

Vehicle Speed Estimation & Management using Blockchain

Project submitted to the
SRM University – AP, Andhra Pradesh
for the partial fulfillment of the requirements to award the degree of
Bachelor of Technology

In
Computer Science and Engineering
School of Engineering and Sciences

Submitted by
Nanda Kishore Nallagopu (AP19110010157)
Vaishnavi Datla (AP19110010374)
Yamini Bhargavi Alapati (AP19110010296)



Under the Guidance of
Dr Amit Kumar Singh
Assistant Professor
Department of Computer Science and Engineering
SRM University-AP
Neerukonda, Mangalagiri, Guntur
Andhra Pradesh – 522 240
June, 2023

Certificate

Date: 23-May-23

This is to certify that the work present in this Project entitled “**Vehicle Speed Estimation & Management using Blockchain**” has been carried out by **Nanda Kishore Nallagopu, Vaishnavi Datla, Yamini Bhargavi Alapati** under me. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of **Bachelor of Technology School of Engineering and Sciences**.

Supervisor

Dr. Amit Kumar Singh

Assistant Professor,

Department of Computer Science and Engineering.

Acknowledgements

I would like to extend my sincere gratitude to everyone who helped this **Vehicle Speed Estimation & Management using Blockchain** project be completed successfully. Throughout the process, their assistance, direction, and encouragement were vital.

I want to start by expressing my sincere gratitude to Dr. Amit Kumar Sing for his constant support and knowledgeable direction. His astute recommendations, persistent encouragement, and helpful criticism were crucial in shaping our work in a perfect way.

I want to express my gratitude to my team members for their hard work and dedication. Their inputs, creative ideas, and labor of love were crucial to the creation and implementation of this website.

I appreciate the SRM University AP, faculty members for creating a supportive learning atmosphere and giving us the information and skills, we needed to complete this project.

I also want to thank the open-source community for making the many technologies and frameworks utilized in this project accessible. Their contributions and ongoing development work have greatly improved the project's capabilities.

I would like to conclude by expressing my gratitude to everyone who has contributed, whether directly or indirectly. This "Vehicle Speed Estimation & Management using Blockchain" project would not have been possible without their assistance and efforts.

Table of Contents

Certificate	2
Acknowledgements	3
Abstract	5
Abbreviations	7
List of Figures	8
Introduction	10
Methodology	13
Discussion	16
Concluding Remarks.....	25
Future Work	27
References.....	29

Abstract

This project aims to develop a system for vehicle speed estimation and management using blockchain technology. The system utilizes computer vision techniques to track vehicles in a video, estimate their speeds, and record the data on a blockchain network. The integration of blockchain technology ensures secure and tamper-proof storage of vehicle speed information, allowing for transparent and reliable management.

The system begins by allowing the user to upload a video containing vehicle footage. Computer vision algorithms, including object detection and tracking, are applied to detect and track vehicles in the video frames. The system utilizes the Dlib library for object tracking and OpenCV for image processing tasks.

Once the vehicles are tracked, their speeds are estimated based on the distance travelled between consecutive frames and the frame rate of the video. The calculated speeds are then recorded in a blockchain network, ensuring the integrity and immutability of the data. The Ethereum blockchain is used for this purpose, with a smart contract deployed to store and retrieve vehicle speed details.

The system provides a user-friendly interface implemented using the Tkinter library in Python. The interface allows users to upload videos, track vehicle speeds, fetch and display recorded vehicle details from the blockchain, and exit the application.

This project combines computer vision, blockchain technology, and user interface development to create a comprehensive solution for vehicle speed estimation and management. It offers potential applications in traffic monitoring, law enforcement, and transportation management systems. The use of blockchain ensures data integrity and enhances trust in the recorded vehicle speed information.

Abbreviations

CV2	OpenCV(Open Source Computer Vision)
XML	Extensible Markup Language
JSON	JavaScript Object Notation
HTTP	Hyper Text Transfer Protocol
IP	Internet Protocol
FPS	Frames Per Second
ID	Identification
HTML	Hypertext Markup Language
GUI	Graphical User Interface
Tk	Tkinter
API	Application Programming Interface
ABI	Application Binary Interface
UI	User Interface
ETH	Ethereum

List of Figures

Figure 1: Architecture

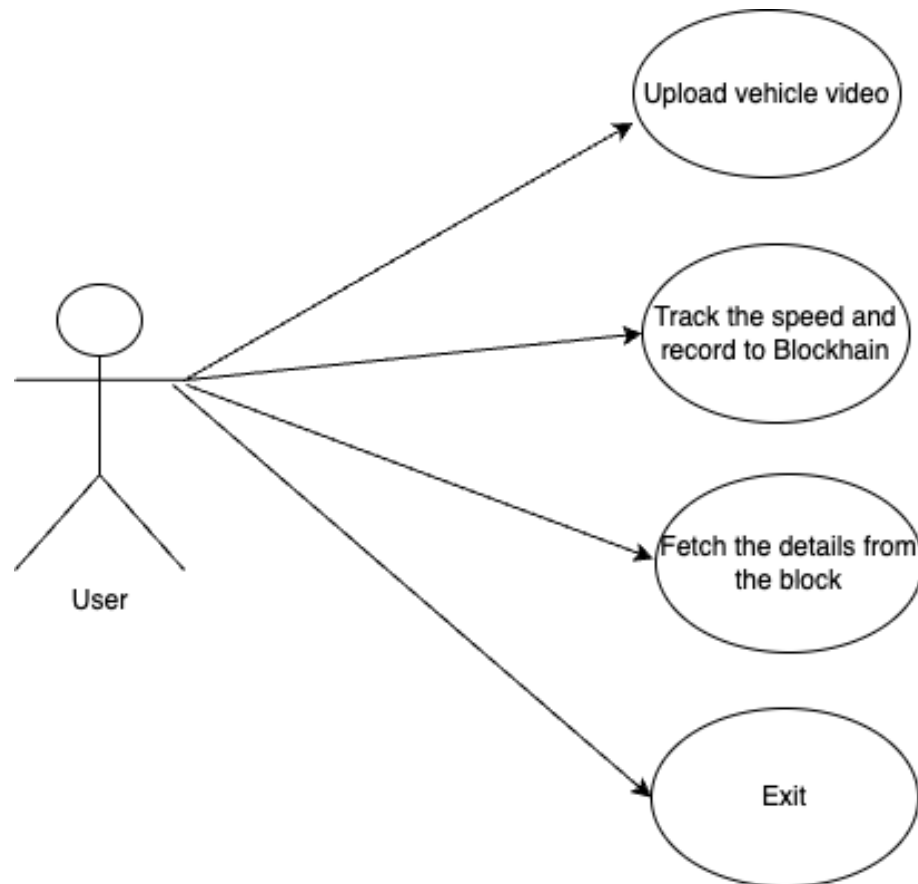


Figure 2: Sequence diagram for fetching data from blockchain

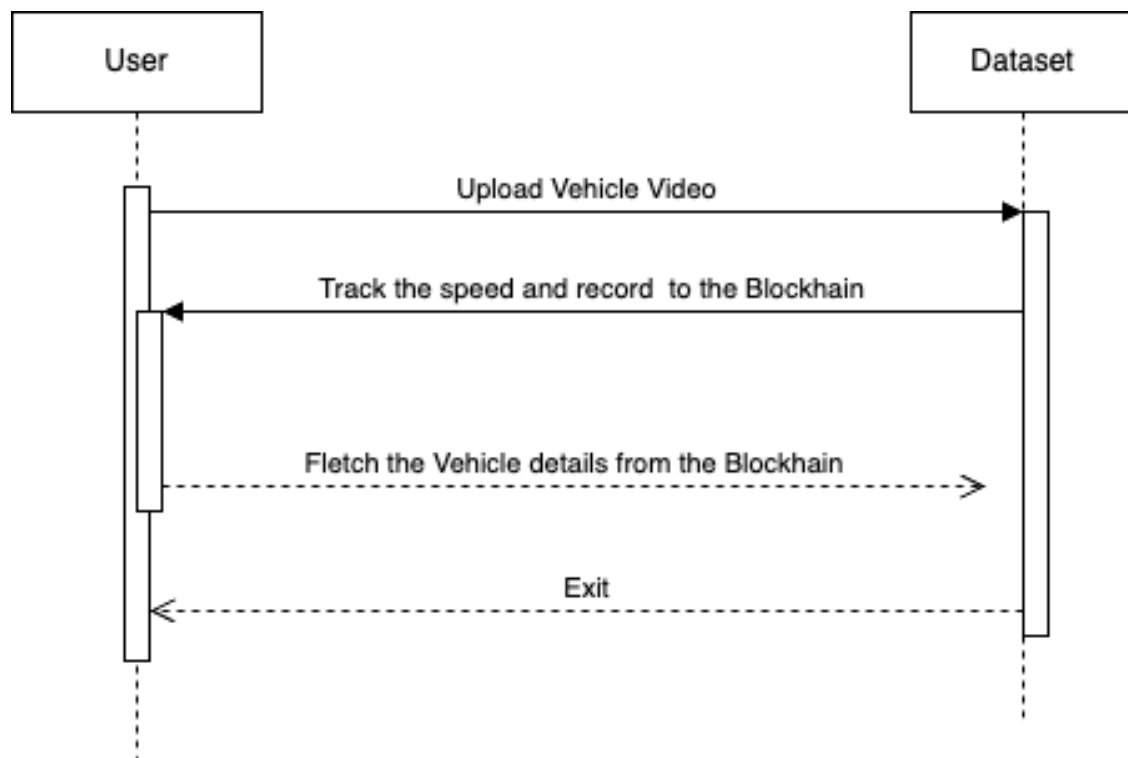
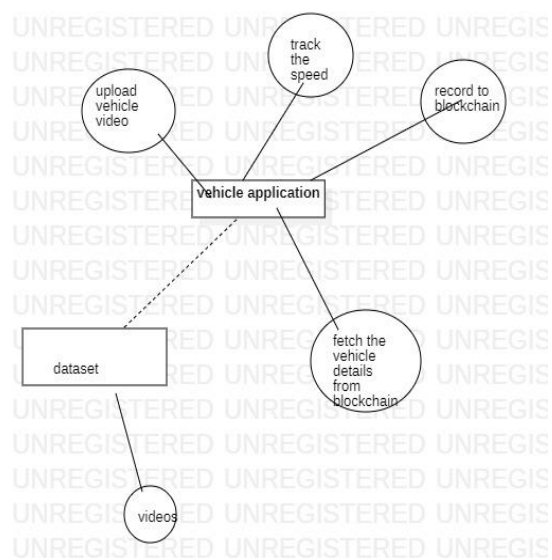


Figure 3: ER diagram for the model



Introduction

The rapid advancements in technology have transformed the way we interact with vehicles and manage traffic data. As vehicles traverse the roadways, capturing and preserving accurate information about their speeds and timestamps have become crucial for various purposes, including investigations, research, and decision-making processes. However, the integrity and security of such data have become significant concerns in today's centralized data management systems.

The "Vehicle Speed Estimation & Management using Blockchain" project presents a groundbreaking solution to address these challenges. By leveraging the power of computer vision techniques, specifically Python's OpenCV and Haar Cascade, this project aims to develop a robust system capable of accurately estimating and securely storing vehicle speeds and timestamps. Unlike conventional methods, this system offers unparalleled data integrity and protection against manipulation, ensuring the trustworthiness of the collected information.

In addition to employing computer vision technologies, the project integrates blockchain, a decentralized and tamper-proof ledger, to further enhance data security. By utilizing blockchain technology, the system establishes an immutable record of the captured vehicle speed and timestamp data. This integration ensures that the information remains secure, transparent, and tamper-proof, making it suitable for critical applications such as investigations, legal proceedings, and research endeavors.

The objectives of this project encompass multiple facets. First and foremost, the system focuses on providing a secure data collection process by leveraging computer vision techniques. The accurate estimation of vehicle speeds and timestamps in real-time is achieved through the utilization of Haar Cascade and OpenCV. The collected data is then securely stored on the blockchain, ensuring its immutability and protection against unauthorized modifications.

Furthermore, the project aims to develop a robust mechanism for efficient data storage and retrieval. Authorized users will have seamless access to historical vehicle speed and timestamp data through a user-friendly interface. This interface will enable users to retrieve, visualize, and analyze the data, facilitating investigations, research, and decision-making processes.

The significance of the "Vehicle Speed Estimation & Management using Blockchain" project lies in its ability to revolutionize the way we secure, manage, and utilize critical traffic data. By ensuring the integrity and authenticity of the collected information, this project holds the potential to enhance investigations, facilitate accurate research outcomes, and enable informed decision-making in various domains that rely on precise vehicle speed and timestamp data.

1.1 Project Overview

The "Vehicle Speed Estimation & Management using Blockchain" project aims to develop a robust and secure system for capturing and preserving accurate data related to vehicle speeds and timestamps. By employing computer vision techniques with Python, OpenCV, and Haar Cascade, the system provides real-time analysis of video data to estimate the speeds of vehicles on the road. The primary objective is to ensure the integrity and security of the collected data, enabling its utilization for various purposes such as investigations and research.

1.2 Background

In the current landscape of traffic data management, relying on a centralized server poses several challenges. There is a potential risk of data manipulation or unauthorized access, compromising the reliability of the information. Additionally, the dependence on a single server introduces vulnerabilities, as service availability is contingent on the server's operation. To address these concerns and enhance data security, a more resilient and tamper-proof approach is necessary.

1.3 Objectives

The key objectives of this project are as follows:

1.3.1 Secure Data Collection

Implement computer vision techniques, specifically utilizing Haar Cascade and OpenCV, to accurately capture and record vehicle speeds and timestamps in real-time. The system ensures the integrity and immutability of the collected data, preventing any unauthorized modifications or tampering.

1.3.2 Blockchain Integration

Leverage the power of blockchain technology to enhance the security and transparency of the system. By utilizing a decentralized and immutable ledger, the project aims to create a tamper-proof record of the vehicle speed and timestamp data. This integration ensures the authenticity and trustworthiness of the collected information, making it suitable for investigative or legal purposes.

1.3.3 Data Storage and Retrieval

Develop a robust mechanism for efficient storage and retrieval of the collected vehicle speed and timestamp data. The system will provide secure and scalable storage capabilities, enabling authorized users to access historical data for investigative purposes, research, or other relevant applications.

1.3.4 User Interface and Accessibility

Create an intuitive and user-friendly interface that allows authorized users to interact with the system effectively. The interface will provide functionalities for data retrieval, visualization, and analysis, ensuring seamless access to the secured vehicle speed and timestamp data.

1.4 Significance of the Project

The "Vehicle Speed Estimation & Management using Blockchain" project addresses the critical need for secure and reliable data management in the context of vehicle speeds and timestamps. By leveraging computer vision techniques and blockchain technology, the system ensures the integrity, authenticity, and accessibility of the collected data. This project holds the potential to strengthen investigations, research endeavors, and decision-making processes that rely on accurate and unalterable vehicle speed and timestamp information.

Methodology

The methodology employed in this project encompasses a systematic approach to achieve the objectives of securing and accurately managing vehicle speed and timestamp data. This chapter outlines the key steps involved in the process, from data collection and speed estimation to ensuring data integrity and security through blockchain integration. Additionally, it highlights the development of a user-friendly interface for data management, evaluation of system performance, and considerations for privacy and deployment. By following this methodology, the project aims to provide a reliable and robust solution for the estimation and secure management of vehicle speeds and timestamps.

2.1 Data Collection

The first step in the methodology involves the collection of vehicle speed and timestamp data. This is achieved through the implementation of computer vision techniques, specifically utilizing Python's OpenCV library and Haar Cascade classifiers. The system captures video footage from traffic surveillance cameras or other relevant sources. The video frames are processed using Haar Cascade classifiers to detect and track vehicles within the footage. Through this process, the system extracts the necessary information such as vehicle positions and speeds.

2.2 Speed Estimation

Once the vehicles are detected and tracked, the next step is to estimate their speeds accurately. The system employs various algorithms and techniques, including frame differencing, optical flow, or other velocity estimation methods. By analyzing the changes in vehicle positions over consecutive frames, the system calculates the speed of each vehicle in real-time. The estimated speeds are then associated with the respective timestamps.

2.3 Data Integrity and Security

One of the primary objectives of this project is to ensure the integrity and security of the collected data. To achieve this, the system integrates blockchain technology. Each recorded vehicle speed and timestamp entry is securely stored in a decentralized and tamper-proof blockchain ledger. The blockchain provides immutability and

transparency, preventing any unauthorized modification or tampering of the data. This ensures the authenticity and trustworthiness of the captured information.

2.4 User Interface and Data Management

To facilitate efficient data storage, retrieval, and analysis, a user-friendly interface is developed. Authorized users can access the system through the interface, which provides functionalities for data management and exploration. Users can retrieve historical speed and timestamp data, visualize the data in various formats, and perform data analysis tasks. The interface also allows users to search and filter data based on specific criteria, enabling seamless access to the required information.

2.5 System Evaluation and Performance Analysis

To assess the effectiveness and accuracy of the developed system, thorough evaluation and performance analysis are conducted. Real-world scenarios are simulated, and the system's performance in different traffic conditions is measured. Metrics such as speed estimation accuracy, processing time, and system reliability are evaluated. The results are analyzed to identify potential areas for improvement and to validate the system's effectiveness in securely estimating and managing vehicle speeds and timestamps.

2.6 Security and Privacy Considerations

In addition to data integrity, security, and blockchain integration, this project also addresses security and privacy concerns. Measures are implemented to safeguard sensitive information and ensure compliance with relevant data protection regulations. Encryption techniques may be employed to protect data during transmission and storage, and access controls are implemented to restrict unauthorized access to the system and its data.

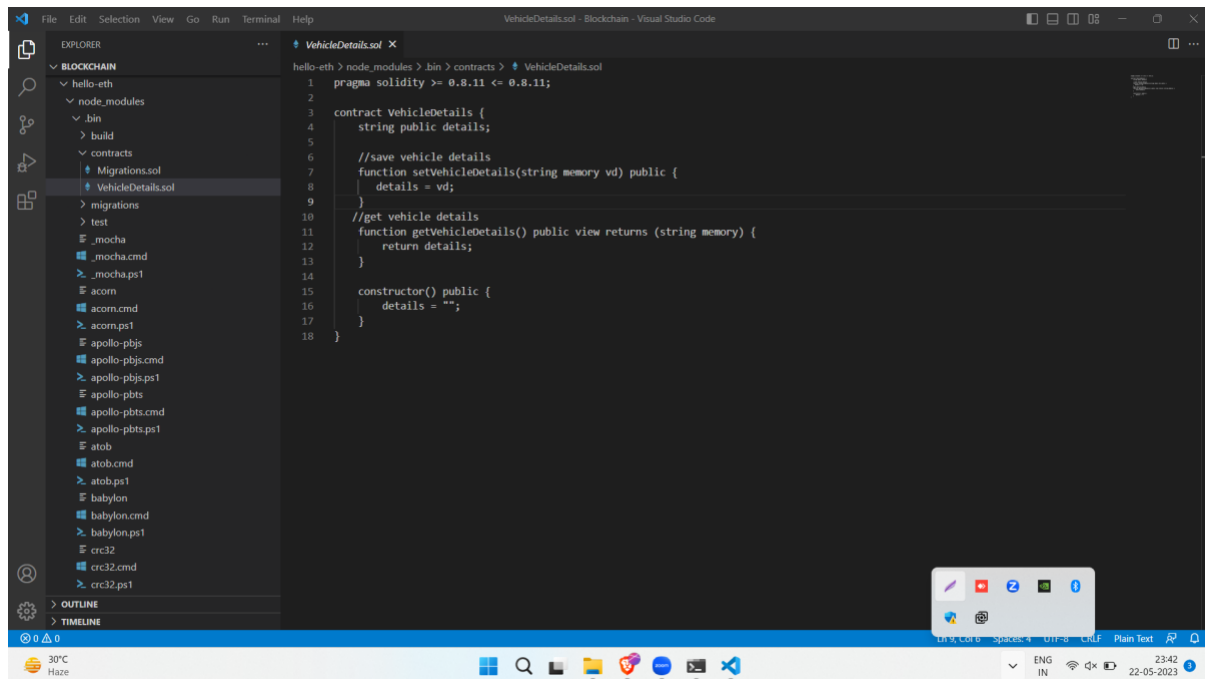
2.7 Project Deployment and Integration

Upon successful development and evaluation, the system is deployed and integrated into the existing infrastructure or relevant applications. This may involve collaboration with relevant stakeholders, such as traffic management authorities or surveillance system providers. The deployment process includes thorough testing, documentation, and user training to ensure a smooth transition and successful adoption of the system.

The methodology outlined above provides a comprehensive approach to achieving the objectives of the "Vehicle Speed Estimation & Management using Blockchain" project. By combining computer vision techniques, blockchain integration, and user-friendly interfaces, this methodology ensures the secure and accurate estimation and management of vehicle speeds and timestamps, addressing the need for reliable and tamper-proof traffic data for various applications.

Discussion

To store and retrieve data securely, we have designed a Blockchain smart contract using Solidity programming language. The contract includes functions for storing and retrieving vehicle details. Here is the Solidity code for the contract:



```
hello-eth > node_modules > bin > contracts > VehicleDetails.sol
1  pragma solidity >= 0.8.11 <= 0.8.11;
2
3  contract VehicleDetails {
4      string public details;
5
6      //save vehicle details
7      function setVehicleDetails(string memory vd) public {
8          details = vd;
9      }
10
11     //get vehicle details
12     function getVehicleDetails() public view returns (string memory) {
13         return details;
14     }
15
16     constructor() public {
17         details = "";
18     }
19 }
```

To deploy the contract and utilize its functionalities for data storage and retrieval, we follow the steps below:

Navigate to the 'hello-eth/node-modules/bin' folder.

Locate the 'runBlockchain.bat' file.

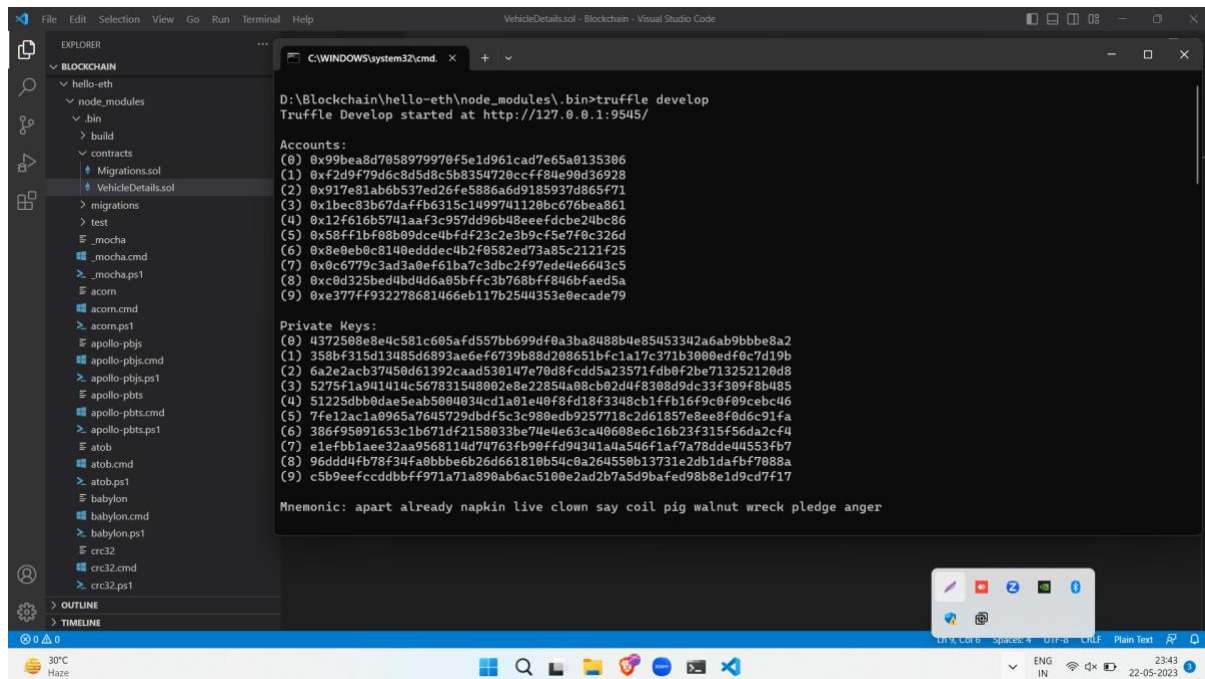
Double-click on the 'runBlockchain.bat' file to initiate the deployment process.

The 'runBlockchain.bat' file launches a script that sets up a local blockchain network environment. This environment enables the deployment of the smart contract and interaction with it. Once the deployment is successful, you will be provided with a screen displaying the blockchain network details and transaction information.

By deploying the smart contract to the blockchain, we establish a secure and tamper-proof platform for storing and retrieving vehicle details. The contract's functions can be accessed through the deployed contract address, allowing authorized users to store

and retrieve data securely. This ensures data integrity and provides a reliable audit trail for future investigations or reference purposes.

To deploy the smart contract to the Ethereum Blockchain, follow these steps after launching the 'runBlockchain.bat' file and accessing the blockchain network:



```
D:\Blockchain\hello-eth\node_modules\.bin>truffle develop
Truffle Develop started at http://127.0.0.1:9545/

Accounts:
(0) 0x99bea8d7058979970f5e1d961cad7e65a0135306
(1) 0xf2d9f79d6c8d5d8c5b8354720ccff84e90d36928
(2) 0x917e81ab6b537ed26fe5886a6d9185937d865f71
(3) 0x1bec83b67daffb6315c1499741120bc676bea861
(4) 0x12f616b5741aaf3c957dd96b48eeefdcbe24bc86
(5) 0x58ff1bf08b09dce4bfdf23c2e3b9cf5e7f0c326d
(6) 0x8e0eb0c8140edddc4b2f0582ed73a85c2121f25
(7) 0x0c6779c3ad3a0ef61ba7c3dbc2f97ede4e6643c5
(8) 0xc0d325bed4bd4d6a05bffc3b768bfbf846bfaed5a
(9) 0xe377ff932278681466eb117b2544353e0ecade79

Private Keys:
(0) 4272508e8e4c581c605afd557bb699df0a3ba8488b4e85453342a6ab9bbbe8a2
(1) 358bf315d13485d6893ae6ef6730b88d208651bfc1a17c371b3000edf0c7d19b
(2) 6a2e2acb37450d61392caad530147e70d8fcd5a23571fdb0f2be713252120d8
(3) 5275f1a941414c567831548002e8e22854a08cb02d4f8308d9dc33f309f8b485
(4) 51225dbb0dae5eab5004834cd1a01e40f8fd18f3348cb1ffb16f9c0f09cebc46
(5) 7fe12ac1a0965a7645729dbdf5c3c980ed9257718c2d61857e8ee8f0d6c91fa
(6) 386f95091653c1b671df2158033be74e4e63ca0608e6c16b23f315f56da2cf4
(7) e1efbb1aee32aa9568114d74763fb90ff94341a4a546f1af7a78dde44553fb7
(8) 96ddd4fb78f34fa0bbbe6b26d66180b54c0a264550b13731e2db1dafb7088a
(9) c5b9eefccddbbff971a71a890ab6ac5100e2ad2b7a5d9bafed98b8e1d9cd7f17

Mnemonic: apart already napkin live clown say coil pig walnut wreck pledge anger
```

After the network initialization, you will be presented with a command prompt or terminal window.

In the command prompt, type the command 'migrate' and press the enter key.

The migration process will begin, deploying the smart contract to the Ethereum Blockchain.

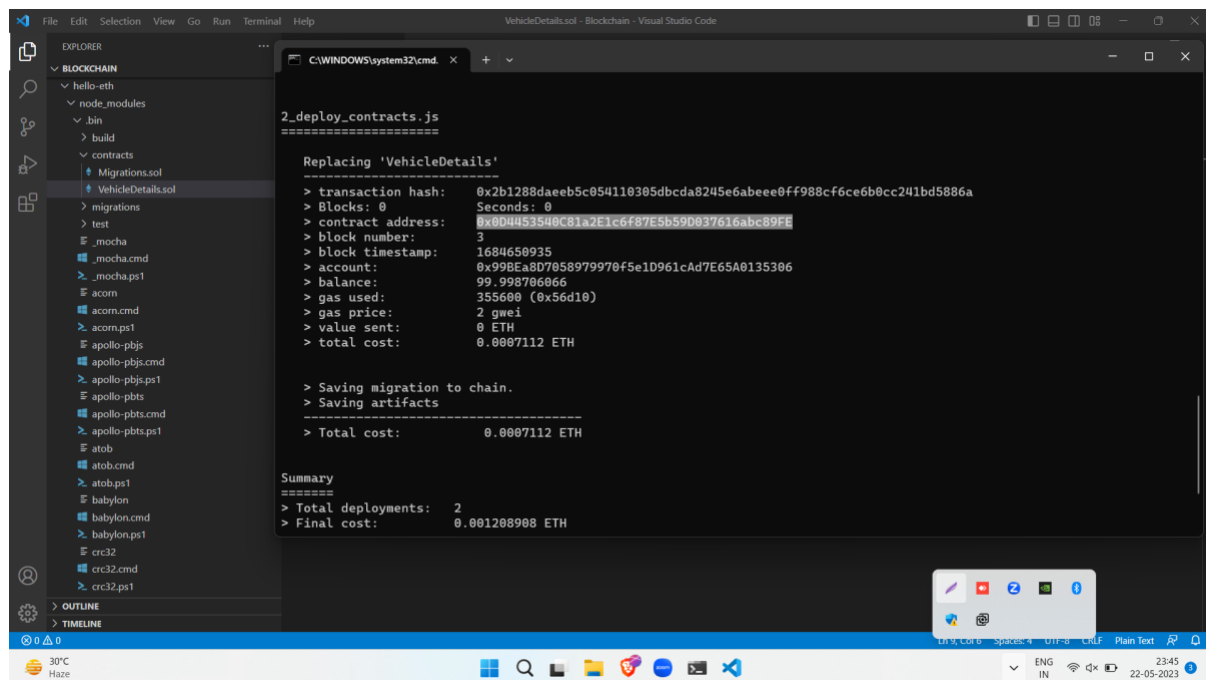
During the migration, you will see progress updates and transaction information displayed in the command prompt.

Once the migration is complete, you will receive a confirmation message indicating the successful deployment of the contract to the blockchain.

The 'migrate' command triggers the deployment process, which involves compiling the smart contract, creating a transaction, and interacting with the Ethereum network to deploy the contract. The process may take some time, depending on network conditions and the complexity of the contract.

After the contract is successfully deployed, it will be assigned a unique address on the Ethereum Blockchain. This address can be used to interact with the contract and access its functions for storing and retrieving vehicle details securely.

After deploying the Vehicle Details contract to the blockchain, we obtain a contract address. This address is essential for interacting with the contract in Python code to store and retrieve data from the blockchain.



```
2_deploy_contracts.js
=====

Replacing 'VehicleDetails'
> transaction hash: 0x2b1288daeeb5c854110305dbcd8245e6abee0ff988cf6ce6b0cc241bd5886a
> Blocks: 0 Seconds: 0
> contract address: 0x0D4453540C61a2E1c6f87E5b59D037616abc89F8
> block number: 3
> block timestamp: 1684650935
> account: 0x998Ea8D7058979970f5e1D961cAd7E65A0135306
> balance: 99.998706066
> gas used: 355600 (0x56d10)
> gas price: 2 gwei
> value sent: 0 ETH
> total cost: 0.0007112 ETH

> Saving migration to chain.
> Saving artifacts
=====
> Total cost: 0.0007112 ETH

Summary
=====
> Total deployments: 2
> Final cost: 0.001208908 ETH
```

In the Python code snippet, we establish a connection to the Ethereum network by specifying the blockchain address. We then load the contract's ABI (Application Binary Interface) from the compiled contract JSON file. The ABI contains the contract's function signatures and other relevant information required to interact with it.

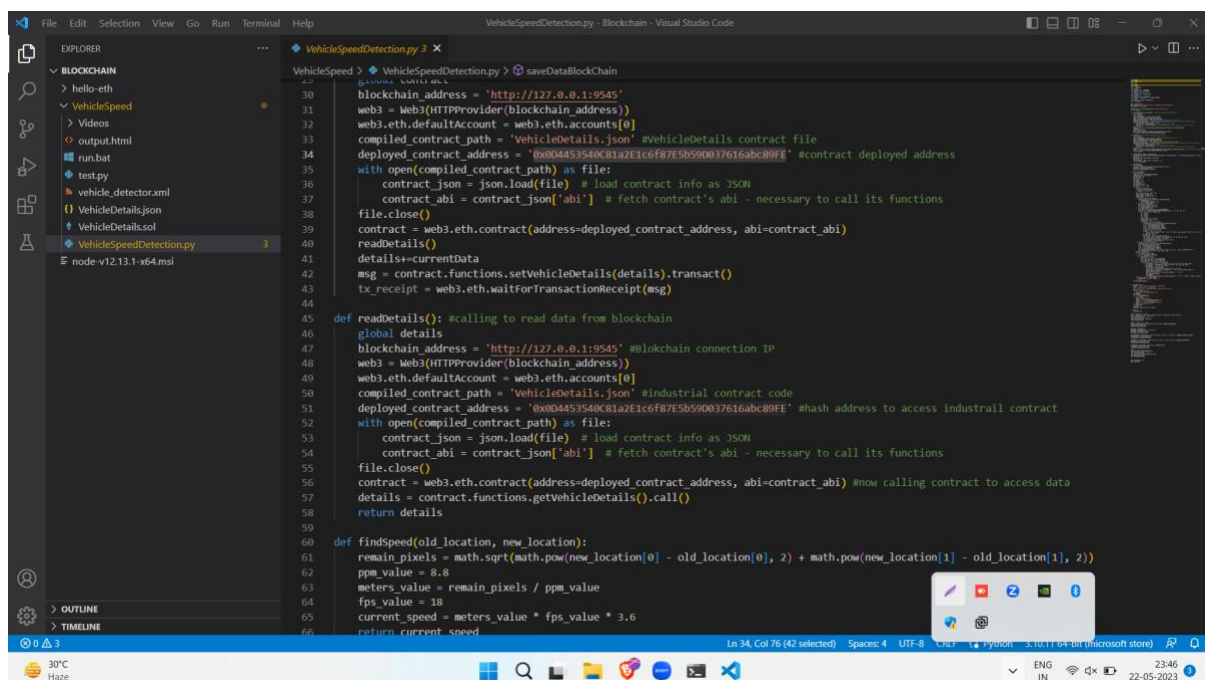
Using the contract address and ABI, we create an instance of the contract. This instance allows us to call the contract's functions and access its data. In the provided example, we make use of the `getVehicleDetails()` function to retrieve vehicle details from the blockchain and the `setVehicleDetails()` function to store vehicle details to the blockchain.

By executing these function calls, we can interact with the deployed contract and perform operations such as storing and retrieving vehicle details in a decentralized manner. This ensures that the data is securely stored and accessible on the blockchain network.

It's important to note that the specific details, such as the blockchain address, contract address, and contract ABI, would vary depending on your deployment environment

and the specific contract you are working with. Make sure to replace the placeholder values in the code with the actual values obtained during your deployment process.

In the provided Python code, you can identify comments highlighted in red that explain the process of calling the Blockchain contract. These comments provide valuable insights into the steps involved in interacting with the contract using Python. By reading these comments, you can gain a better understanding of how the code is utilizing the Web3 library to connect to the blockchain and interact with the deployed contract.



```
VehicleSpeedDetection.py - Blockchain - Visual Studio Code

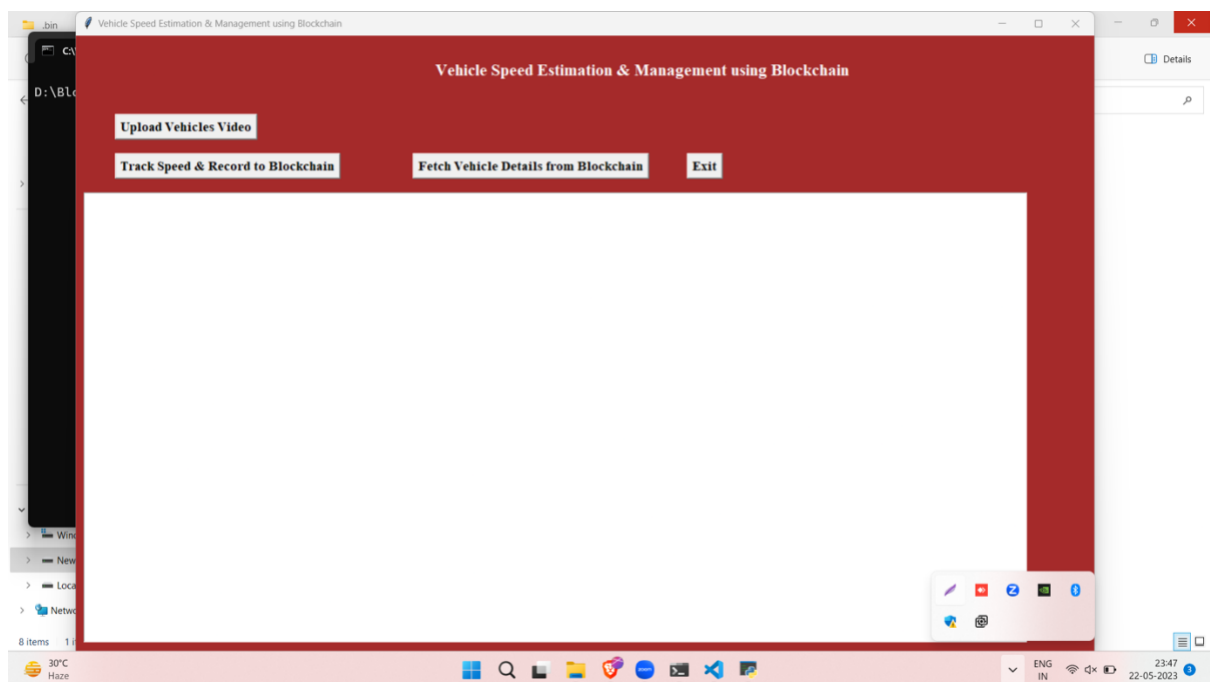
EXPLORER
  BLOCKCHAIN
    hello-eth
    VehicleSpeed
    Videos
    output.html
    run.bat
    test.py
    vehicle_detector.xml
    VehicleDetails.json
    VehicleDetails.sol
    VehicleSpeedDetection.py
    node-v12.13.1-x64.msi

VehicleSpeedDetection.py
30 blockchain_address = 'http://127.0.0.1:9545'
31 web3 = Web3(HTTPProvider(blockchain_address))
32 web3.eth.defaultAccount = web3.eth.accounts[0]
33 compiled_contract_path = 'VehicleDetails.json' #VehicleDetails contract file
34 deployed_contract_address = '0x004453540C81a2E1ceF87E5b590037616abc89FE' #contract deployed address
35 with open(compiled_contract_path) as file:
36     contract_json = json.load(file) # load contract info as JSON
37     contract_abi = contract_json['abi'] # fetch contract's abi - necessary to call its functions
38     file.close()
39 contract = web3.eth.contract(address=deployed_contract_address, abi=contract_abi)
40 readDetails()
41 details=currentData
42 msg = contract.functions.setVehicleDetails(details).transact()
43 tx_receipt = web3.eth.waitForTransactionReceipt(msg)
44
45 def readDetails(): #calling to read data from blockchain
46     global details
47     blockchain_address = 'http://127.0.0.1:9545' #blockchain connection IP
48     web3 = Web3(HTTPProvider(blockchain_address))
49     web3.eth.defaultAccount = web3.eth.accounts[0]
50     compiled_contract_path = 'VehicleDetails.json' #industrial contract code
51     deployed_contract_address = '0x004453540C81a2E1ceF87E5b590037616abc89FE' #hash address to access industrail contract
52     with open(compiled_contract_path) as file:
53         contract_json = json.load(file) # load contract info as JSON
54         contract_abi = contract_json['abi'] # fetch contract's abi - necessary to call its functions
55         file.close()
56     contract = web3.eth.contract(address=deployed_contract_address, abi=contract_abi) #now calling contract to access data
57     details = contract.functions.getVehicleDetails().call()
58     return details
59
60 def findSpeed(old_location, new_location):
61     remain_pixels = math.sqrt(math.pow(new_location[0] - old_location[0], 2) + math.pow(new_location[1] - old_location[1], 2))
62     ppm_value = 8.8
63     meters_value = remain_pixels / ppm_value
64     fps_value = 18
65     current_speed = meters_value * fps_value * 3.6
66     return current_speed
```

Comments play a crucial role in code documentation and understanding. They provide explanations, clarifications, and instructions to make the code more readable and understandable for developers. In this case, the red color highlights draw attention to the specific comments related to calling the Blockchain contract, ensuring that they are easily identifiable and can be followed for a comprehensive understanding of the code's functionality.

First, navigate to the project directory and locate the 'run.bat' file. Once you have located it, you can proceed with the following steps.

To initiate the execution of the project, simply double-click on the 'run.bat' file. This will launch the necessary processes and dependencies required for the project to run successfully.

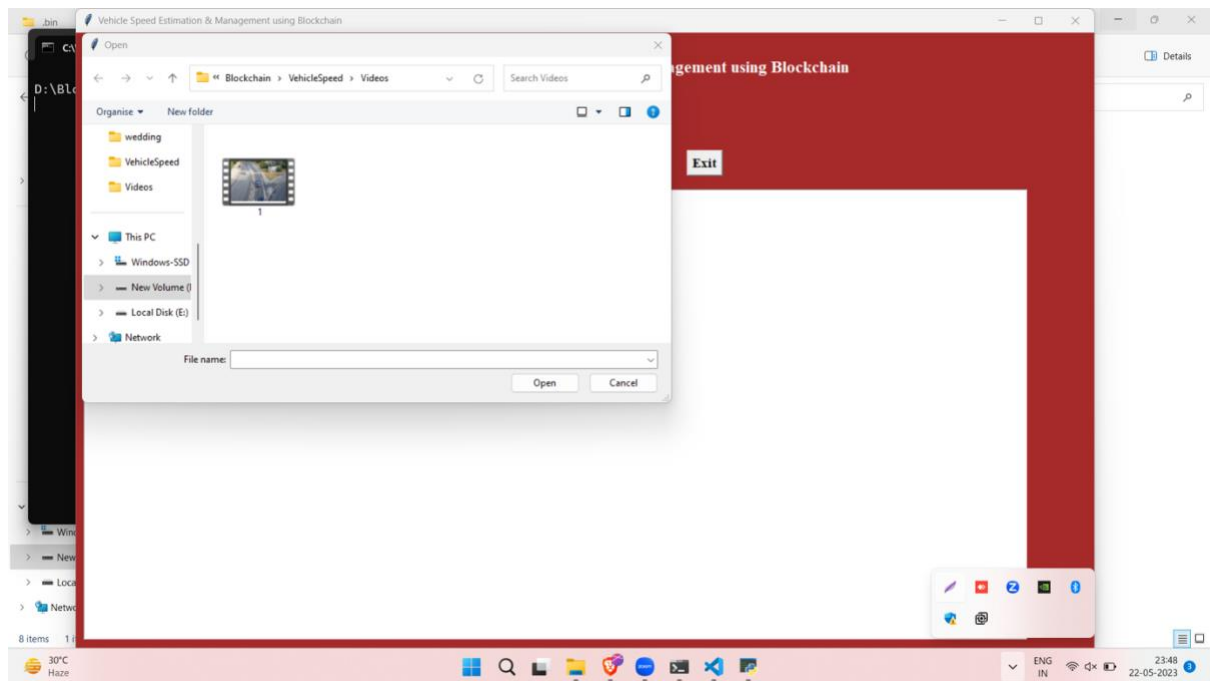


As the project starts running, a command prompt window or terminal will open, displaying the output and any relevant information or prompts. Pay attention to the information presented in this window, as it may provide important details about the project's progress or any user interactions required.

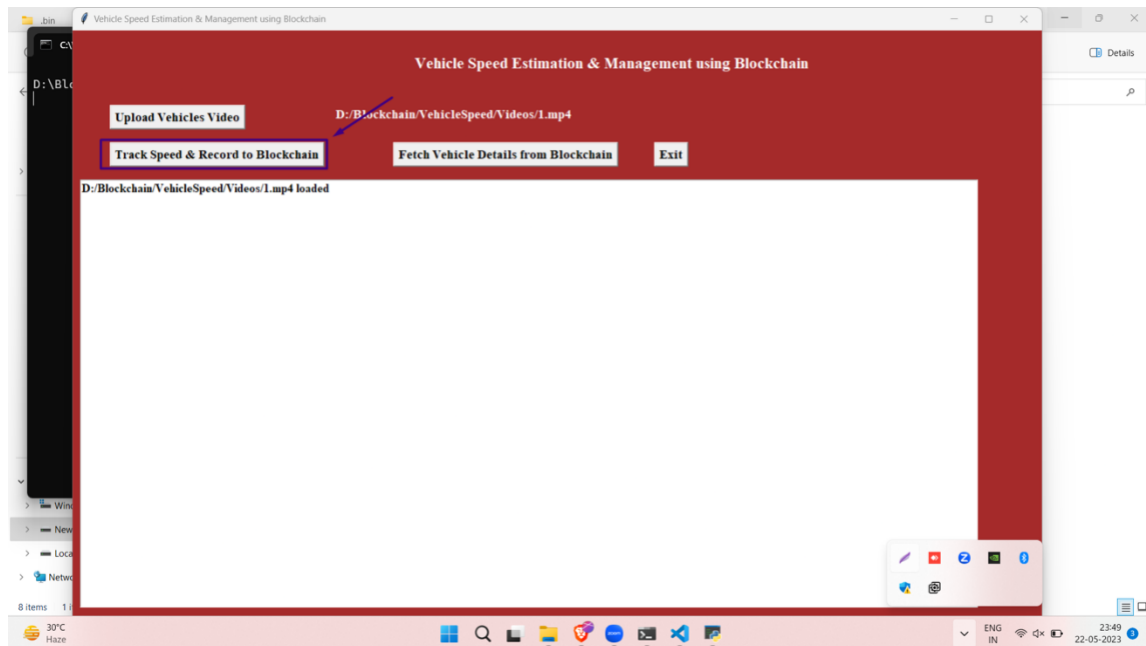
You can now interact with the project according to its functionality. Follow the instructions presented in the command prompt window, provide any necessary inputs, and observe the program's behavior.

By following these steps, you will be able to run the project and explore its features and functionalities.

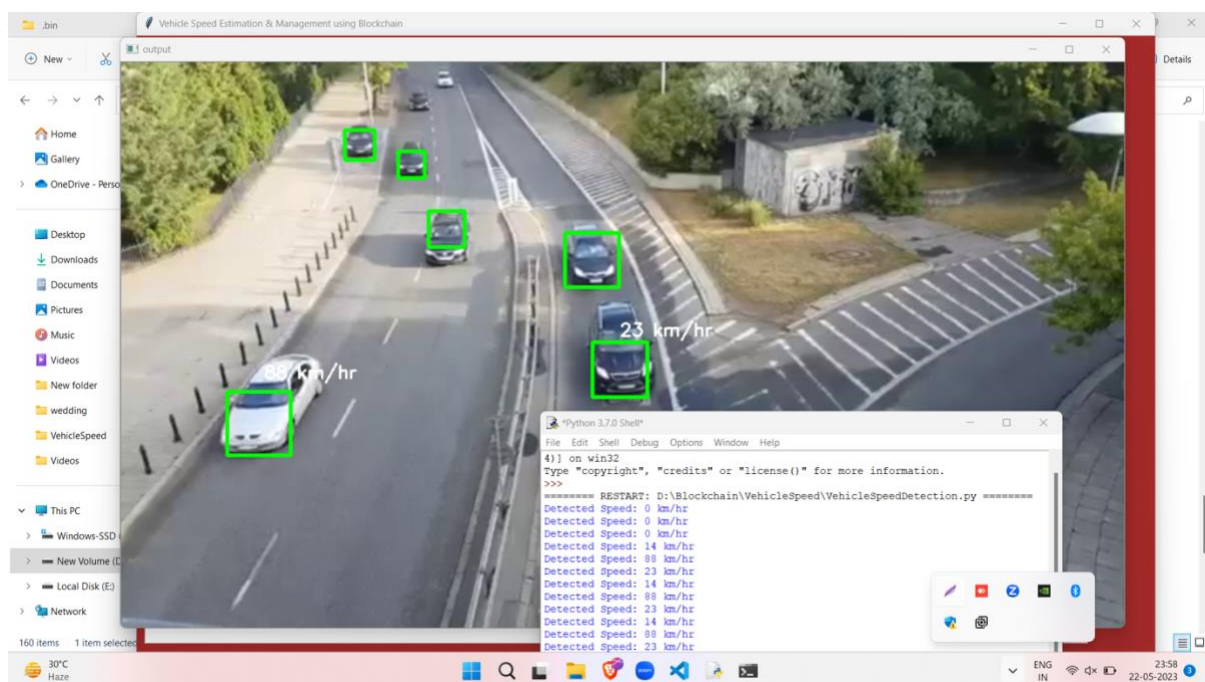
After selecting and uploading a vehicle video and clicking the "Open" button, the program will load the video and provide the corresponding output.



After loading the video, the program processes each frame of the video to detect and track vehicles. It uses computer vision techniques and the dlib library to identify and track vehicles in the video frames. The program applies a vehicle detection algorithm using a pre-trained cascade classifier to locate vehicles in each frame.



After clicking on the "Track Speed & Record to Blockchain" button, the program initiates the vehicle speed estimation and recording process. It analyzes each frame of the uploaded video and tracks the vehicles present in the footage. Using computer vision techniques and the dlib library, the program detects and tracks the vehicles in the video frames.



As the program progresses, it calculates the speed of each tracked vehicle based on the changes in their positions between consecutive frames. By utilizing a predetermined pixels-per-meter (ppm) value and the frame rate (fps) of the video, the program estimates the speed of each vehicle in kilometers per hour.

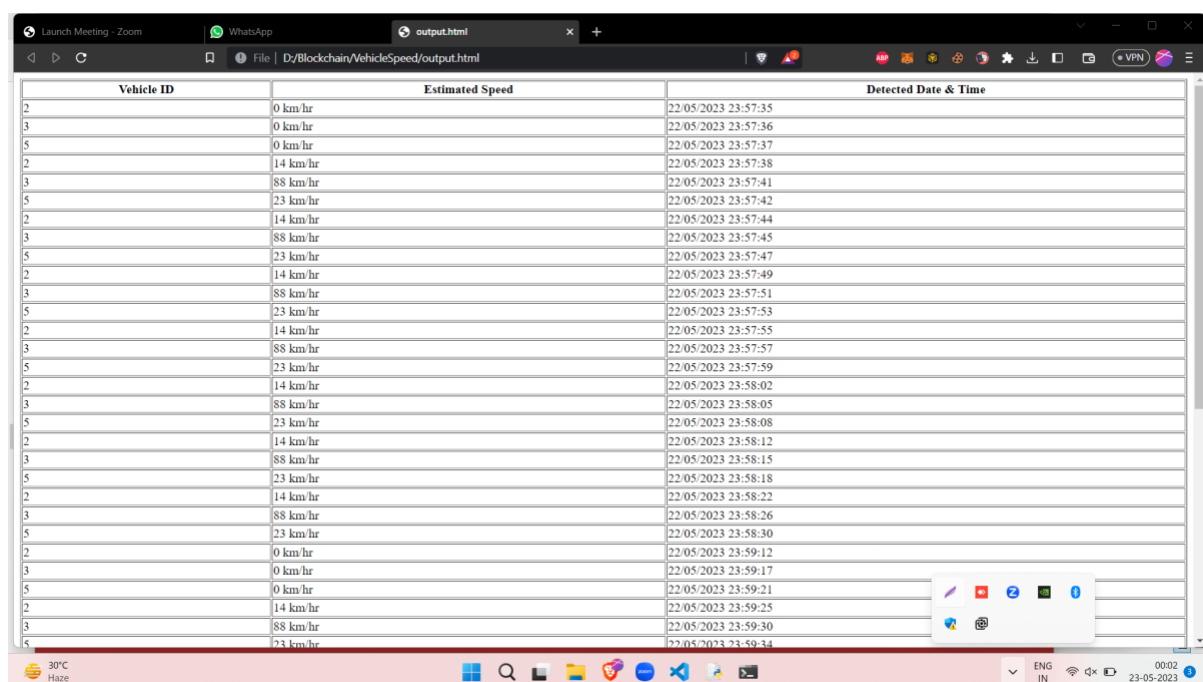
The calculated vehicle speeds are then displayed on the video frames and printed in the console for visual and informational purposes. Additionally, the program securely records the vehicle details, including the vehicle ID, estimated speed, and the date and time of detection, in the blockchain.

By leveraging the blockchain technology, the program ensures the integrity and immutability of the recorded vehicle details. Each recorded entry is stored as a transaction on the blockchain, making it tamper-resistant and providing a transparent and decentralized storage solution.

The program communicates with the blockchain using the web3 library in Python. It connects to the specified blockchain network and interacts with the deployed Vehicle Details smart contract. The contract address is used to invoke the necessary functions to store the vehicle details securely on the blockchain.

The program provides the user with a comprehensive output, including the estimated vehicle speeds, the recorded vehicle details in the blockchain, and any relevant notifications or messages. This allows users to track and manage vehicle speed data efficiently while leveraging the benefits of blockchain technology for data integrity and security.

Additionally, the program generates an HTML output that displays the recorded vehicle details, including the vehicle ID, estimated speed, and the date and time of detection. This HTML output provides a convenient and organized way to review and analyze the recorded data.



The screenshot shows a web browser window with the address bar displaying 'D:/Blockchain/VehicleSpeed/output.html'. The browser window has tabs for 'Launch Meeting - Zoom', 'WhatsApp', and 'output.html'. The table displayed has three columns: 'Vehicle ID', 'Estimated Speed', and 'Detected Date & Time'. The table contains 30 rows of data, showing vehicle IDs (2, 3, 5), estimated speeds (0 km/hr, 14 km/hr, 88 km/hr, 23 km/hr), and detected dates/times (22/05/2023 23:57:35 to 22/05/2023 23:59:34). The table is displayed in a light purple theme.

Vehicle ID	Estimated Speed	Detected Date & Time
2	0 km/hr	22/05/2023 23:57:35
3	0 km/hr	22/05/2023 23:57:36
5	0 km/hr	22/05/2023 23:57:37
2	14 km/hr	22/05/2023 23:57:38
3	88 km/hr	22/05/2023 23:57:41
5	23 km/hr	22/05/2023 23:57:42
2	14 km/hr	22/05/2023 23:57:44
3	88 km/hr	22/05/2023 23:57:45
5	23 km/hr	22/05/2023 23:57:47
2	14 km/hr	22/05/2023 23:57:49
3	88 km/hr	22/05/2023 23:57:51
5	23 km/hr	22/05/2023 23:57:53
2	14 km/hr	22/05/2023 23:57:55
3	88 km/hr	22/05/2023 23:57:57
5	23 km/hr	22/05/2023 23:57:59
2	14 km/hr	22/05/2023 23:58:02
3	88 km/hr	22/05/2023 23:58:05
5	23 km/hr	22/05/2023 23:58:08
2	14 km/hr	22/05/2023 23:58:12
3	88 km/hr	22/05/2023 23:58:15
5	23 km/hr	22/05/2023 23:58:18
2	14 km/hr	22/05/2023 23:58:22
3	88 km/hr	22/05/2023 23:58:26
5	23 km/hr	22/05/2023 23:58:30
2	0 km/hr	22/05/2023 23:59:12
3	0 km/hr	22/05/2023 23:59:17
5	0 km/hr	22/05/2023 23:59:21
2	14 km/hr	22/05/2023 23:59:25
3	88 km/hr	22/05/2023 23:59:30
5	23 km/hr	22/05/2023 23:59:34

Furthermore, the project leverages blockchain technology for secure data storage and retrieval. A smart contract, implemented in Solidity, is deployed to the Ethereum blockchain. This contract contains functions to store and retrieve vehicle details. By interacting with the blockchain contract using Python code, the project ensures the

immutability and transparency of the recorded data. The contract address is obtained during the deployment process and can be specified in the Python code for seamless integration with the blockchain.

Overall, the combination of HTML output generation and blockchain integration enhances the functionality and reliability of the project, providing an efficient solution for vehicle speed estimation and management.

Concluding Remarks

In conclusion, the Vehicle Speed Estimation & Management project utilizing blockchain technology presents a robust and innovative solution for tracking and managing vehicle speed. The project incorporates computer vision techniques, data processing, and blockchain integration to deliver accurate speed estimation and secure data storage. This project holds great potential for various real-world applications, such as traffic monitoring, law enforcement, and transportation management.

By leveraging computer vision algorithms, the project effectively detects and tracks vehicles in video footage. The integration of the OpenCV and Dlib libraries enables the accurate identification of vehicles and their movements. The speed estimation algorithm calculates the speed based on the distance traveled by the vehicles over time, providing valuable insights into vehicle velocity.

To ensure data integrity and transparency, the project integrates blockchain technology. The deployment of a smart contract on the Ethereum blockchain allows for secure storage and retrieval of vehicle details. The smart contract serves as a decentralized ledger, enabling tamper-proof record-keeping and enhancing trust in the recorded data. By interacting with the blockchain contract using Python code, the project enables seamless integration with the blockchain, facilitating data management and analysis.

The project's user interface, implemented using the Tkinter library, offers a user-friendly experience for uploading videos, tracking vehicle speed, and retrieving

recorded details. The HTML output generated by the program provides a clear and organized presentation of the recorded vehicle data, aiding in data analysis and decision-making.

Overall, this project showcases the potential of combining computer vision, data processing, and blockchain technology to create an efficient and secure system for vehicle speed estimation and management. The integration of these technologies opens up possibilities for enhancing traffic control, road safety, and transportation efficiency. Further advancements could include real-time monitoring, integration with IoT devices, and advanced analytics to derive valuable insights from the recorded data.

As technology continues to evolve, projects like this one highlight the importance of leveraging emerging technologies to address real-world challenges. The Vehicle Speed Estimation & Management project serves as a testament to the power of innovation and interdisciplinary approaches in creating solutions that can contribute to a safer and more efficient transportation ecosystem.

Future Work

In the future, several areas can be explored to further enhance and improve the real-time communication website. Some of the ideas are:

Real-Time Speed Monitoring: Enhance the project to enable real-time speed monitoring of vehicles. This could involve integrating the system with live video feeds from traffic cameras or surveillance drones. By analyzing the video stream in real-time, the system can provide immediate feedback on vehicle speeds and flag any instances of speeding.

Intelligent Traffic Management: Combine the speed estimation capabilities with intelligent traffic management systems. By integrating with existing traffic infrastructure and signals, the project can contribute to optimizing traffic flow and reducing congestion. The system could dynamically adjust signal timings based on estimated vehicle speeds to ensure efficient traffic movement.

Automated Traffic Law Enforcement: Develop the project further to automate traffic law enforcement processes. By integrating with automated license plate recognition (ALPR) systems, the project can identify and record speeding vehicles. This data can be used to issue automated traffic violation tickets, reducing the need for manual intervention and improving overall traffic safety.

Machine Learning for Speed Prediction: Utilize machine learning algorithms to improve the accuracy of speed estimation. By training the system on a large dataset of vehicle movements, it can learn patterns and factors affecting speed, such as vehicle type, road conditions, and time of day. This would result in more precise speed estimations and enhanced decision-making capabilities.

Integration with Connected Vehicles: Explore integration possibilities with connected vehicle technologies. By receiving speed and location data from vehicles equipped with Internet of Things (IoT) devices, the project can gather real-time speed information from a larger network of vehicles. This would enable comprehensive speed monitoring and analysis on a city or regional scale.

Blockchain Data Analytics: Develop advanced analytics capabilities for the recorded blockchain data. By applying data mining and data analytics techniques, valuable insights can be derived from the stored vehicle details. This could include identifying traffic patterns, detecting areas with high instances of speeding, and analyzing speed-related incidents to improve road safety measures.

Collaborative Traffic Management: Enable collaboration between vehicles, traffic infrastructure, and the proposed system to optimize traffic management. Vehicles equipped with connected technologies can communicate their speed and location information to the system, which can then provide feedback and guidance to drivers to maintain safe speeds and reduce congestion.

Integration with Autonomous Vehicles: Explore the integration of the project with autonomous vehicles. By leveraging vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication, the system can gather real-time speed information from autonomous vehicles and provide insights to improve their navigation and decision-making processes.

These future ideas illustrate the potential for further innovation and expansion of the **Vehicle Speed Estimation & Management project**. By embracing emerging technologies, such as real-time monitoring, machine learning, connected vehicles, and advanced analytics, the project can continue to evolve and contribute to creating smarter, safer, and more efficient transportation systems.

References

- [1] E. Atko FLXQDV, R. Blake, A. Juozapavičius, M. Kazimianec, Image Processing in Road Traffic Analysis, Nonlinear Analysis: Modelling and Control, 2005, Vol. 10, No. 33–315 ,4
- [2] Arash Gholami Rad, Abbas Dehghani and Mohamed Rehan Karim, “Vehicle speed detection in video image sequences using CVS method”, at International Journal of the Physical Sciences Vol. 5(17), pp. 2555-2563, 18 December, 2010 , ISSN 1992 - 1950 ©2010 Academic Journals
- [3] H. P. Moravec. Towards Automatic Visual Obstacle Avoidance. Proc. 5th International Joint Conference on Artificial Intelligence, pp. 584, 1977
- [4] Z.Zheng, H.Wang and EKTeoh. Analysis of Gray Level Corner Detection. Pattern Recognition Letters , Vol. 20, pp. 1999 ,149-162
- [5] Rafael C.Gonzalez and Richard E.Woods “Digital Image Processing” Prentice Hall, 2001
- [6] D.A. Forsyth, J. Ponce. “Computer Vision. A Modern Approach”, Prentice Hall, 2003
- [7] Open source computer vision library - reference manual, INTEL Corporation, 1999-2001.