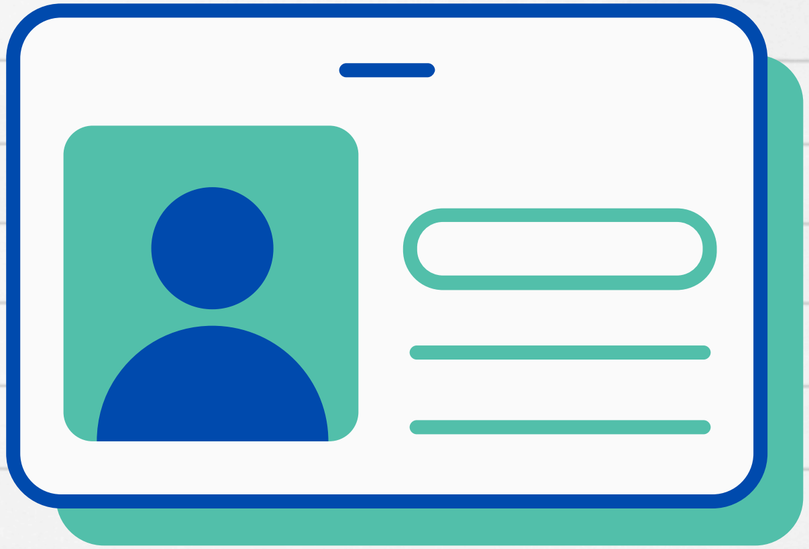

Training Loss
Loss
0.77
0.76
0.75
0.74
0.73
0.72
0.71
2
4
6
8
10
Epochs
Training Accuracy
Accuracy
0.956
0.954
0.952
0.950
0.948
0.946
0.944
2
4
6
8
10
Epochs

Result

Model	VAL_Accuracy	Test_Accuracy	Precision	Recall	F1-Score
Resnet50	77	62.91	33.50	79.09	47.07
Inception Resnet-V1	90.28	95.13	85.67	92.03	88.74

08

ID DATA EXTRACTION



ID VERIFICATION

Why Do We Need ID Verification?

Problem:

- In digital voting systems, anyone can enter a national ID number manually.
  But...
  ➡ How do we know if this number really matches the person's ID?

ID VERIFICATION

🎯 Our Solution:

- We detect and extract the national ID number directly from the uploaded ID image (using our YOLOv8 models and OCR), then:

🟡 Compare it with the manually entered ID number


DATA
XX01
XX02
XX03
01011011

DataSet:

- **Dataset Name:** **Egyptian-ID-Dataset**
- **Sources:** **Roboflow**
- **Usage Breakdown:**
  - for YOLO Model 1 (ID Detection): 4200 images to detect ID cards within larger images
  - for YOLO Model 2 (Field Detection): 150 images to detect fields like Name, ID Number, Birthdate, etc.

Pre Processing

01

ID Detetction
DataSet
(4200 image)

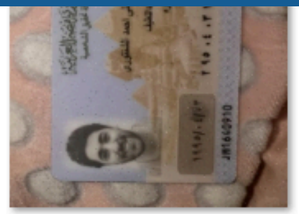
02

Fileds Detection
DataSet
(150 image)

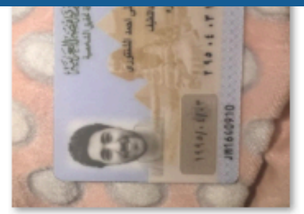
Pre Processing & Augmentation

Brightness Adjustment	Randomly between **-20% to +20%**
Exposure Adjustment	Randomly between **-5% to +5%**
Gaussian Blur	Random blur between **0 to 0.5 pixels**
◯ **Salt & Pepper Noise**	Applied to **0.5% of pixels**

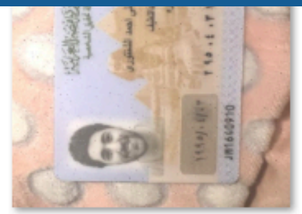
Augmentation Real Example



59_3_jpg.rf.1bb4db02ad64e534e030a802daac05e5.jpg



59_3_jpg.rf.3363385d940a5a6a01289ea89ff0da8a.jpg



59_3_jpg.rf.3541492457facfb18658da01c9f1e8a1.jpg

59_jpg.rf.121fd3603593de40de2c25ee7a917bad.jpg

59_jpg.rf.743c11341c4515a0552e4a90462b8611.jpg

59_jpg.rf.ef6b53d2baf545e0e96b97175d3217c3.jpg

60_1_jpg.rf.f96ff1250cdf07bba2e5ff24e086254f.jpg

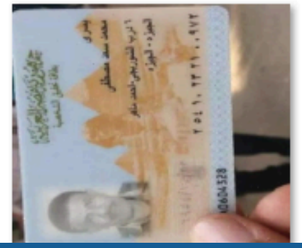
60_2_jpg.rf.3a100d1afa7d34de578eebdf00a8b683.jpg

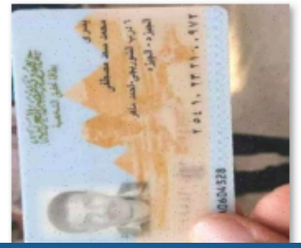
60_2_jpg.rf.8bbf9a835d98cb684a5d2092163209c1.jpg

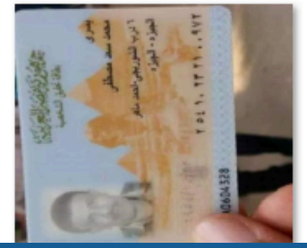
60_2_jpg.rf.c01e63bcbad9cc289f71cc458328e9d3.jpg

60_3_jpg.rf.004c4db0bbd51e640313d731459780d8.jpg

60_3_jpg.rf.144e223f04e118d3286129f20bba836c.jpg







Pre Processing

01

ID Detetction
DataSet
(4200 image)

02

Fileds Detection
DataSet
(150 image)

No Augmentation Needed

5_jpg.rf.3dc54f6d8f2110a5fdb40e4005191da1.jpg

6_jpg.rf.0891adb594818dfc3e1fcd46af579f2a.jpg

7-2-_jpg.rf.8fcfdf433f9308177f4b8585afcebdb7.jpg

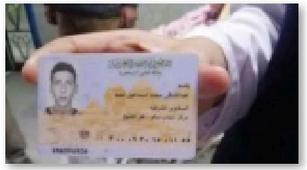
8_jpg.rf.d3c7b54a948ae315525435d4bd1df50c.jpg

8-2-_jpg.rf.a966c180cfdb1eeadc9fe3a91379b826.jpg

12-2-_jpg.rf.8968 9fa56748647fd7b8106796e65a4f.jpg

13_jpg.rf.403bd2593350506de760d079da39b1d0.jpg

13-2-_jpg.rf.f8e1486a832362f51ecc0c8ad15d0cd9.jpg



14_jpg.rf.126a300f6b22b485518402189245bcbf.jpg

15_jpg.rf.1bcce9eeb9e9494188bbc297288cfc39.jpg

18_jpg.rf.1856127130f5dd95ca943403d9bd4f94.jpg

18-2-_jpg.rf.e7d0fc78afb8cedef1c30994338b33c6.jpg

19-2-_jpg.rf.26e827072363a62caa389be9842b4433.jpg

21_jpg.rf.f79cd6dd934e95e46e743306659d77fa.jpg

22_jpg.rf.bd5c4651d92bbd458065937a427956b0.jpg

Datasets Pipeline Using Roboflow

Labeling using Roboflow Tools

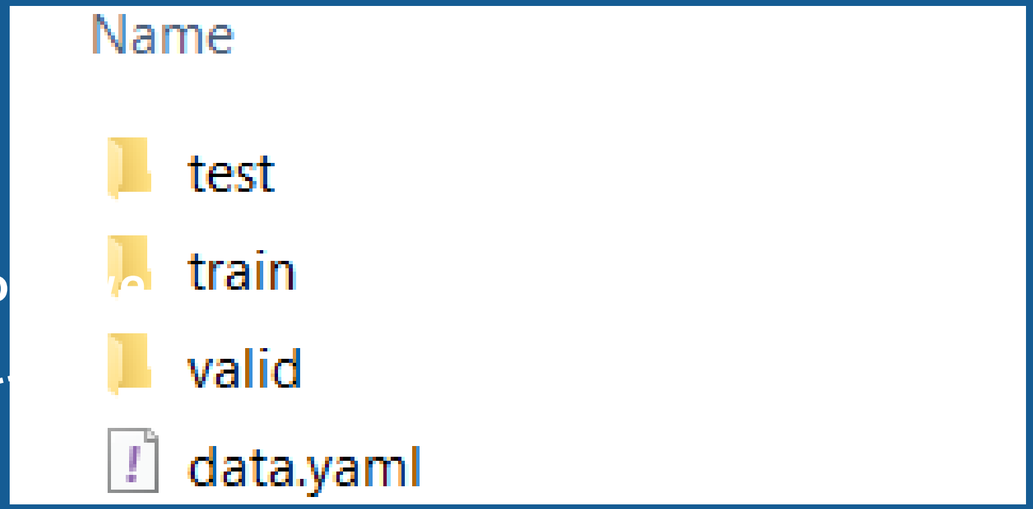
File Edit Format View Help

```
7 0.735830078125 0.38463988919667585 0.32576171875 0.15905817174515235
0 0.71365234375 0.5599861495844876 0.37291015625 0.1536149584487534 5
6 0.678369140625 0.768393351800554 0.4741796875 0.0915927977839335
1 0.24103515625 0.7428670360110804 0.246728515625 0.1078116343490304 7
2 0.238525390625 0.8764542936288088 0.22384765625 0.07819944598337951
```

class_id x_center y_center width height

🗂️ splitting Dataset using YOLOv8 format

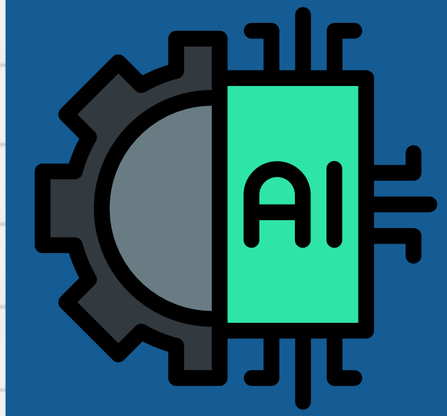
- DataSet Splits into:
  - 70% Training
  - 20% Validation
  - 10% Testing
- Annotated via Roboflow



After collecting and labeling the dataset in Roboflo
downloaded the YOLOv8-compatible format”

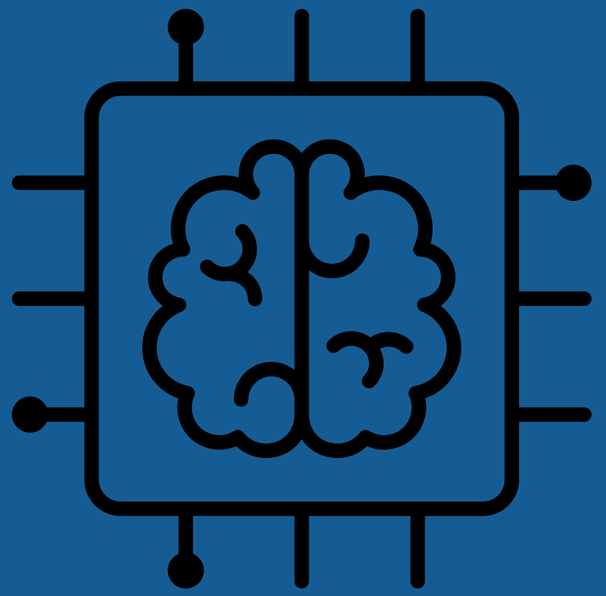
Data Set Strucure :

📂 train/
├── images/
└── labels/ ← Each .txt file contains bounding boxes for fields in the image
📂 valid/
📂 test/
📄 data.yaml ← Defines class names and paths



Let’s fine-tune our models

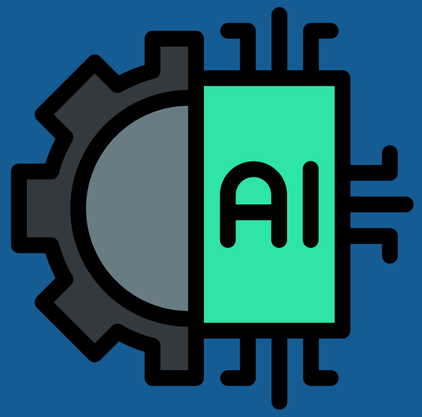
Model Selection



Why we Choose Yolov8 ?

We Used YOLOv8n.pt from Ultralytics

Because it’s Lightweight and fast — good for prototyping

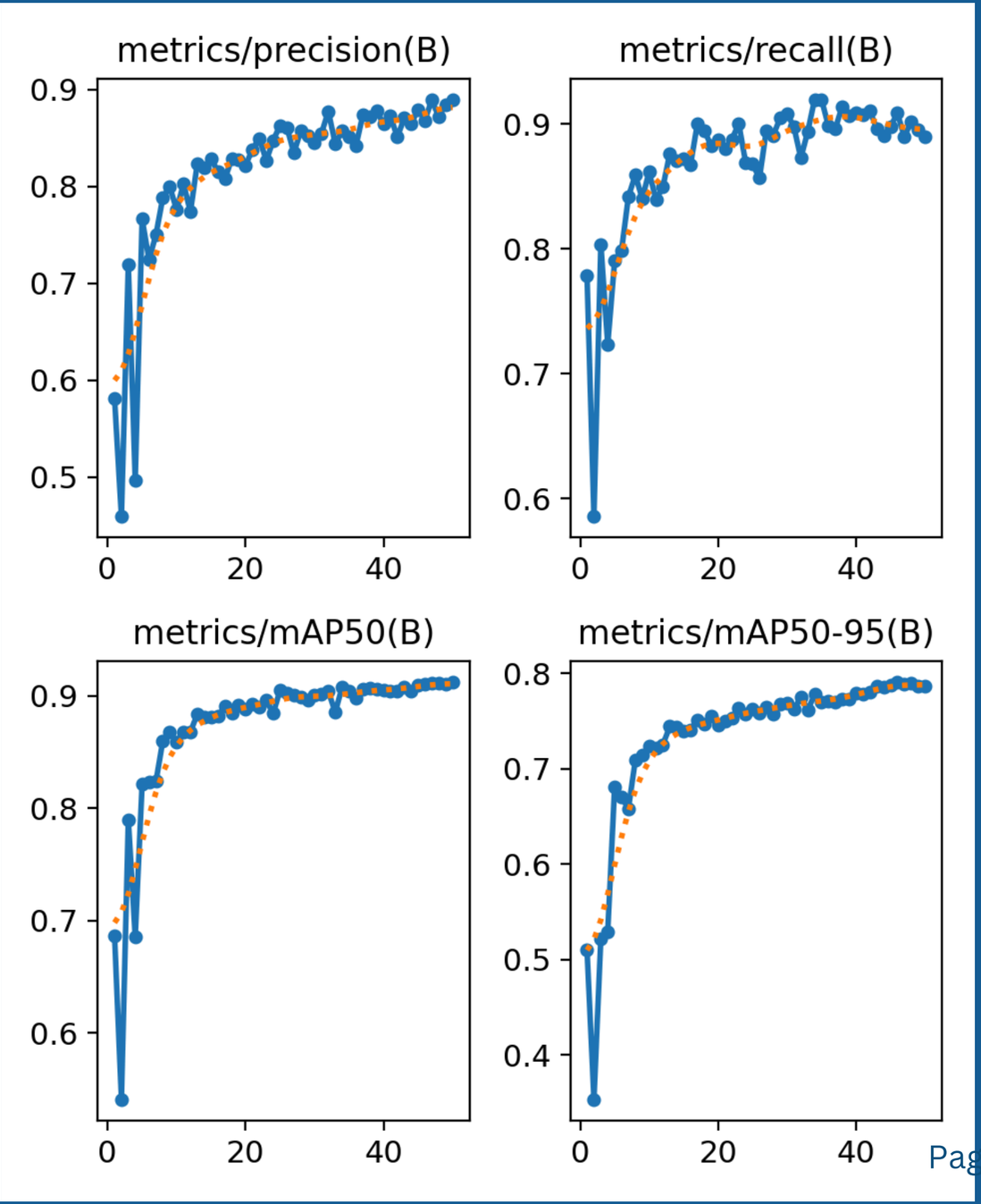


Model 1: Detect ID Card inside image

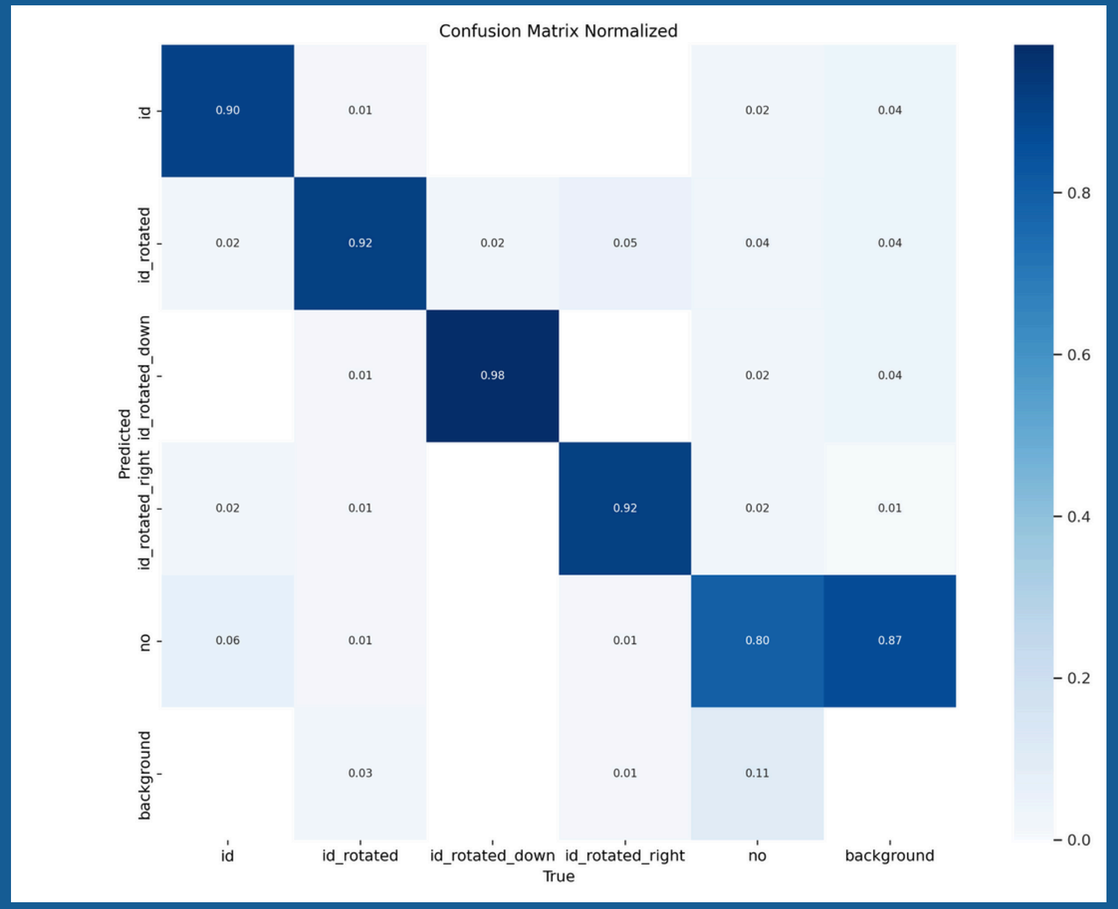
Model Performance

Started at 0.58 (epoch 1), reached 0.89 by epoch 50

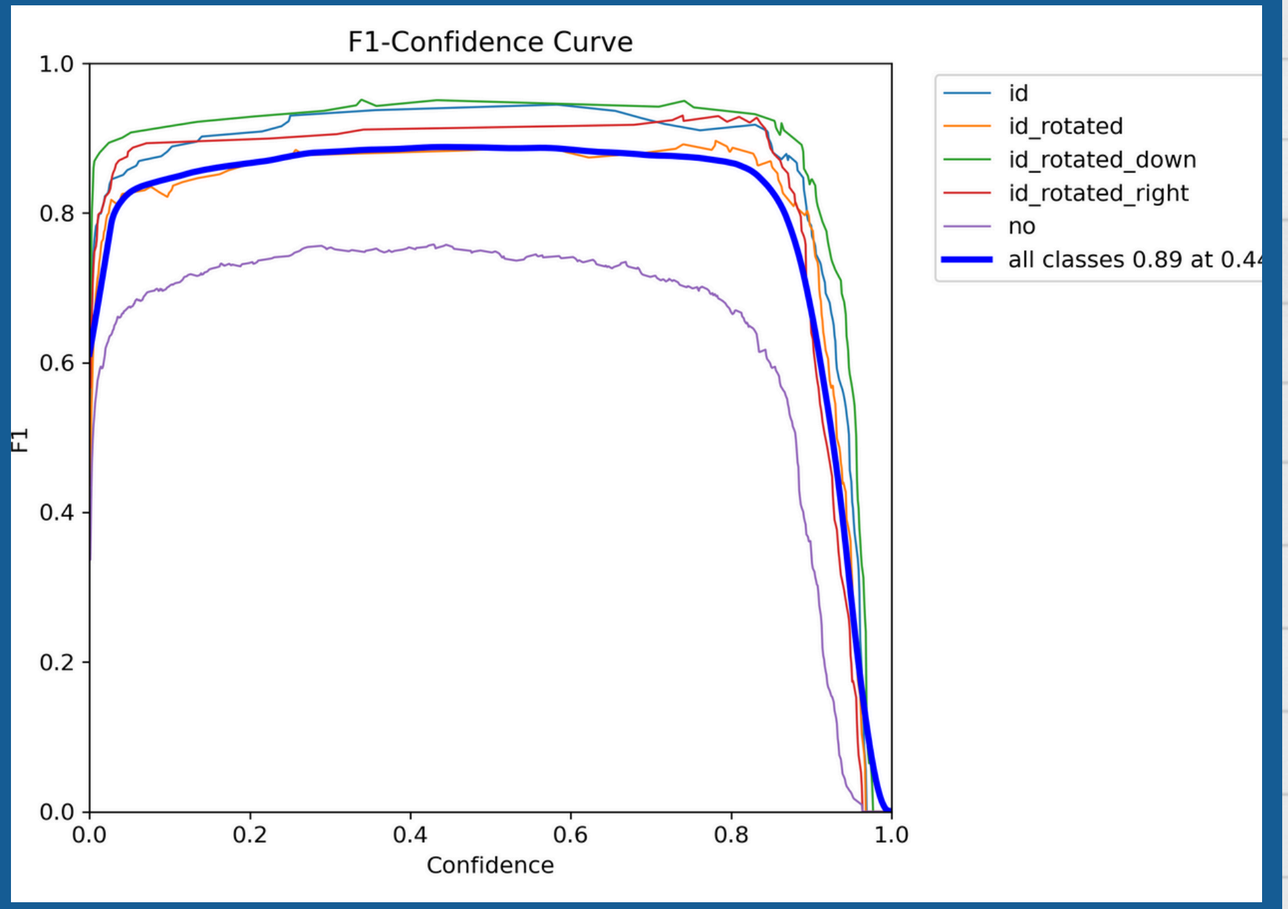
- Precision (B): 0.890
- Recall (B): 0.890
- mAP50 (B): 0.912 (IOU)
- mAP50–95 (B): 0.787



Confusion Matrix

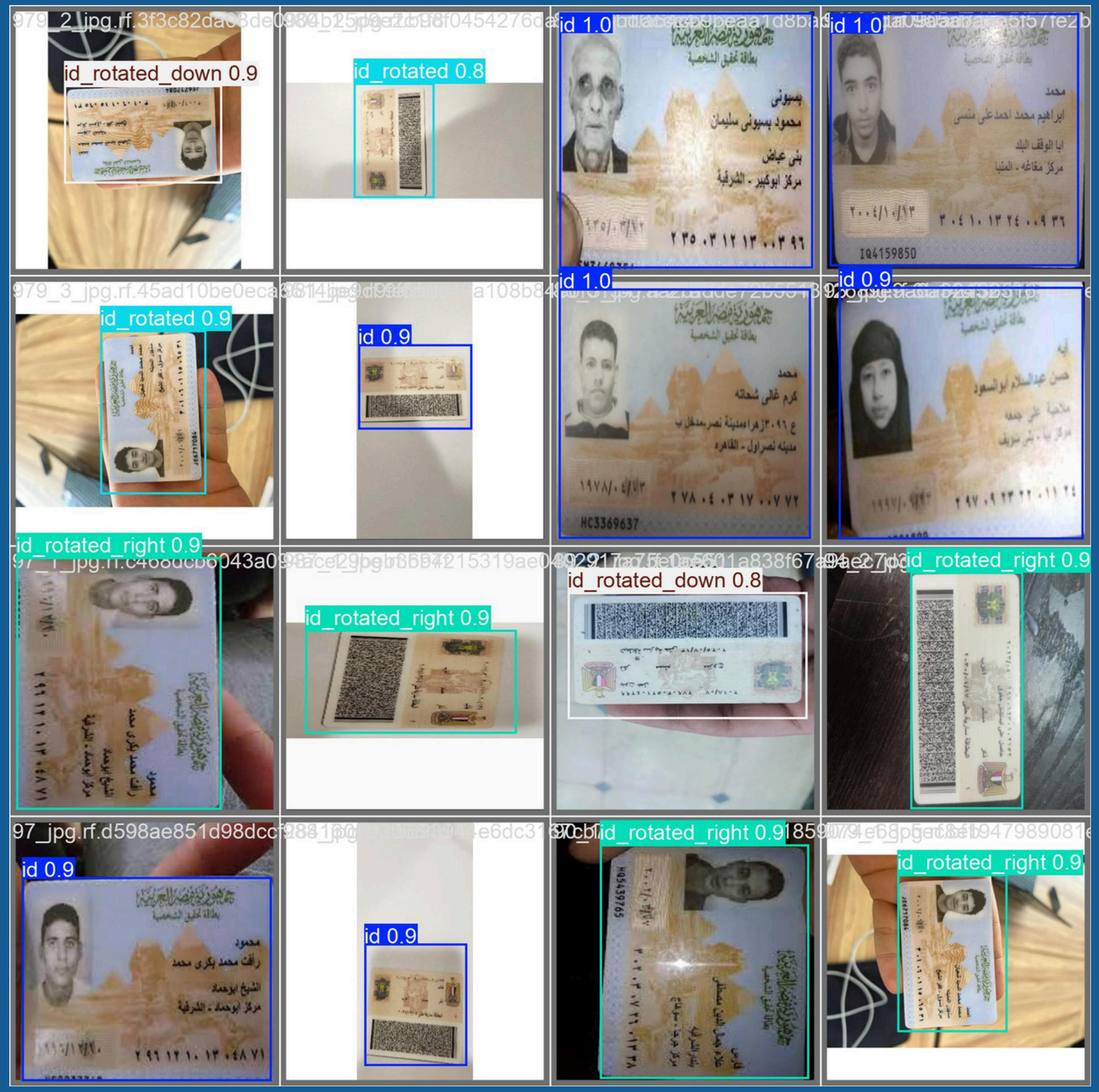


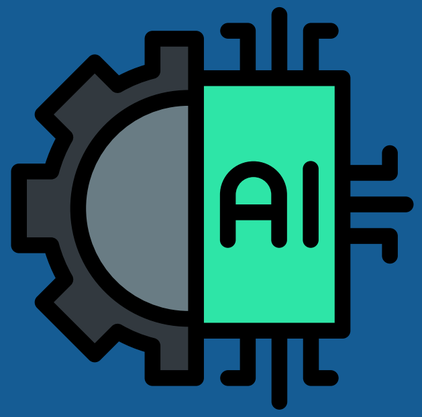
F1-Curve



Predictions

- id_rotated_down: 98%
- id_rotated: 92%
- id: 90%
- id_rotated_right: 92%



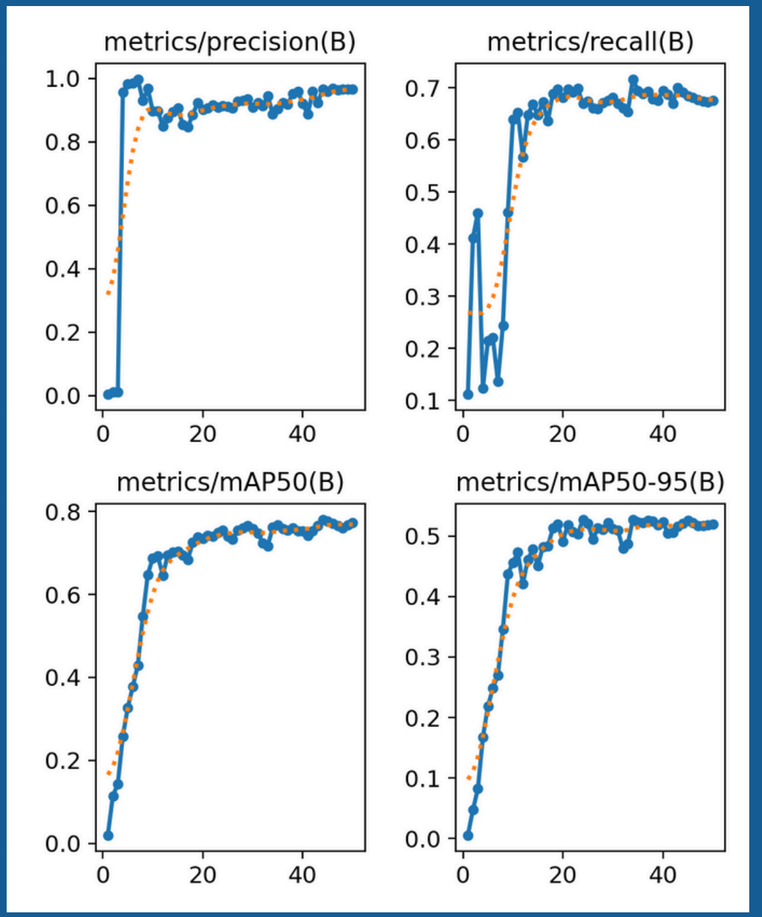


Model 2: Detect Feilds inside ID Card

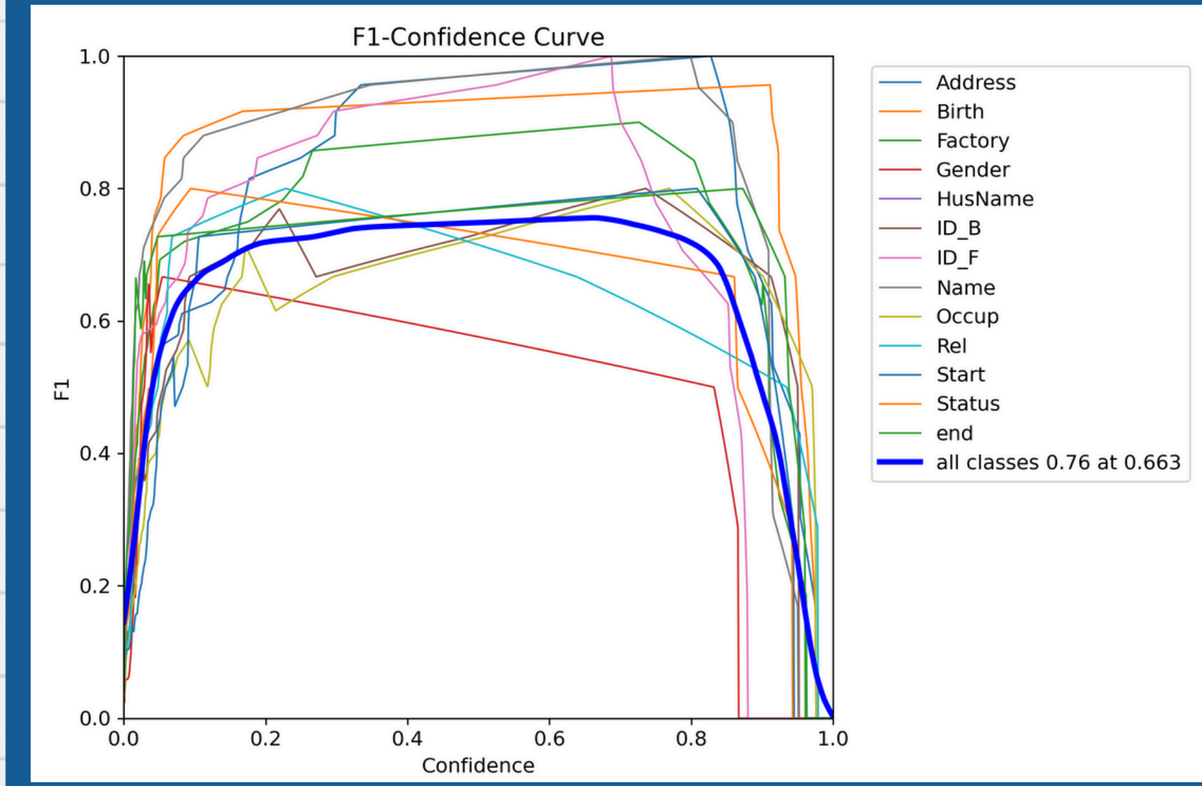
Performance

- Precision (B): 0.96719
- Recall (B): 0.7060
- mAP50 (B): 0.77345

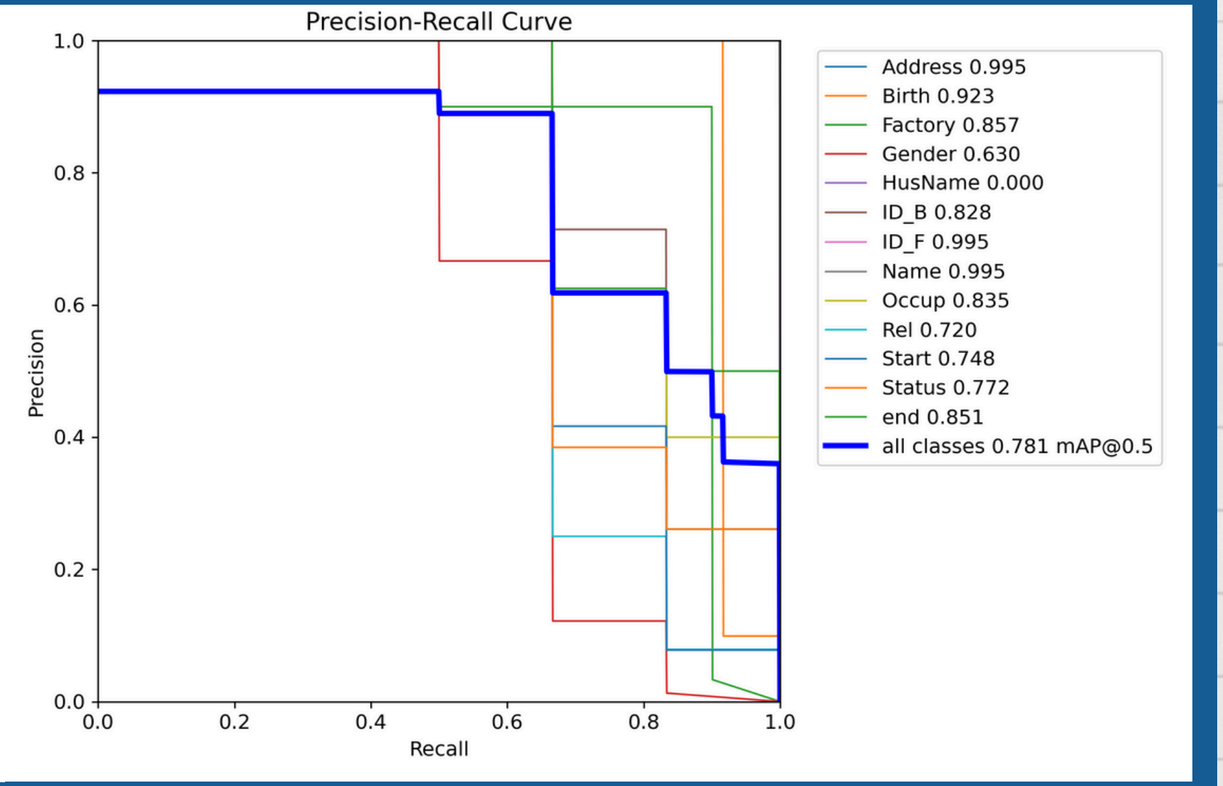
A	B	C	D	E	F	G	H
epoch	time	train/box_loss	train/cls_loss	train/dfl_loss	metrics/precision(B)	metrics/recall(B)	metrics/mAP50(B)
1	8.81423	2.80502	4.73856	2.25405	0.00368	0.11189	0.01985
2	10.8757	1.96899	4.37742	1.63641	0.01212	0.41224	0.11365
3	12.8917	1.58298	3.88634	1.33118	0.01065	0.46026	0.14372
43	92.5058	1.04318	0.97104	1.06246	0.92344	0.70088	0.76659
44	94.414	1.02939	0.94208	1.05802	0.96624	0.69217	0.78025
45	96.3594	1.01359	0.93279	1.0423	0.96031	0.68517	0.7764
46	98.4796	1.04938	1.04172	1.07853	0.96782	0.68162	0.77214
47	100.398	0.98898	0.91159	1.02056	0.96399	0.67684	0.7666
48	102.316	0.98395	0.92567	1.03883	0.96735	0.67385	0.76101
49	104.313	0.95591	0.89438	1.03052	0.96619	0.67341	0.76738
50	106.178	1.00275	0.90595	1.0468	0.96719	0.67603	0.77345



F1-Curve



PR Curve



Test Results

🎯 Top Performing Classes

Class	Accuracy
Address	100% ✅
ID_F	100% ✅
Birth	92% ✅
Factory	90% ✅

ID Extraction Phases

01

ID detection using YOLO

02

Field detection using YOLO

03

Content extraction using OCR

Phase 1

Phase 2

Phase 1 Summary

- **YOLO Model 1:** ID Detection

This model detects and localizes the National ID card within a larger image, especially useful if the ID is placed on a cluttered background or captured from a camera.

- **YOLO Model 2:** Field Detection

After cropping the ID, we feed it to a second YOLO model trained to detect individual fields on the card: national ID number, name, date of birth ,gender ..etc.

Phase 1 Final Results

This two-step detection helps us isolate only the relevant parts of the ID for further processing

National ID front sample

National ID back sample

READY FOR PHASE 2

🧾 Why OCR is Needed

To **automatically read the text content** from the Egyptian National ID provided during voter registration,
so we can **extract and verify** key information such as:

- 🆔 National ID Number
- 👨‍💼 Full Name
- 📅 Date of Birth

This ensures the user-provided data matches the official ID, enabling a secure and automated identity verification process.

OCR and Field Extraction

After detecting the National ID and its individual fields using our two-stage YOLO pipeline, we crop each detected field and pass it to our OCR module for text recognition.

We used **EasyOCR**, a deep learning–based text recognition model that supports both Arabic and English scripts. This was essential for accurately extracting the ID content, especially the Arabic numbers, and converting them into English digits to match the format that will be provided by the user during the voting process.

Why EasyOCR?

✅ Multilingual Support

- Supports both Arabic and English, which is essential for Egyptian National IDs.

✅ Deep Learning-Based

- Uses CRNN + CTC decoding, providing high accuracy for real-world images with mixed fonts and noisy backgrounds.

✅ No Training Required

- Pretrained and ready to use — saved time and effort compared to building a custom OCR from scratch.

Why EasyOCR?

✅ Field-Level Flexibility

- Performs well on small cropped fields, which matches our YOLO-based pipeline.

✅ Lightweight and Easy to Integrate

- Simple Python API that fits easily into our preprocessing and post-processing pipeline.

OCR Workflow

01

ID detection using YOLO

02

Field detection using YOLO

03

Content extraction using OCR

01

Preprocessing

02

Apply EasyOCR model

03

Post-Processing

04

Final Output

Preprocessing for OCR Accuracy

Before feeding the cropped fields into the OCR model, we applied several preprocessing steps to improve accuracy:

- **Grayscale conversion** to simplify the image and focus on text.
- **Inverse thresholding** to enhance contrast between text and background.
- **Resizing** the image to make small text more readable for OCR.
- **Noise reduction** by simplifying the image structure.

These steps significantly improve OCR performance and ensure the extracted data is clean and usable.

Preprocessing for OCR Accuracy

```
14%|          | 1/7 [00:04<00:24,  4.15s/it]

Image: ID_B_0.jpg
Original Text: ٢٩ ٤ ٠ ١ ١ ٥ ١ ٦ ٠ ٤٣٥١
Extracted Numbers: 29401151604351
```

```
29%|          | 2/7 [00:06<00:14,  2.91s/it]

Image: Start_3.jpg
Original Text: ٢ ٠ ٢ ٢١٠ ٩
Extracted Numbers: 2022/09
```

```
57%|          | 4/7 [00:11<00:08,  2.71s/it]

Image: Occup_5.jpg
Original Text: بكالوريوس هندسة
```

Post-Processing for OCR Accuracy

We applied custom post-processing techniques to improve text extraction quality:

- Converted Arabic digits to English
- Extracted clean numeric values (e.g., NID, dates)
- Removed unwanted symbols and spaces
- Fixed OCR errors like slashes misread as "1"

This approach helps us accurately extract numeric fields for use in our e-voting system.

FINAL RESULTS

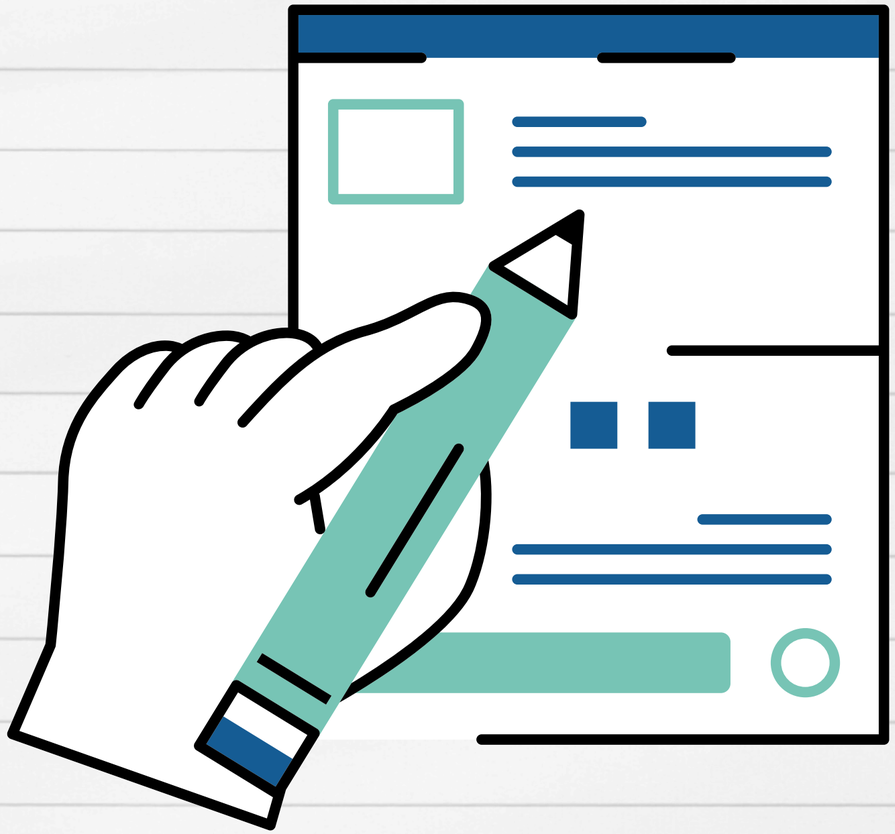
OCR Final Results

This table is generated automatically after running the full pipeline on a batch of ID images.
Each field is detected, cropped, preprocessed, and passed through OCR.
The result is then **cleaned and normalized** to ensure it’s accurate and consistent then **saved as CSV file.**

	image_id	original_text	extracted_numbers
0	ID_B_0	٢٩٤٠١١٥١٦٠٤٣٥١	29401151604351
1	Start_3	٢٠٢٢١٠٩	2022/09
2	Status_4	اعزب	
3	Occup_5	بكالوريوس هندسة	
4	Rel_2	مسلم	3
5	Gender_6	ذكر	26
6	end_1	٢٠٢٩١٠٩٢١	2029/09/21

07

FAKE ID DETECTION



ID CARD VERIFICATION MODEL

- **Purpose:**
  - Detects fake IDs by analyzing image inconsistencies using AI + forensics techniques (ELA & LBP).
- **Why It Matters:**
  - Used for identity theft & fraud detention .
  - Catches what humans miss under time pressure.
  - Learns subtle tampering patterns automatically.

PREPROCESSING

we use two forensic techniques:

1. **Error Level Analysis (ELA):** highlights areas of an image that may have been edited by showing differences in compression quality.
    - Re-compresses images at 90% quality.
    - Compares with the original to highlight edited regions.
    - Amplifies differences by 15× so Tampered areas appear brighter.

ELA acts like a Photoshop detector.

PREPROCESSING

2. Local Binary Patterns (LBP): detects texture patterns in an image to find inconsistencies that might indicate tampering.

- Analyzes neighborhood pixel relationships.
- Encodes texture patterns as numerical features.
- Detects Variations in fonts/backgrounds.

LBP acts like fingerprint scanner for image textures.

PREPROCESSING

Feature	ELA	LBP
Detects	Digital tampering	Physical tampering
Best for	Photoshopped regions	Printed/Scan quality issues
strengths	Catches subtle digital edits	Works on physical copies
Outputs	Heatmap of edits	Texture pattern map

MODEL ARCHITECTURE

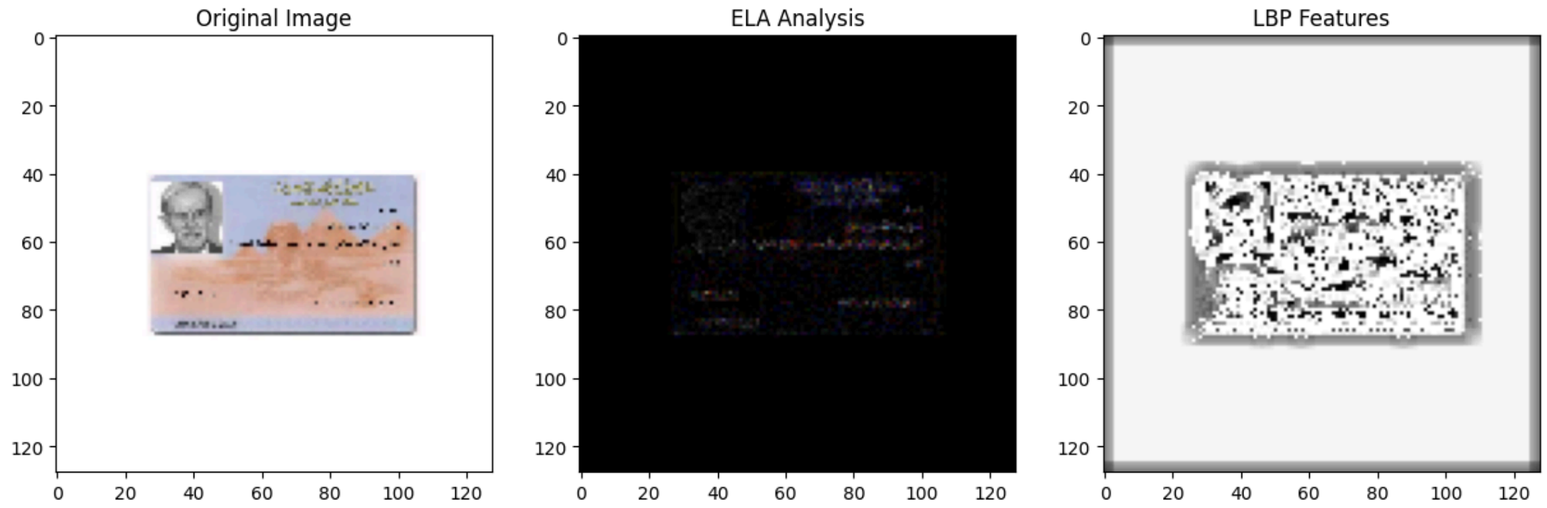
Three-Branch Hybrid Network:

- **Original Image Branch:** EfficientNetB0,cause it recognizes fine details.
- **ELA Branch:** Processing layer, it finds editing traces.
- **LBP Branch:** Processing layer,it catches physical tampering.
- **Feature Fusion from all three branches.**

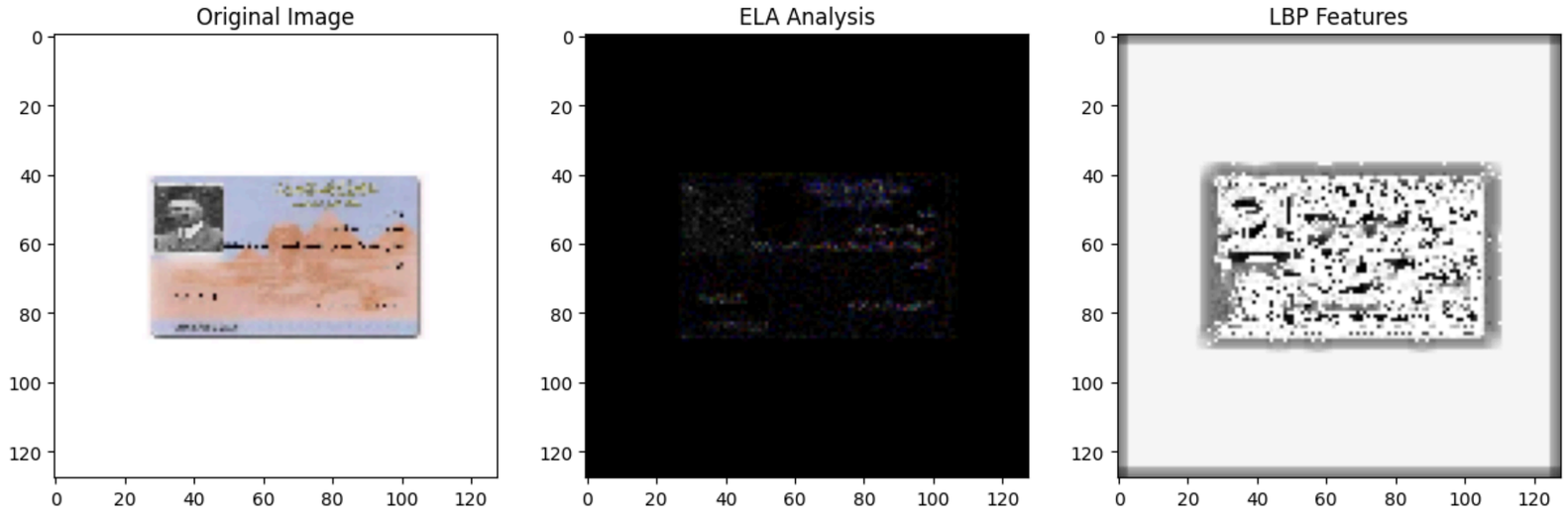
```
Train or predict? (t/p):  t
Epoch 1/5
3830/3830 ━━━━━━━━━━━━━━━━━━━━ 4359s 1s/step - accuracy: 0.9571 - loss: 0.1423 - val_accuracy: 0.9967 - val_loss: 0.0117
Epoch 2/5
3830/3830 ━━━━━━━━━━━━━━━━━━━━ 4278s 1s/step - accuracy: 0.9981 - loss: 0.0096 - val_accuracy: 0.9992 - val_loss: 0.0030
Epoch 3/5
3830/3830 ━━━━━━━━━━━━━━━━━━━━ 4256s 1s/step - accuracy: 0.9993 - loss: 0.0032 - val_accuracy: 0.9999 - val_loss: 0.0011
Epoch 4/5
3830/3830 ━━━━━━━━━━━━━━━━━━━━ 4247s 1s/step - accuracy: 0.9997 - loss: 0.0016 - val_accuracy: 0.9997 - val_loss: 0.0011
Epoch 5/5
3807/3830 ━━━━━━━━━━━━━━━━━━━━ 20s 905ms/step - accuracy: 0.9997 - loss: 0.0015
```

- The model gets accuracy of 99.9% on train and 99.9%on validation

```
Testing image: 0.jpg
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
Prediction: FAKE (100.0% confidence)
```

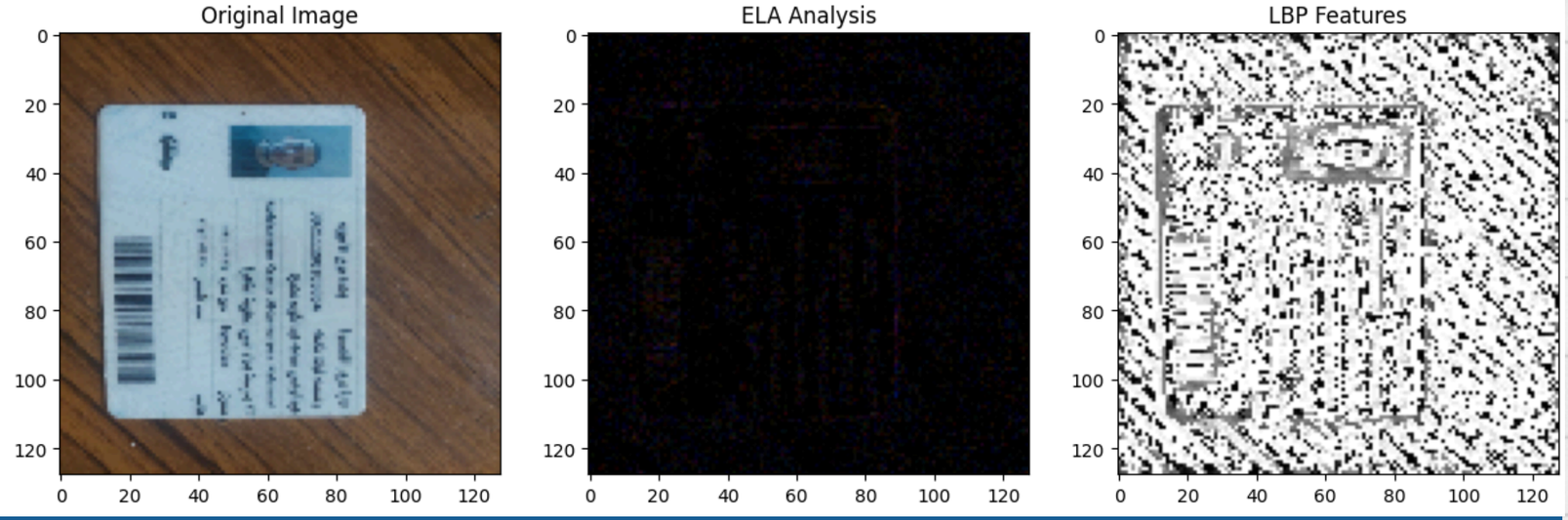



```
Testing image: 1.jpg
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
Prediction: FAKE (100.0% confidence)
```

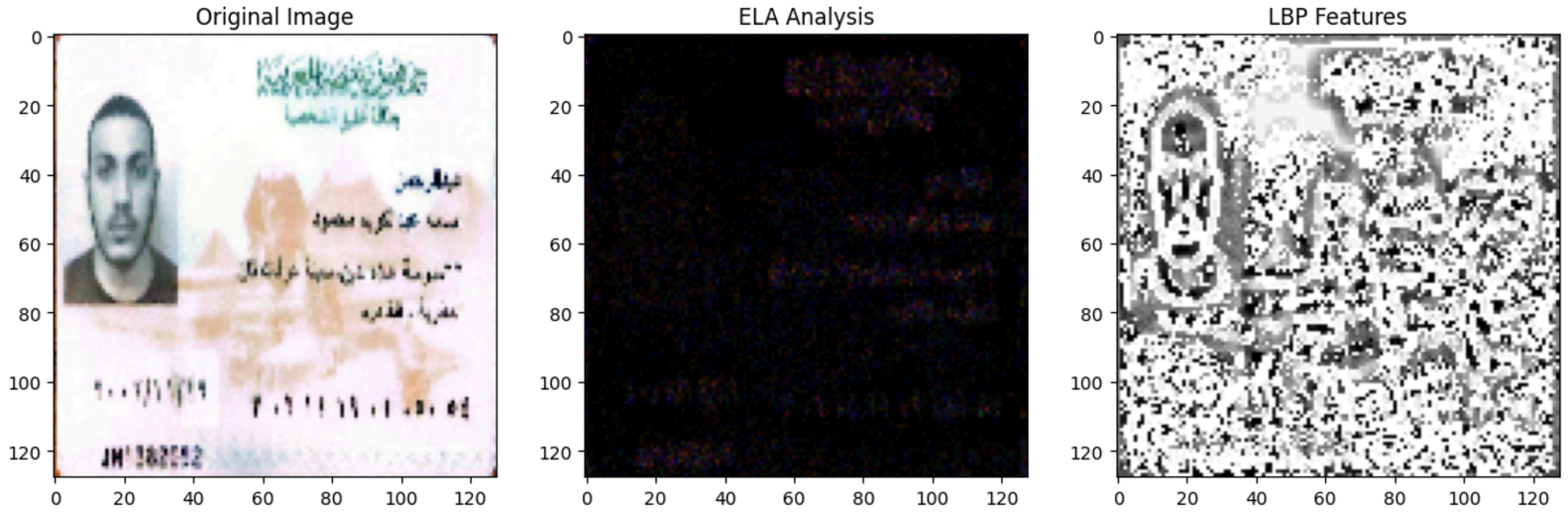


- And here are samples of the test on fake id cards

```
Testing image: WhatsApp Image 2025-04-12 at 22.59.56_cfb2df2b.jpg
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
Prediction: REAL (100.0% confidence)
```



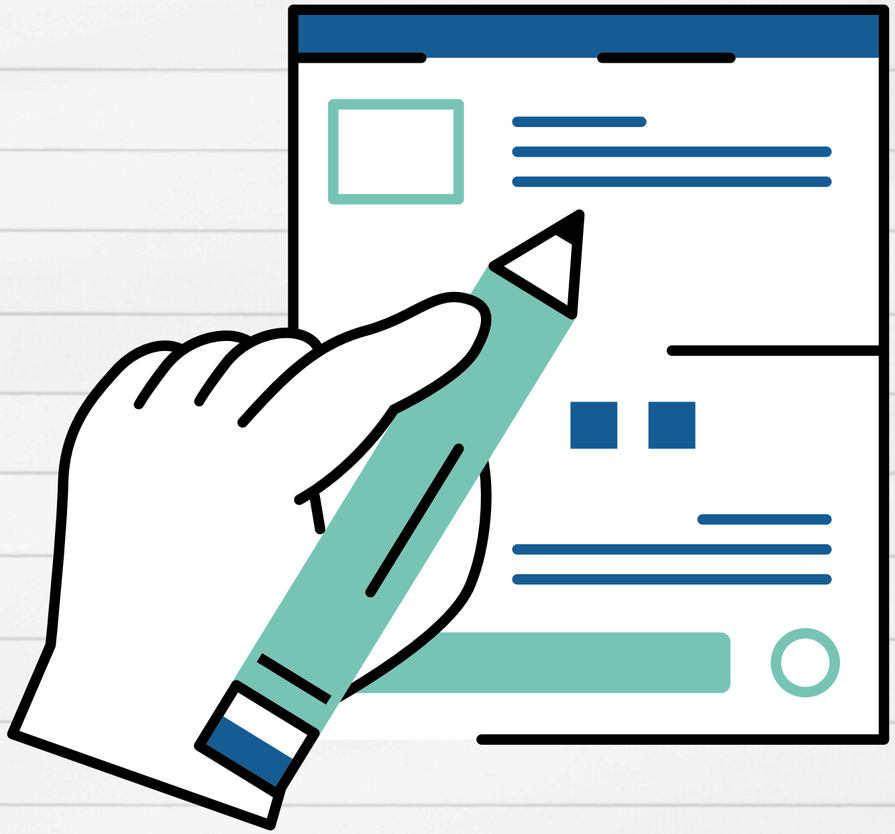

```
Testing image: WhatsApp Image 2025-03-25 at 00.47.22_daab1142.jpg
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
Prediction: REAL (78.7% confidence)
```



- And here are samples of real id cards

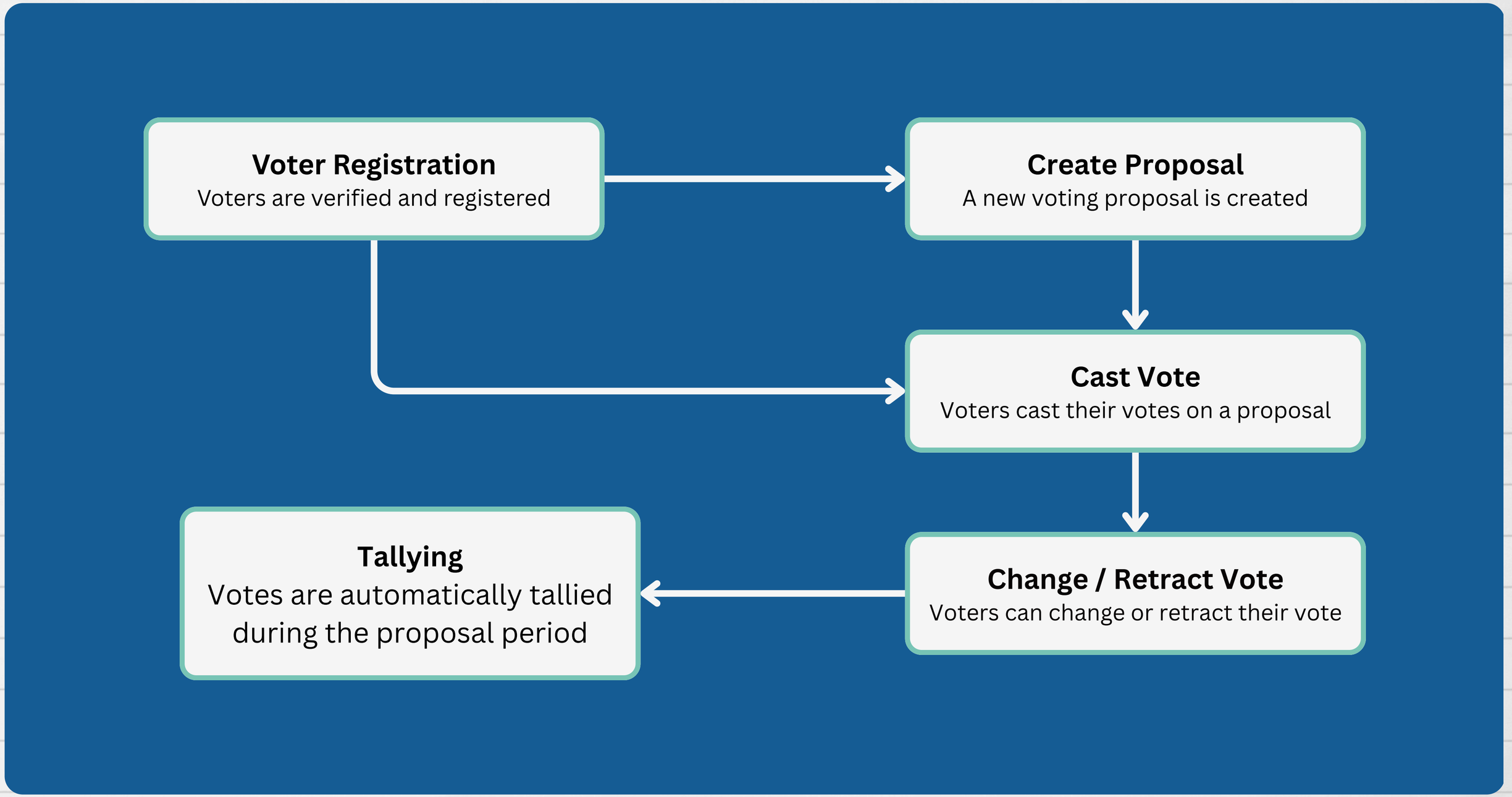
09

BLOCKCHAIN

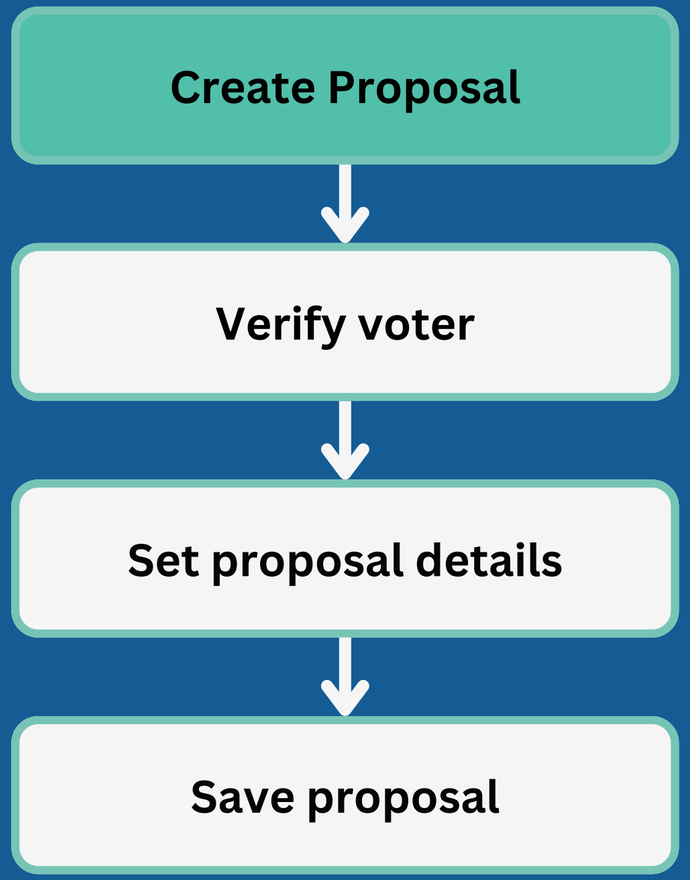


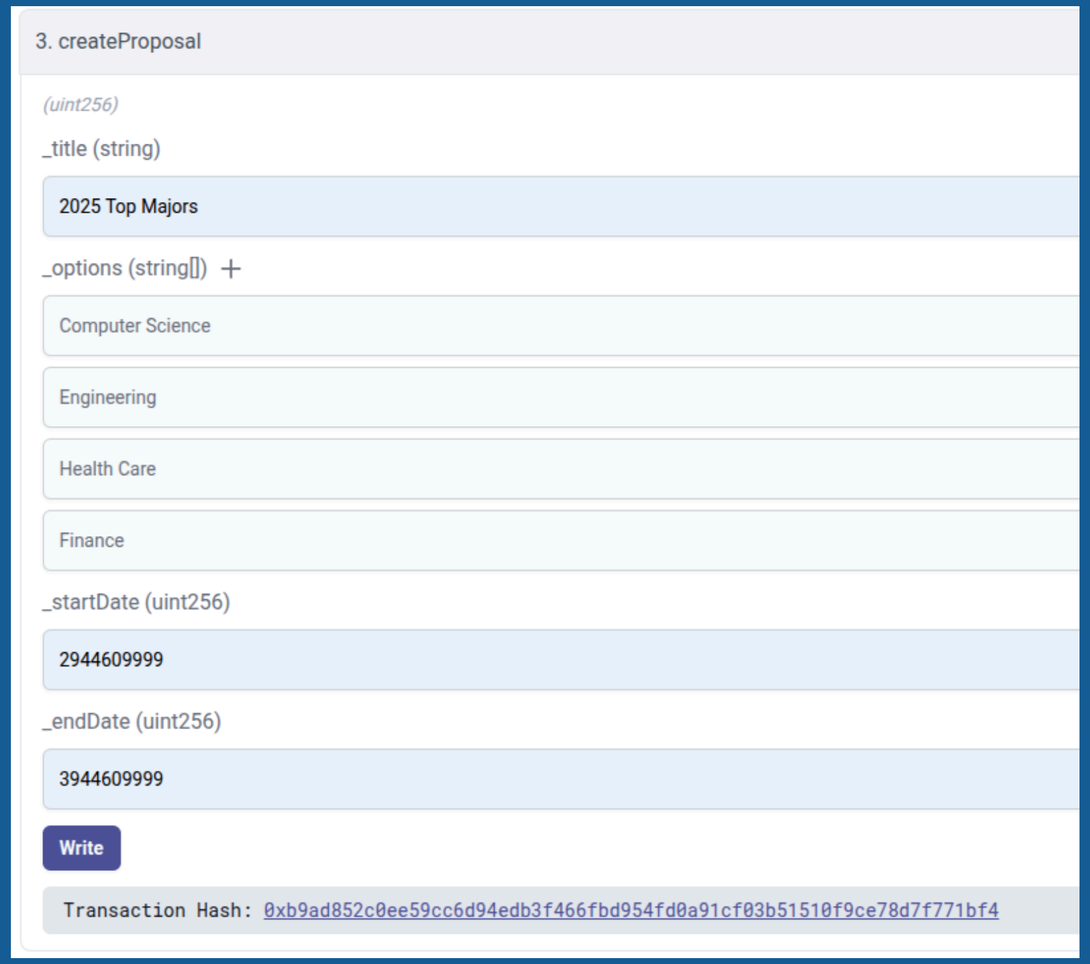
Implementation

- User Registration & Verification
- Creating Proposals
- Casting Votes
- Retracting Votes
- Changing Votes

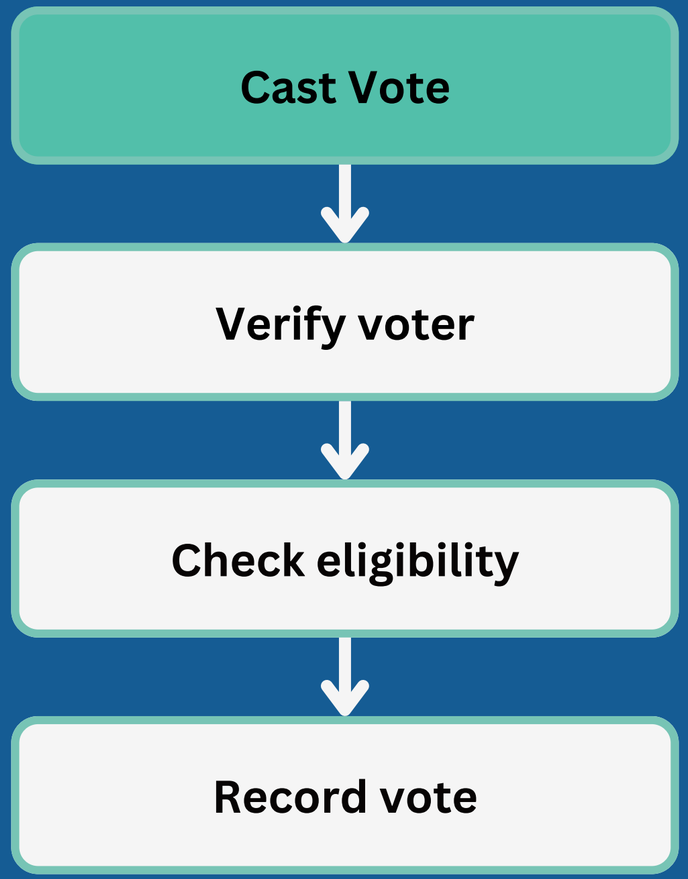

Voter Registration
Voters are verified and registered
Create Proposal
A new voting proposal is created
Cast Vote
Voters cast their votes on a proposal
Change / Retract Vote
Voters can change or retract their vote
Tallying
Votes are automatically tallied during the proposal period

Creating Proposals





Casting Votes



1. castVote

proposalId (uint256)

1

option (string)

Computer Science

Write

Transaction Hash: 0x1f7e545a46dc0aa2dfde92f5dca288dbb1

Only Once

1. castVote

proposalId (uint256)

1

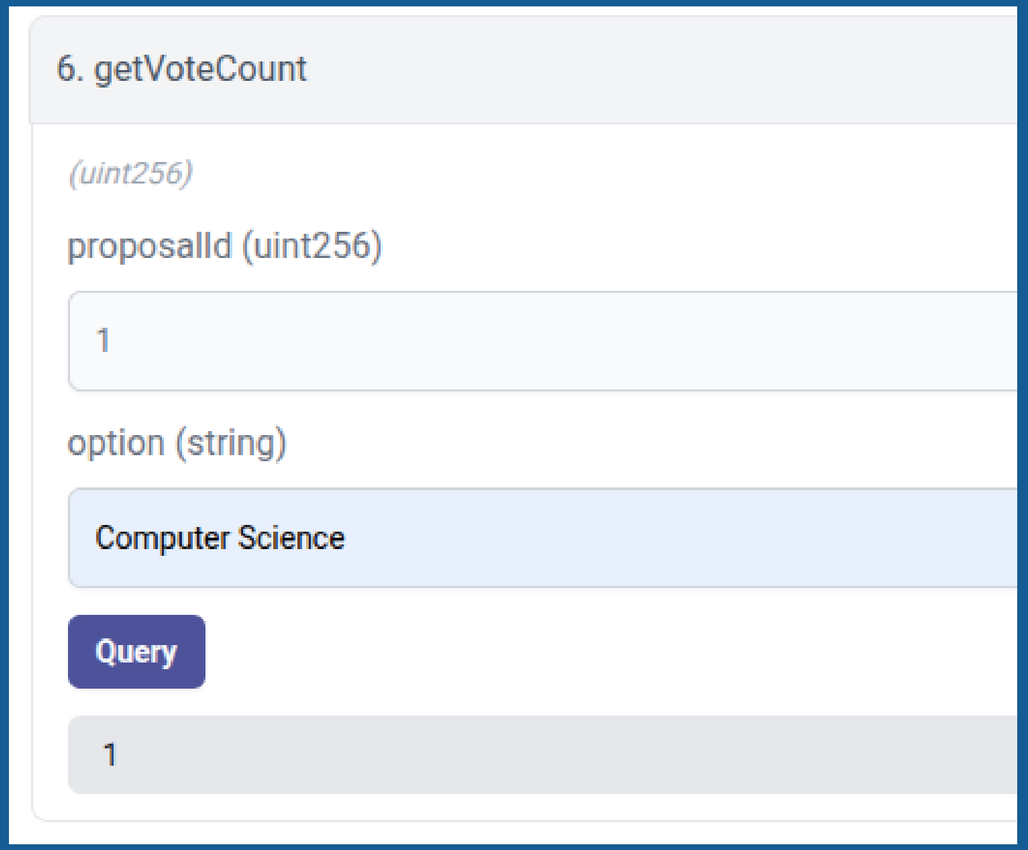
option (string)

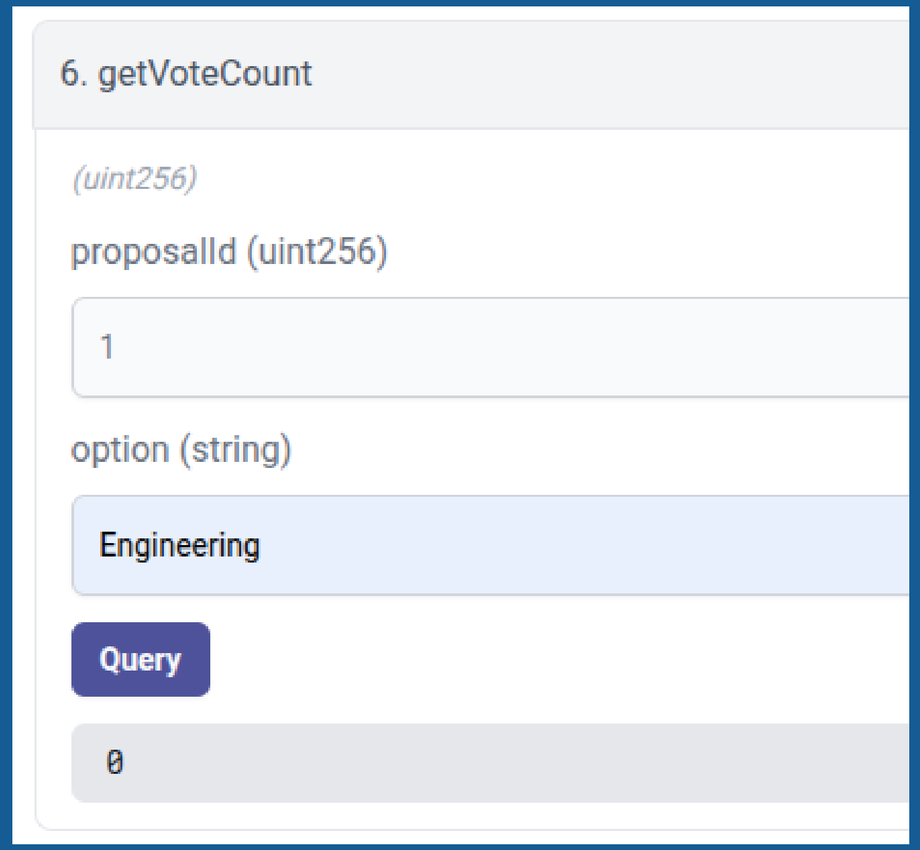
Engineering

Write

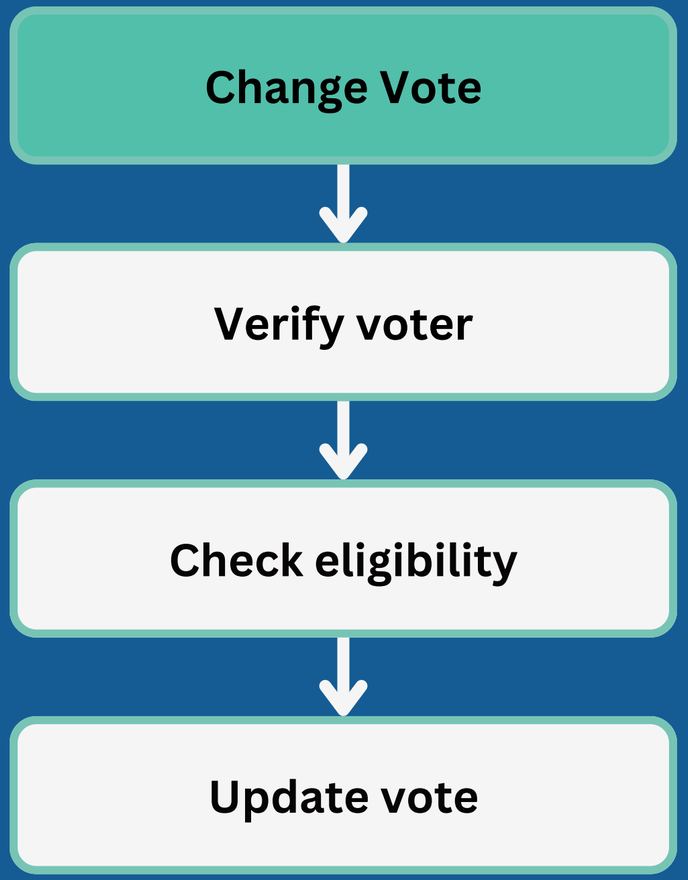
⚠ execution reverted (payload={ "id": 8, "jsonrpc": "2.0", "method": "zks_estimateFee", "params": [{ "data": "0xd037853a00000000000000000000000000000000000000000000000000
00000000000001000000000000000000000000000000000000000000000000000000000000004000000000000000000000000000000000000000000000000000000000000000
0b456e67696e656572696e67000000000000000000000000000000000000000000", "from": "0x45586259e1816ac7784ae83e704ed354689081b1", "to": "0x6b080f6e05b2e0b844
526db78a192a660c2835e7", "type": "0x0" }] }, error={ "code": 3, "data": "0xa64afec0000000000000000000000000000000000000000000000000000000000000000100000000000
000000000000045586259e1816ac7784ae83e704ed354689081b1", "message": "execution reverted" }, code=3, version=6.13.4)

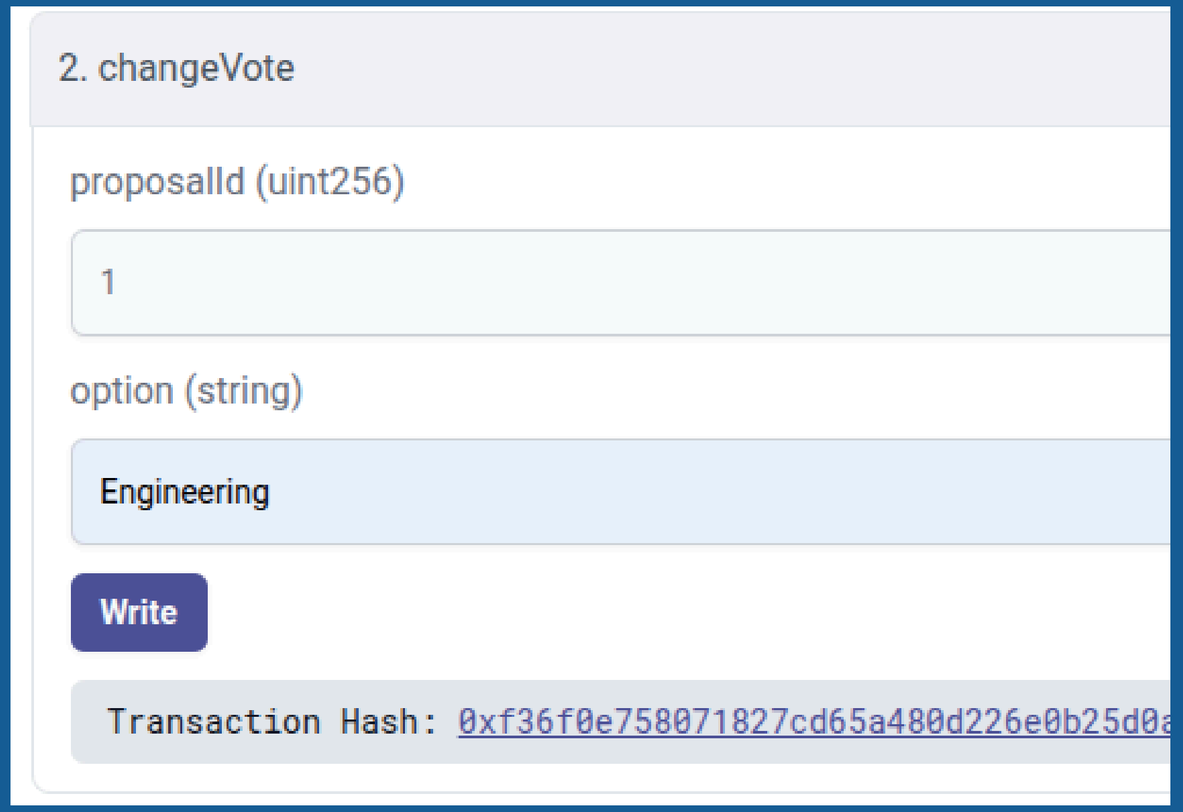
Viewing Vote Counts

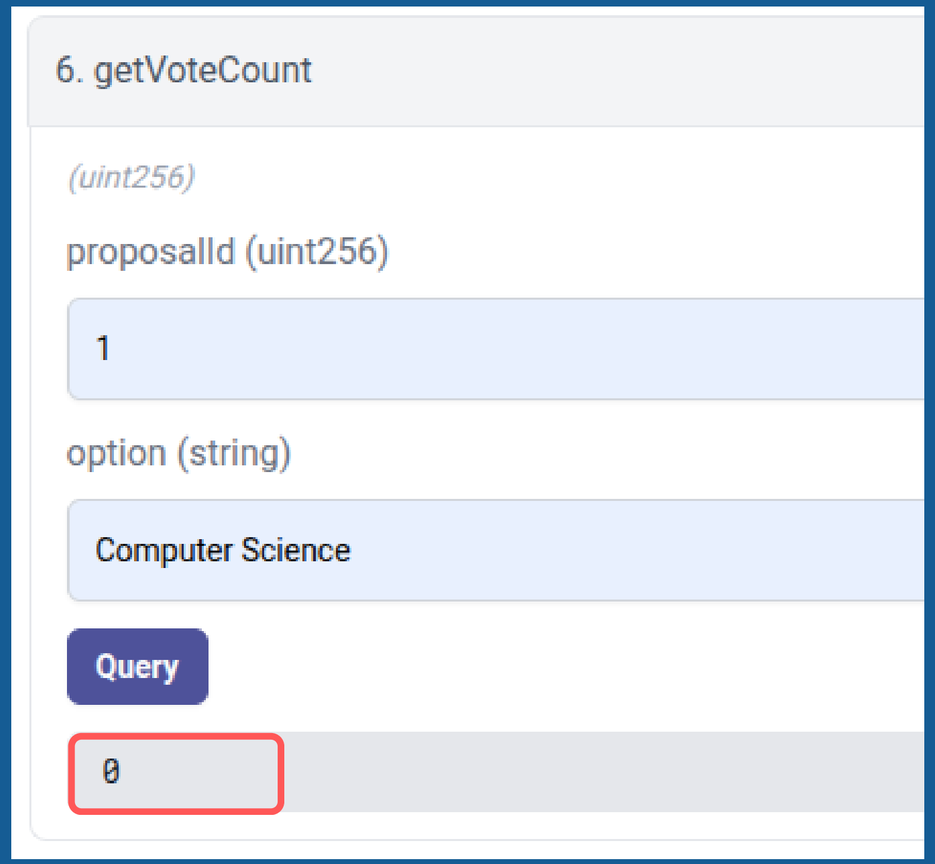


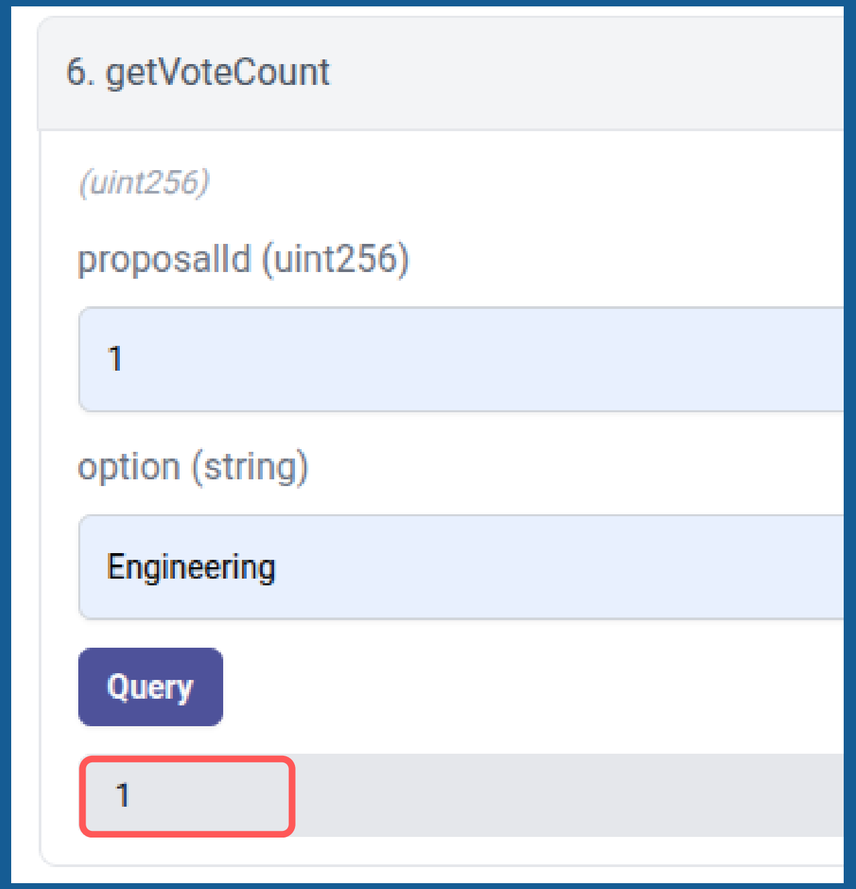


Changing Votes

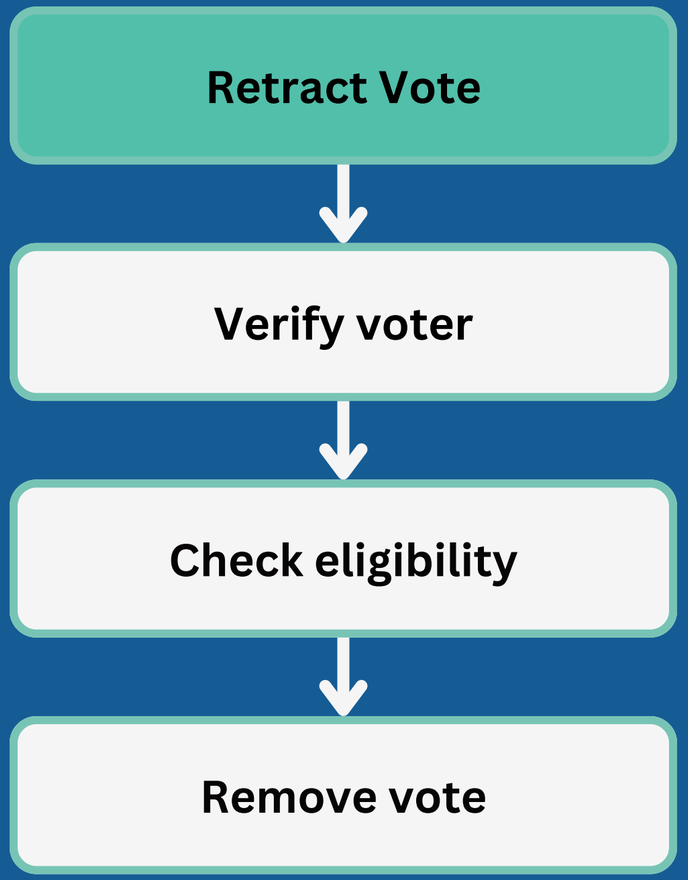


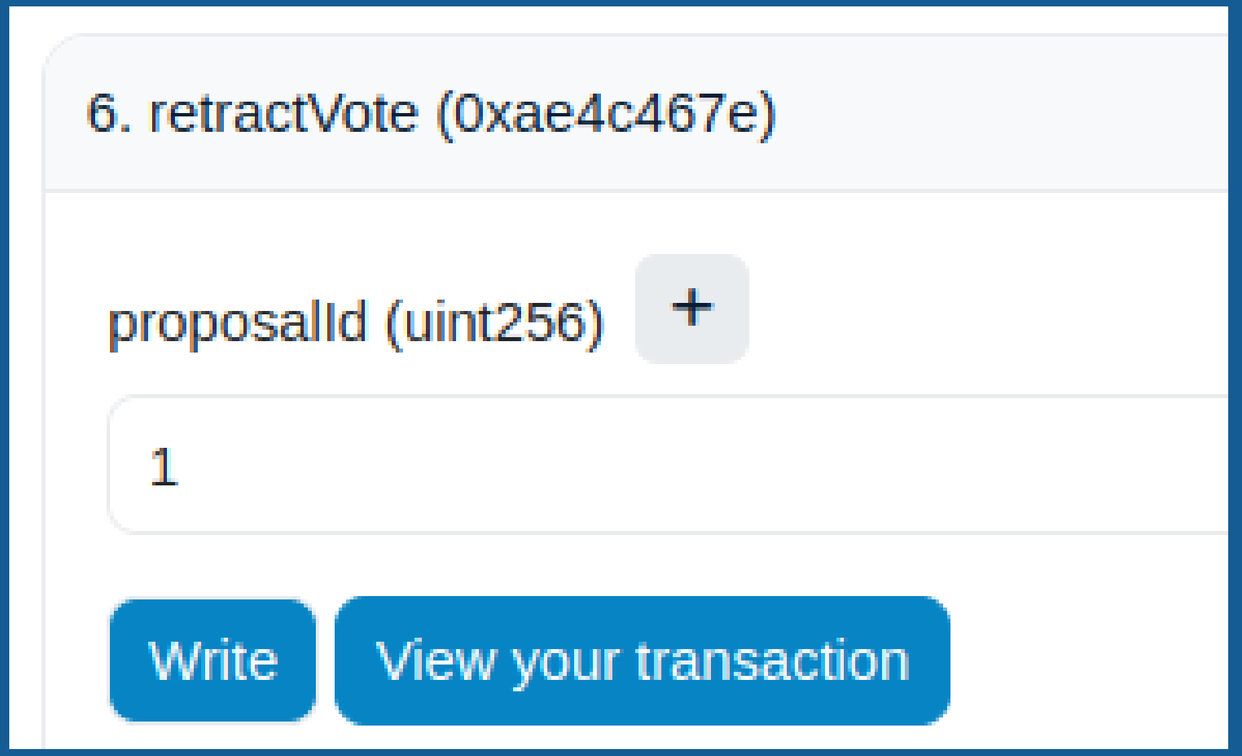



6. getVoteCount
(uint256)
proposalId (uint256)
1
option (string)
Computer Science
Query
0


6. getVoteCount
(uint256)
proposalId (uint256)
1
option (string)
Engineering
Query
1

Retracting Votes





6. getVoteCount

(uint256)

proposalId (uint256)

1

option (string)

Computer Science

Query

0

6. getVoteCount

(uint256)

proposalId (uint256)

1

option (string)

Engineering

Query

0

Deployment

zkSync

create	10 hours ago	L2 0x45586...81b1	OUT	L2 Contract created	0 ETH $0	0.003...75 ETH $6.19

Ethereum L1

0x45586259...4689081b1	OUT	Create: Vote	0 ETH	0.00712724

zkSync

Transactions | Transfers | Contract | Events

STATUS	TRANSACTION HASH	METHOD	AGE	FROM		TO	VALUE	FEE	
Processed on	0x337150a...c053	retractVote	3 hours ago	L2 0x45586...81b1	IN	L2 0xf4799...1874	0 ETH $0	0.000002291575 ETH $0.004	→ $ 0.003
Processed on	0xf36f0e7...05df	changeVote	3 hours ago	L2 0x45586...81b1	IN	L2 0xf4799...1874	0 ETH $0	0.0000115482 ETH $0.02	→ $ 0.019
Processed on	0x630c476...4fed	castVote	3 hours ago	L2 0x45586...81b1	IN	L2 0xf4799...1874	0 ETH $0	0.000028760725 ETH $0.05	→ $ 0.048
Processed on	0xcb9ece9...bb1a	createProposal	3 hours ago	L2 0x45586...81b1	IN	L2 0xf4799...1874	0 ETH $0	0.000105677175 ETH $0.17	→ $ 0.177

Ethereum L1

Transactions | Internal Transactions | Token Transfers (ERC-20) | Contract | Events

Latest 4 from a total of 4 transactions — Download Page Data

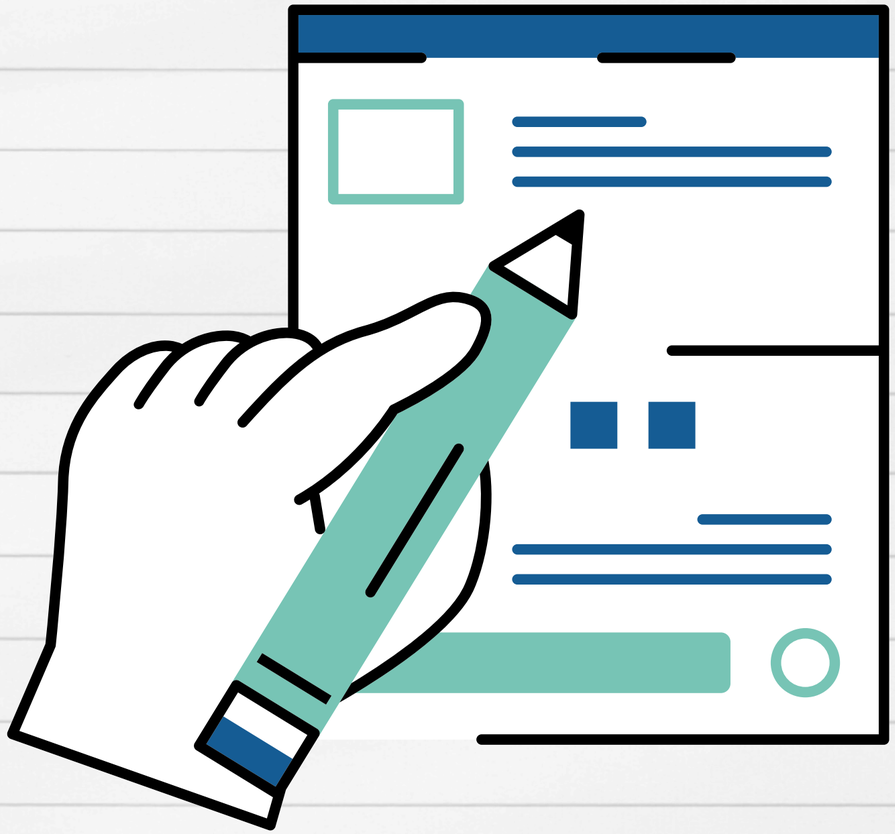
Transaction Hash	Method	Block	Age	From		To	Txn Fee	
0x8b0bf8cbb9e...	Retract Vote	8116668	34 secs ago	0x45586259...4689081b1	IN	0xC41e0B13...503a11318	0.0001273	→ $ 0.213
0x4267dd5810...	Change Vote	8116661	1 min ago	0x45586259...4689081b1	IN	0xC41e0B13...503a11318	0.00035363	→ $ 0.592
0xbba59a9035f...	Cast Vote	8116654	3 mins ago	0x45586259...4689081b1	IN	0xC41e0B13...503a11318	0.0004209	→ $ 0.705
0x704605f440a...	Create Propos...	8116649	4 mins ago	0x45586259...4689081b1	IN	0xC41e0B13...503a11318	0.00122821	→ $ 2.057

Results Comparison:

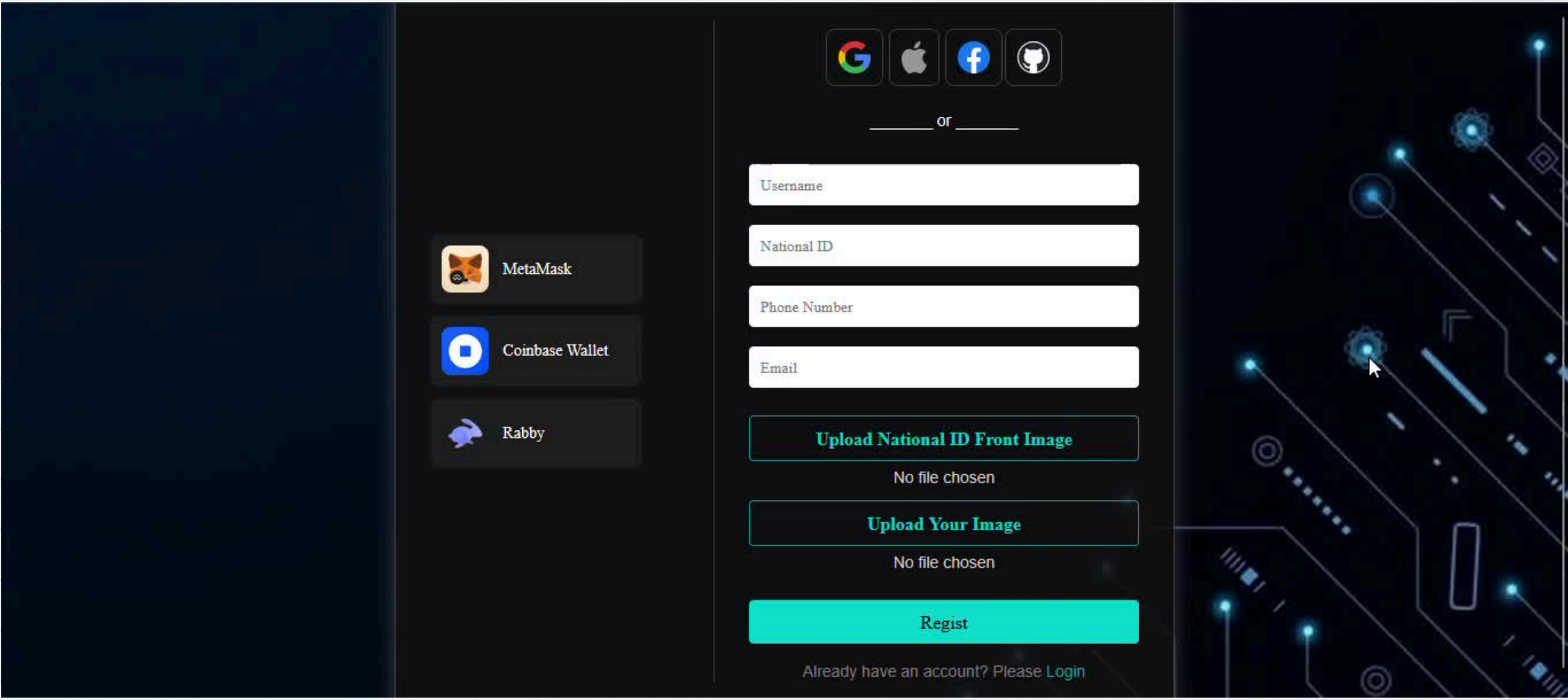
Metric	zkSync	Ethereum L1
Deployment Time	~2 seconds	~20 seconds
Tx. Confirmation Time	~0-1 minutes	~3-9 minutes
Deployment Cost (USD)	$6.19	$11.759
Avg Transaction Cost (USD)	$0.061	$0.891
Transaction Throughput (Transactions per second)	~2,000 TPS	15-30 TPS

10

FRONTEND

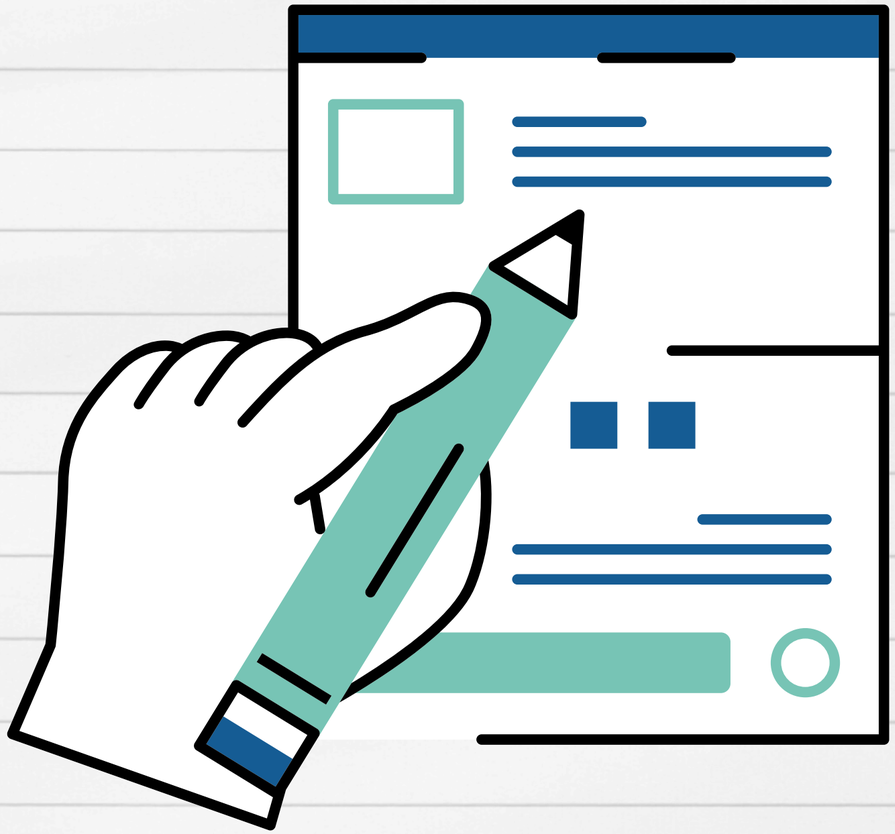


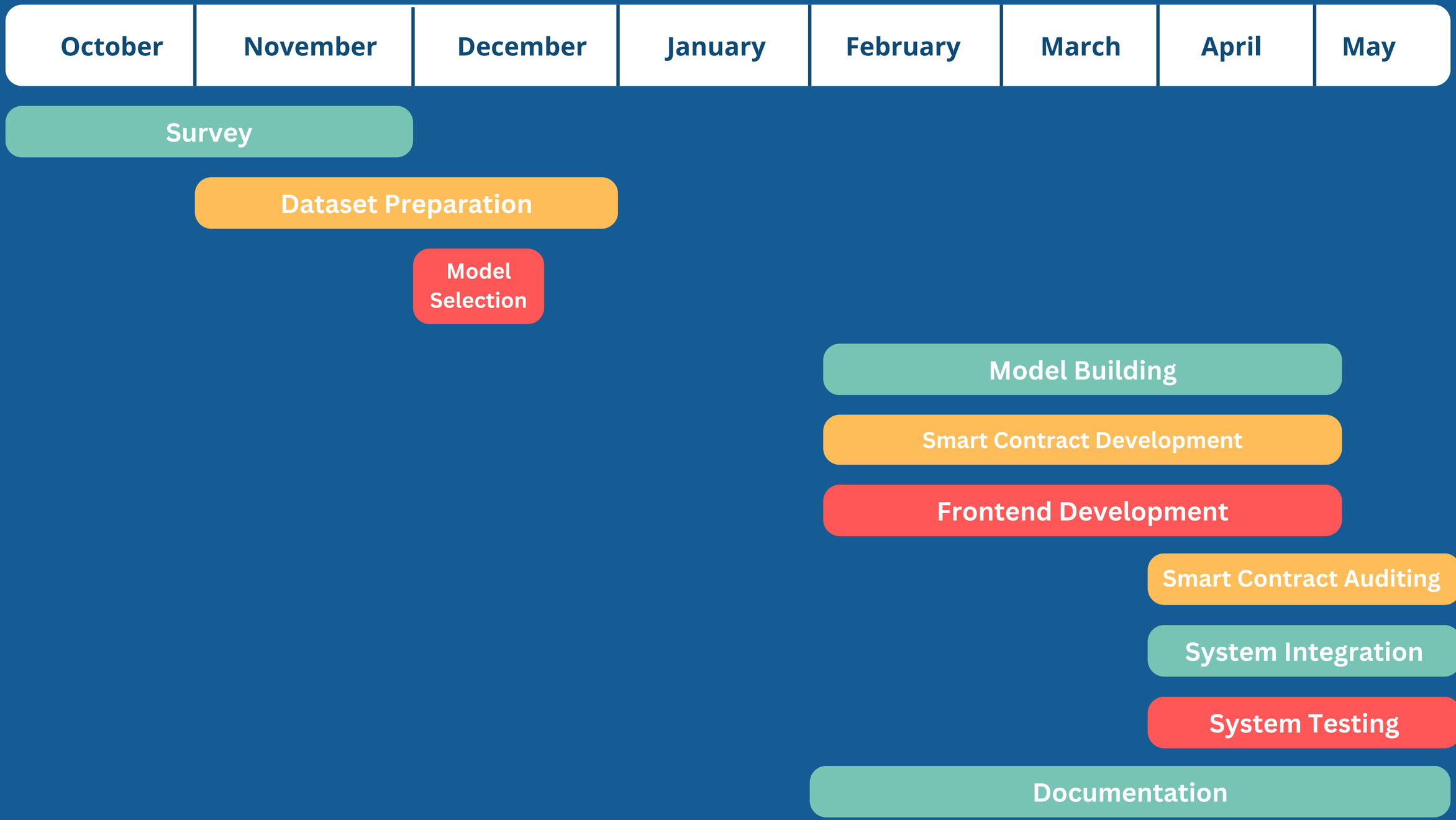
DEMO TIME


MetaMask
Coinbase Wallet
Rabby
or
Username
National ID
Phone Number
Email
Upload National ID Front Image
No file chosen
Upload Your Image
No file chosen
Regist
Already have an account? Please Login

11

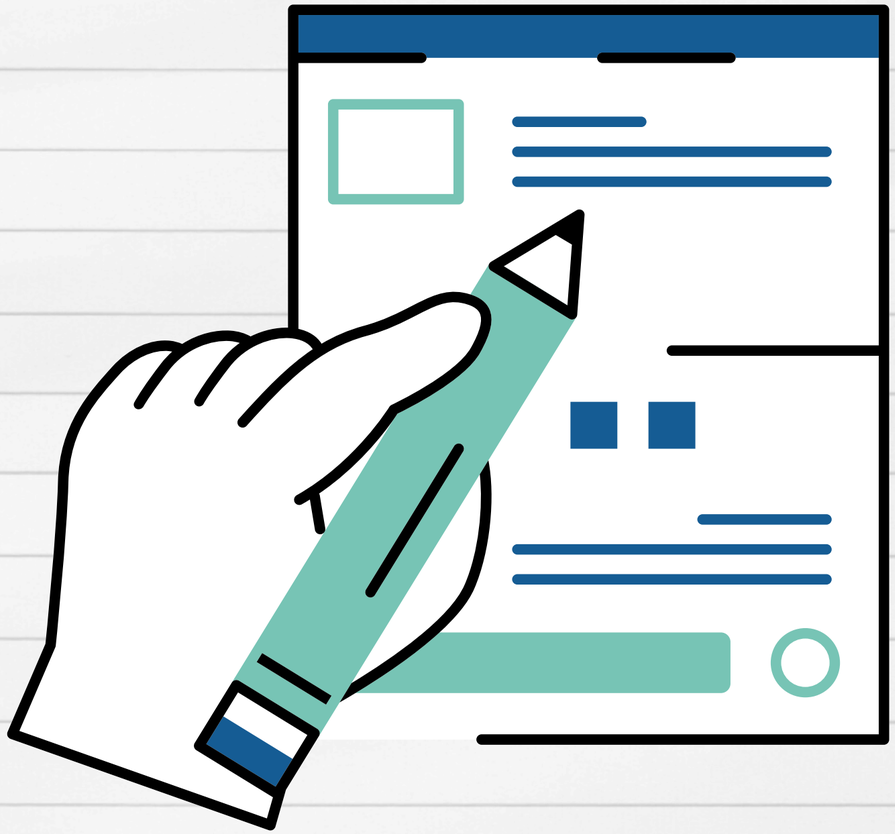
TIME PLAN




October
November
December
January
February
March
April
May
Survey
Dataset Preparation
Model Selection
Model Building
Smart Contract Development
Frontend Development
Smart Contract Auditing
System Integration
System Testing
Documentation

12

NEXT PHASE



NEXT PHASE

- **Models**
  - Improve Performance
  - Reduce Complexity
  - Change some Preprocessing Technique
  - Differentiate between different id cards

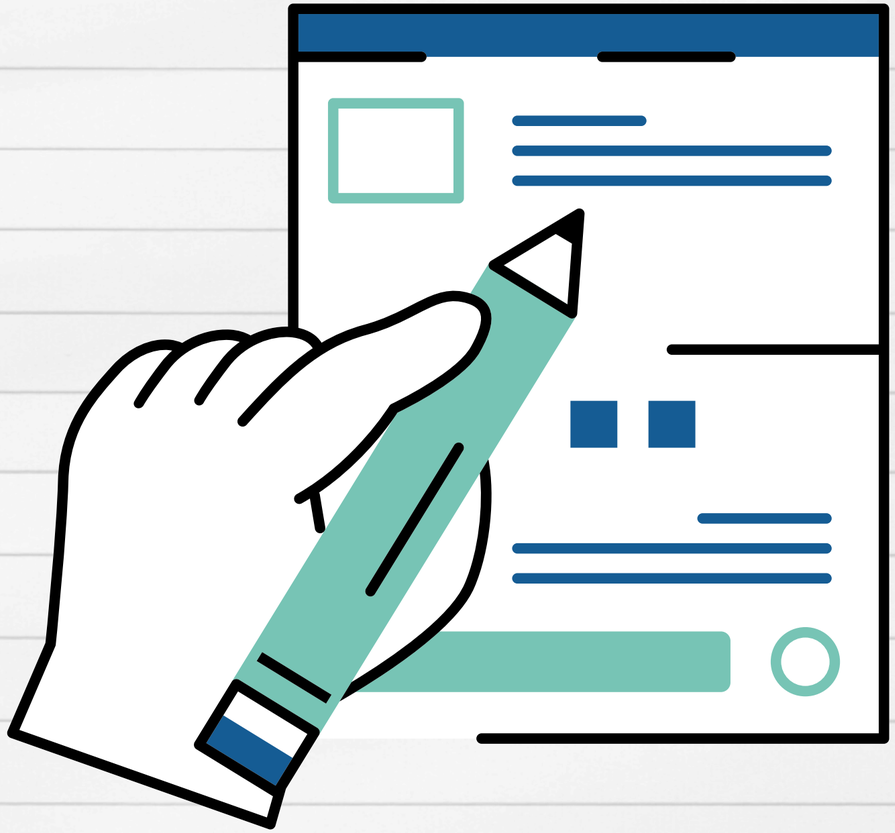
- **BlockChain**
  - Automatic change in the Proposal Status
    - Active, Pending, Closed
  - Determining the most voted Option.
  - Making the Contract Upgradable.
  - Encryption / Decryption Module

NEXT PHASE

- **Frontend**
  - Voter History Page
  - Active Votes Page

- **System**
  - Integration all The System Component
  - Test The System

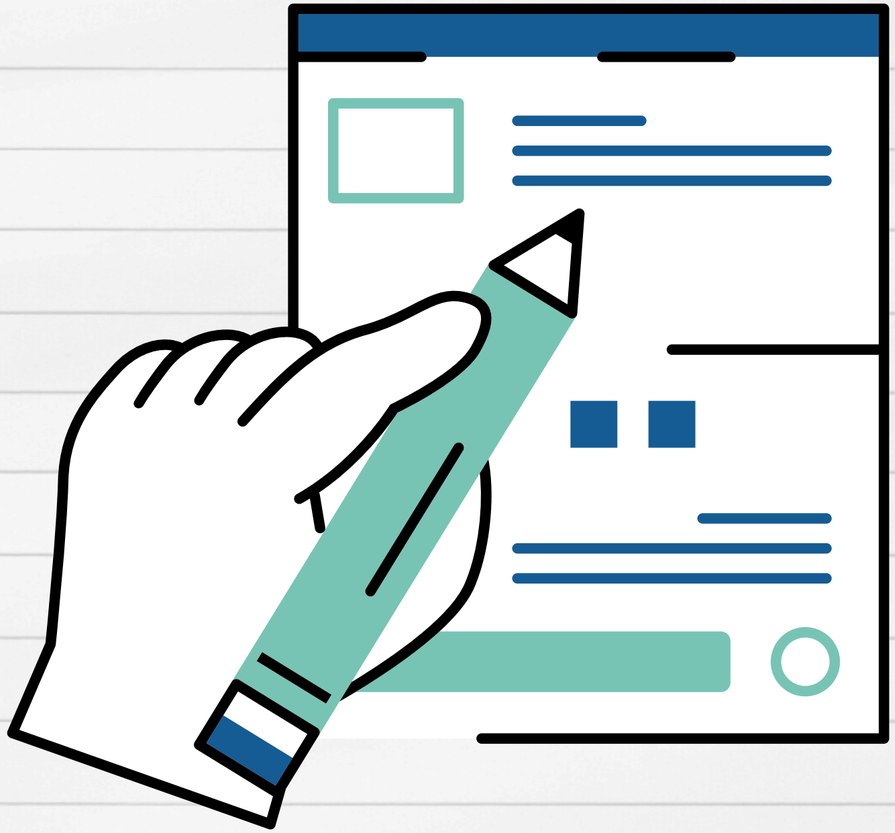
TOOLS



TOOLS

ZK-Sync	Solidity	Python	Angular
			

REFERENCES



REFERENCES

[1] “VGGFace2: A dataset for recognising faces across pose and age” Qiong Cao, Li Shen, Weidi Xie, Omkar M. Parkhi and Andrew Zisserman Visual Geometry Group, Department of Engineering Science, University of Oxford {qiong,lishen,weidi,omkar,az}@robots.ox.ac.uk

[2] “Siamese Neural Networks for One-shot Image Recognition” Gregory Koch, Richard Zemel , Ruslan Salakhutdinov, Department of Computer Science, University of Toronto. Toronto, Ontario, Canada. [https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf]

[3] “FaceNet: A Unified Embedding for Face Recognition and Clustering” Florian Schroff, Dmitry Kalenichenko , James Philbin, Google Inc. [https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Schroff_FaceNet_A_Unified_2015_CVPR_paper.pdf]

[4] “Learning Face Representation from Scratch” Dong Yi, Zhen Lei, Shengcai Liao and Stan Z. Li Center for Biometrics and Security Research & National Laboratory of Pattern Recognition Institute of Automation, Chinese Academy of Sciences (CASIA)

[5] “Learning Robust Deep Face Representation” Xiang Wu University of Science and Technology Beijing Beijing, China

REFERENCES

[6]“Voting System Using Blockchain (Face Recognition)” Gaddam Harsha Vardhan, Swapnil Shah, Vanshika Gupta, Rohithreddy.B.C, Tanya Bisht

[6] “E-Voting using Blockchain: Moving Away from the Ballot Paper” Irvine Mutandiwa, Calvin P Mugauri, Maronge Musara

[7] “Blockchain-Based Electronic Voting: A Secure and Transparent Solution” Bruno Miguel Batista Pereira, José Manuel Torres, Pedro Miguel Sobral, Rui Silva Moreira, Christophe Pinto de Almeida Soares, Ivo Pereira

[8] “Blockchain-Based E-Voting System with Face Recognition” V. Sathya Preiya, V. D. Ambeth Kumar, R. Vijay, Vijay K., N. Kirubakaran

[9] “A Blockchain-based Electronic Voting System: EtherVote” Achilleas Spanos Department of Informatics and Computer Engineering, University of West Attica Athens, Greece,
Ioanna Kantzavelou Department of Informatics and Computer Engineering, University of West Attica Athens, Greece

[10] “A Privacy-Preserving Blockchain-based E-voting System” Arnab Mukherjee, Souvik Majumdar, Anup Kumar Kolya, Saborni Nandi

THANK YOU VERY MUCH!