GOPI BIRLA MEMORIAL SCHOOL

Computer Science Project File

BLOCKCHAIN VOTING SYSTEM

Ashutosh Pawar

XII Sci, 05

2024-2025

GOPI BIRLA MEMORIAL SCHOOL

Department of Computer Science

Certificate

This is to certify that the Computer Science Project completed by the Candidate: Ashutosh Pawar with Seat number:

for Std XII practical examination of the Central Board of Secondary Education(AISSCE) in the year 2024-2025.

It is further certified that this project is the individual work of the candidate.

Date:

Faculty's Signature Examiner's Signature Principal's Signature

Index

Acknowledgement

I would like to express my gratitude to my computer science teacher Ms. Jaini Shah who provided me with valuable advice and suggestions to improve my experiment

I am thankful to my principal Mrs. Madhu Wadke for providing me with the resources and facilities needed for my experiment. Their encouragement and motivation pushed me to work hard and strive for excellence.

I am grateful to my parents for their unwavering support throughout this project.

Ashutosh Pawar

XII Sci.

About Python

Python, a versatile high-level programming language, emphasises readability and simplicity. Created by Guido van Rossum, Python was first released in 1991. Its design focused on code clarity, making it easy to learn and use. With a clear syntax and dynamic typing, it's favoured for web development, data analysis, scientific computing, and more. Python's growth was fueled by its extensive standard library and third-party packages, enabling rapid development. It supports both procedural and object-oriented paradigms and promotes code reuse. Its interpreted nature allows for quick testing and debugging. Python's community and resources make it ideal for beginners and professionals, fostering learning and innovation.

About the project

Introduction

This program is a blockchain-based voting system designed for secure, transparent, and efficient voting. It leverages blockchain technology to ensure the integrity and transparency of the voting process, providing a tamper-proof system where votes are securely recorded and cannot be altered or deleted. The system is implemented using Python and MySQL, making it a powerful yet user-friendly tool for conducting digital elections.

Scope :

In modern elections, ensuring the security and integrity of votes is paramount. Traditional voting systems can be prone to tampering, mismanagement, and lack of transparency. This program addresses these issues by:

- **Security**: Votes are stored on the blockchain, making them immutable and resistant to tampering.
- **Transparency**: Every vote is recorded and visible in the blockchain ledger, ensuring full transparency.
- **Efficiency**: The system allows for quick and easy vote casting and counting, significantly reducing the time required for vote tallying.
- **Accessibility**: Voters and moderators can access the system remotely, allowing for broader participation and monitoring.

Usage Instructions

The program offers two main modes of operation: Voter Mode and Moderator Mode, each with distinct functionalities.

1. **Voter Mode**:
    - **Log In**: Existing voters can log in using their credentials.
    - **Sign Up**: New users can create an account and get registered to vote, each new account is saved in the database and 1 VoteCn is added to each user.
    - **Vote**: Logged-in voters can cast their votes for their preferred candidates. Each vote is securely recorded and added to the blockchain. The VoteCn is subtracted from their respective accounts in the database to prevent them from voting again.
2. **Moderator Mode**:
    - **Log In**: Moderators can log in to access administrative functionalities.
    - **Validate Blockchain**: Moderators can check the integrity of the blockchain, ensuring that all votes are correctly recorded by comparing it to the saved image of the blockchain before closing the servers.
    - **Flag and Fix Invalid Blocks**: If any inconsistencies are found via ‘Validate Blockchain’, moderators can flag and fix invalid blocks to maintain data integrity.
    - **Check Vote Tampering**: This feature allows moderators to verify that no votes have been tampered with after casting.
    - **Show Votes**: Displays the current voting statistics for all candidates.
    - **Stop Server**: Allows the moderator to shut down the server, ending the voting session securely.

3. **General Use**:
    - **Show Blockchain**: Displays the entire blockchain, showing all recorded blocks and transactions.

Modules:

The system is composed of several key components:

1. **Login and Registration System**: Manages user credentials and allows voters and moderators to access their respective functionalities.
2. **Voting System**: Allows voters to cast their votes securely, storing them in the blockchain.
3. **Blockchain Implementation**: Handles the creation, storage, and validation of blockchain records. Ensures that each vote is stored as a transaction in the blockchain.
4. **Moderation Tools**: Provides moderators with tools to check the integrity of the blockchain, fix any issues, and ensure that no votes have been tampered with.
5. **Data Management**: All data is stored in a MySQL database, ensuring secure and structured storage of user credentials, candidate details, and blockchain records.
6. **Set Data**: The moderator User Id and Password are premade to reduce the users accessing the moderator tools.

Licensing:

This project is licensed under the **GNU General Public License v3.0 (GPL-3.0)** to promote transparency, open collaboration, and community-driven development.

By using this license, the project remains free and open for anyone to use, modify, and redistribute, as long as the same rights are preserved in derivative works. This ensures that the core principles of security, transparency, and accessibility are maintained, allowing for continuous improvement and contribution from developers.

The GPL-3.0 license safeguards the project’s freedoms while encouraging innovation and wide adoption. Choosing this license ensures that the software remains freely available to everyone, fostering collaboration while preventing proprietary use, which aligns with the project's goal of maintaining an open and secure voting system.

Source Code–Python

[The following source code is broken down into 6 parts to make separate modules]

I] blockchain_main:

```
import hashlib

import time

# BLOCKCHAIN_VOTING_SYSTEM - Copyright (C) 2024 Ashutosh Pawar

#

# This program is free software: you can redistribute it and/or modify

# it under the terms of the GNU General Public License as published by

# the Free Software Foundation, either version 3 of the License, or

# (at your option) any later version.

#

# This program is distributed in the hope that it will be useful,

# but WITHOUT ANY WARRANTY; without even the implied warranty of

# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

# GNU General Public License for more details.

#

# You should have received a copy of the GNU General Public License

# along with this program. If not, see <https://www.gnu.org/licenses/>.


# Function to calculate the hash of a block

def calculate_hash(index, timestamp, transactions_hash, previous_hash):

    block_string = str(index) + str(timestamp) + str(transactions_hash) +
str(previous_hash)
```

```
    return hashlib.sha256(block_string.encode()).hexdigest()  # Creates a SHA-256 hash


# Function to hash transaction data
def hash_transaction(transaction):
    return hashlib.sha256(transaction.encode()).hexdigest()  # To keep the privacy of a voter


# Function to create a new block
def create_block(index, transaction, previous_hash):
    timestamp = int(time.time())  # Use integer for timestamp (Unix time)
    transactions_hash = hash_transaction(transaction)  # Hash the transactions for privacy
    block_hash = calculate_hash(index, timestamp, transactions_hash, previous_hash)

    block = {
        'index': index,
        'timestamp': timestamp,  # Save as integer
        'transactions_hash': transactions_hash,
        'previous_hash': previous_hash,
        'hash': block_hash
    }
    return block


# Function to create the genesis block
def create_genesis_block():
    return create_block(0, "Genesis Block", "0")
```

```
# Function to add a new block to the blockchain and save it to the database
def add_block(blockchain, transaction, connection, cursor):
    if blockchain is None:  # Double-check to ensure blockchain is initialised
        blockchain = (create_genesis_block(),)
    previous_block = blockchain[-1]  # Gets the last block in the blockchain
    index = previous_block['index'] + 1
    previous_hash = previous_block['hash']
    new_block = create_block(index, transaction, previous_hash)
    # Save the new block to the database
    cursor.execute("""
        INSERT INTO blockchain_metadata (block_index, timestamp,
transaction_hash, previous_hash, block_hash)
        VALUES (%s, %s, %s, %s, %s)
    """,
(new_block['index'],int(new_block['timestamp']),new_block['transactions_hash
'], new_block['previous_hash'], new_block['hash']))
    connection.commit()
    # Return the updated blockchain tuple
    return blockchain + (new_block,)

# Function to load the blockchain from the database in a list
def load_blockchain(cursor):
    try:
        cursor.execute("SELECT * FROM blockchain_metadata ORDER BY
block_index")
        rows = cursor.fetchall()

        blockchain = []
        for row in rows:
```

```
            block = {
                'index': row[0],
                'timestamp': row[1],
                'transactions_hash': row[2],
                'previous_hash': row[3],
                'hash': row[4]
            }
            blockchain.append(block)

        if blockchain:
            return tuple(blockchain)  # Return the blockchain if it's not empty
        else:
            return None  # Return None if there are no blocks in the database
    except Exception as e:
        print("Error loading blockchain:", e)
        return None

# Function to finally add the blocks to the main blockchain
def initialize_blockchain(cursor, connection):
    # Load blockchain, if empty, create the genesis block
    blockchain = load_blockchain(cursor)
    if blockchain is None:
        blockchain = (create_genesis_block(),)
        cursor.execute("""
            INSERT INTO blockchain_metadata (block_index, timestamp,
transaction_hash, previous_hash, block_hash)
            VALUES (%s, %s, %s, %s, %s)
```

```
        """, (blockchain[0]['index'], int(blockchain[0]['timestamp']),
blockchain[0]['transactions_hash'], blockchain[0]['previous_hash'],
blockchain[0]['hash']))
        connection.commit()
    return blockchain
```

II] main_libs_voters:

```
import random
# BLOCKCHAIN_VOTING_SYSTEM - Copyright (C) 2024 Ashutosh Pawar
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program. If not, see <https://www.gnu.org/licenses/>.

# Function for the Sign-in setup
def login_main(c):
    cursor = c
    print("Enter your user id (alphanumeric code provided at signing up)")
    username = input(">")
```

```
    print("Enter your password (case sensitive)")
    password = input(">")
    print("finding user....")
    query = "SELECT user_accountno FROM data_main WHERE user_accountno
= %s AND user_pass = %s"
    cursor.execute(query, (username, password))
    result = cursor.fetchone()
    if result:
        print("Access Granted")
        return result[0]  # Return the user ID
    else:
        print("Access Denied: Incorrect Username or Password")
        return None  # Return None if login failed


# Function for the Sign-up setup
def signup_main(c,con):
    cursor=c
    print("Enter your full name (as written on the aadhar card)")
    name=input('>')
    print("Enter your age (must be more than or equal to 18)")
    age=input('>')
    print("Enter your valid aadhar card number")
    aadharno=input('>')
    print("Enter your state")
    state=input('>')
    print("Your username is....")
    user=random.randrange(100000,999999)
    print(">",user)
```

```
    print("Enter your password (the password cannot be changed)")
    passwd=input('>')
    cursor.execute("use login_data")
    query= "insert into data_main
(user_name,user_age,user_aadhar,user_state,user_accountno,user_pass)
values(%s,%s,%s,%s,%s,%s)"
    cursor.execute(query,(name,age,aadharno,state,user,passwd))
    con.commit()
```

III] main_libs_mods:

```
import random

# BLOCKCHAIN_VOTING_SYSTEM - Copyright (C) 2024 Ashutosh Pawar
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program. If not, see <https://www.gnu.org/licenses/>.

#Function for the Login Setup of a Moderator
```

```
def login_main_mods(c):
    cursor=c
    print("Enter your user id")
    username=input(">")
    print("Enter your password")
    password=input(">")
    print("finding mod....")
    query = "SELECT * FROM mod_logs WHERE User_Id = %s AND pass = %s"
    cursor.execute(query, (username, password))
    result = cursor.fetchone()
    if result:
        print("Access Granted")
        return True
    else:
        print("Access Denied: Incorrect Username or Password")
```

IV] moderator_tools:

```
import blockchain_main
import pickle
import mysql.connector

# BLOCKCHAIN_VOTING_SYSTEM - Copyright (C) 2024 Ashutosh Pawar
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
```

```
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program. If not, see <https://www.gnu.org/licenses/>.

# Function to save the current state of the votes (or other data) before closing the server
def save_data_image(cursor, filename="vote_snapshot.dat"):
    cursor.execute("SELECT * FROM data_main")
    current_votes = cursor.fetchall()

    file = open(filename, 'wb')
    pickle.dump(current_votes, file)
    file.close()

    print("Data image saved successfully.")

# Function to stop the server, including saving the current data state
def stop_server(cursor, connection):
    save_data_image(cursor)
    print("Stopping server...")
    connection.close()
    print("Server stopped and connection closed.")
```

```
# Function to validate the blockchain by comparing each block with the database records
def validate_blockchain(cursor):
    blockchain = blockchain_main.load_blockchain(cursor)
    if blockchain is None:
        print("No blockchain found. Initialization might be needed.")
        return

    for i in range(1, len(blockchain)):
        current_block = blockchain[i]

        # Recompute the hash and compare
        expected_hash = blockchain_main.calculate_hash(
            current_block['index'],
            current_block['timestamp'],
            current_block['transactions_hash'],
            current_block['previous_hash']
        )

        if current_block['hash'] != expected_hash:
            print(f"Block {i} is invalid.")
            # Retrieve the metadata from the database
            cursor.execute("SELECT * FROM blockchain_metadata WHERE block_index = %s", (current_block['index'],))
            metadata = cursor.fetchone()
            if metadata:
                db_transaction_hash = metadata[2]
                db_block_hash = metadata[4]
```

```
            if current_block['transactions_hash'] == db_transaction_hash and
current_block['hash'] == db_block_hash:
                print(f"Block {i} matches the database records.")
            else:
                print(f"Block {i} does NOT match the database records.")
        else:
            print(f"No metadata found in the database for Block {i}.")

        choice = input("Do you want to delete this block? (yes/no): ")
        if choice.lower() == 'yes':
            return i  # Return the index of the invalid block

    print("Blockchain is valid.")
    return None

# Function to flag and fix an invalid block in the blockchain
def flag_and_fix_block(cursor):
    invalid_index = validate_blockchain(cursor)

    if invalid_index is not None:
        print(f"Block {invalid_index} is invalid.")
        choice = input("Do you want to delete this block? (yes/no): ")
        if choice.lower() == 'yes':
            blockchain = blockchain_main.load_blockchain()
            blockchain = blockchain[:invalid_index] + blockchain[invalid_index + 1:]
            for i in range(invalid_index, len(blockchain)):
                blockchain[i]['previous_hash'] = blockchain[i - 1]['hash']
```

```
            blockchain[i]['hash'] = blockchain_main.calculate_hash(
                blockchain[i]['index'],
                blockchain[i]['timestamp'],
                blockchain[i]['transactions_hash'],
                blockchain[i]['previous_hash']
            )
        blockchain_main.save_blockchain(blockchain)
        print("Invalid block removed and blockchain updated.")
        cursor.execute("DELETE FROM blockchain_metadata WHERE
block_index = %s", (invalid_index,))
        connection.commit()
    else:
        print("No changes made to the blockchain.")
else:
    print("No invalid blocks found.")


# Function to check for vote tampering by comparing the current votes with a
saved snapshot
def check_vote_tampering(cursor):
    try:
        with open('vote_snapshot.dat', 'rb') as file:
            saved_votes = pickle.load(file)
    except FileNotFoundError:
        print("No saved snapshot found.")
        return

    cursor.execute("SELECT * FROM data_main")
    current_votes = cursor.fetchall()
```

```
    if saved_votes == current_votes:
        print("No tampering detected.")
    else:
        print("Vote tampering detected!")

    with open('vote_snapshot.dat', 'wb') as file:
        pickle.dump(current_votes, file)

    print("Vote snapshot updated.")

def show_votes(cursor):
    cursor.execute("SELECT * FROM login_data.candidates")
    candidate_data = cursor.fetchall()  # Fetch all rows as tuples
    index = 1
    for candidate in candidate_data:
        print(str(index) + ". " + candidate[0] + " (" + candidate[1] + ") - " +
str(candidate[2]) + " votes")
        index += 1
```

V] vote_recorder:

```
import mysql.connector
import blockchain_main
from mysql.connector import Error
# BLOCKCHAIN_VOTING_SYSTEM - Copyright (C) 2024 Ashutosh Pawar
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
```

```
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program. If not, see <https://www.gnu.org/licenses/>.

# Function to fetch all names of the candidates from the database
def fetch_candidate_names(cursor):
    query = "SELECT name_cand, house_party FROM candidates"
    cursor.execute(query)
    results = cursor.fetchall()

    if results:
        candidates = []
        for row in results:
            candidates.append({'name': row[0], 'party': row[1]})
        return candidates
    else:
        print("No candidates found.")
        return []

# Function to check if the user has votes left
def can_vote(cursor, user_id):
```

```
    query = "SELECT VoteCn FROM data_main WHERE user_accountno = %s"
    cursor.execute(query, (user_id,))
    result = cursor.fetchone()

    if result is None:
        print("Error: User not found.")
        return False

    vote_count = result[0]
    if vote_count is None or vote_count == 0:
        print("You have no votes left.")
        return False

    return True

# Function for the main voting action
def vote(cursor, connection, user_id, blockchain):
    if not can_vote(cursor, user_id):
        return  # Exit if the user cannot vote

    candidates = fetch_candidate_names(cursor)

    if not candidates:
        print("No candidates available to vote for.")
        return

    print("Please select the candidate number to vote for:")
    for index in range(len(candidates)):
```

```
        print(str(index + 1) + ". " + candidates[index]['name'] + " (" +
candidates[index]['party'] + ")")

    choice = int(input("Enter your choice: "))

    if choice < 1 or choice > len(candidates):
        print("Invalid choice. Please try again.")
        return

    selected_candidate = candidates[choice - 1]['name']

    # Create a string representation of the transaction
    transaction = "User " + str(user_id) + " voted for " + selected_candidate

    blockchain = blockchain_main.add_block(blockchain, transaction,
connection, cursor)
    # Deduct a vote after voting
    cursor.execute("UPDATE data_main SET VoteCn = VoteCn - 1 WHERE
user_accountno = %s", (user_id,))
    cursor.execute("UPDATE candidates SET votes = votes + 1 WHERE name_cand
= %s", (selected_candidate,))
    connection.commit()

    print("\nBlockchain Updated:")
    print(blockchain[-1])
```

VI] main menu:

```
import vote_recorder
import main_libs_voters
```

```
import main_libs_mods
import blockchain_main
import moderator_tools
import mysql.connector
import pickle

def display_license():
    print("""
BLOCKCHAIN_VOTING_SYSTEM - Copyright (C) 2024 Ashutosh Pawar. This program comes with ABSOLUTELY NO WARRANTY. This is free software, and you are welcome to redistribute it under certain conditions. [GNU General Public License Version 3, 29 June 2007] For full details, visit: https://www.gnu.org/licenses/gpl-3.0.en.html
          """)
display_license()

def voter_menu(cursor, connection, blockchain):
    while True:
        print("\nVoter Menu:")
        print("1. Log In")
        print("2. Sign Up")
        print("3. Back")

        choice = input("Enter your choice: ")

        if choice == '1':
            user_id = main_libs_voters.login_main(cursor)
            if user_id:
                while True:
```

```
            print("\nLogged In as Voter:")
            print("1. Vote")
            print("2. Back")

            logged_in_choice = input("Enter your choice: ")

            if logged_in_choice == '1':
                vote_recorder.vote(cursor, connection, user_id, blockchain)
            elif logged_in_choice == '2':
                break
            else:
                print("Invalid choice. Please try again.")
        elif choice == '2':
            main_libs_voters.signup_main(cursor, connection)
        elif choice == '3':
            break
        else:
            print("Invalid choice. Please try again.")

def moderator_menu(cursor, connection, blockchain):
    while True:
        print("\nModerator Menu:")
        print("1. Log In")
        print("2. Back")
        print("WARNING!: If you close the server you can't run other operations,
you need to reopen the program..")

        choice = input("Enter your choice: ")
```

```
if choice == '1':
    mod_id = main_libs_mods.login_main_mods(cursor)
    if mod_id:
        while True:
            print("\nLogged In as Moderator:")
            print("1. Validate Blockchain")
            print("2. Flag and Fix Invalid Block")
            print("3. Check Vote Tampering")
            print("4. Stop Server")
            print("5. Show Votes")
            print("6. Back")

            mod_choice = input("Enter your choice: ")

            if mod_choice == '1':
                moderator_tools.validate_blockchain(cursor)
            elif mod_choice == '2':
                moderator_tools.flag_and_fix_block(cursor)
            elif mod_choice == '3':
                moderator_tools.check_vote_tampering(cursor)
            elif mod_choice == '4':
                moderator_tools.stop_server(cursor, connection)
            elif mod_choice == '5':
                moderator_tools.show_votes(cursor)
            elif mod_choice == '6':
                break
            else:
```

```
                print("Invalid choice. Please try again.")
        elif choice == '2':
            break
        else:
            print("Invalid choice. Please try again.")

# Establishing connection to the database
connection = mysql.connector.connect(
    host="192.168.29.204",
    user="remote_voteruse",
    password="ashu@105",
    database="login_data"
)
cursor = connection.cursor()

# Load blockchain using cursor
blockchain = blockchain_main.initialize_blockchain(cursor, connection)

# Main Menu Loop
while True:
    print("\nMain Menu:")
    print("1. Voter")
    print("2. Moderator")
    print("3. Show Blockchain")
    print("4. Exit")
    choice = input("Enter your choice: ")
    if choice == '1':
        voter_menu(cursor, connection, blockchain)
```

```
    elif choice == '2':
        moderator_menu(cursor, connection, blockchain)
    elif choice == '3':
        cursor.execute("SELECT * FROM blockchain_metadata ORDER BY
block_index")
        print("The Blockchain is:")
        blockchain_rows = cursor.fetchall()
        for row in blockchain_rows:
            print(row)
    elif choice == '4':
        print("Exiting...")
        connection.close()
        break
    else:
        print("Invalid choice. Please try again.")
```

#total lines of code: 678

Source Code–SQL

#Main Database:

```
create database login_data;
```

#Table 1: data_main (to save the users' credentials)

```
CREATE TABLE data_main (
   user_name VARCHAR(200) NOT NULL,
   user_age INT NOT NULL CHECK (user_age >= 18),
   user_aadhar VARCHAR(12) NOT NULL,
   user_state VARCHAR(50) NOT NULL,
   user_accountno VARCHAR(6) NOT NULL UNIQUE,
   user_pass VARCHAR(222) NOT NULL,
   VoteCn TINYINT NOT NULL CHECK (VoteCn <= 1 AND VoteCn >= 0));
```

#Adding a parameter to the column from data_main to keep a default vote count as '1', as the new user should have one vote by default.

```
USE login_data;
ALTER TABLE data_main
CHANGE VoteCn VoteCn TINYINT NULL DEFAULT 1 CHECK (VoteCn <=
1 AND VoteCn >= 0);
```

#Table 2: blockchain_metadata (to save the blockchain the database for non-tampering and security)

```
use login_data;
CREATE TABLE blockchain_metadata (
    block_index INT PRIMARY KEY,
    timestamp BIGINT NOT NULL,
    transaction_hash VARCHAR(64) NOT NULL,
    previous_hash VARCHAR(64) NOT NULL,
    block_hash VARCHAR(64) NOT NULL
    );
```

#Table 3: candidates (A table for the candidates)

```
use login_data;
create table candidates
(name_cand varchar(100) NOT NULL,
house_party varchar(200) NOT NULL,
votes varchar(2200) NULL);
```

#Table 4: mod_logs (A table to save the login credentials of the moderators)

```
USE login_data;
CREATE TABLE mod_logs (
    User_Id CHAR(10) NOT NULL UNIQUE,
    pass VARCHAR(222) NOT NULL
);
```

Output

```
>>>
Main Menu:
1. Voter
2. Moderator
3. Show Blockchain
4. Exit
Enter your choice: 1

Voter Menu:
1. Log In
2. Sign Up
3. Back
Enter your choice: 2
Enter your full name (as written on the aadhar card)
>Project Sample
Enter your age (must be more than or equal to 18)
>32
Enter your valid aadhar card number
>789456123216
Enter your state
>Maharashtra
Your username is....
```

> 852547

Enter your password (the password cannot be changed)

>sample@project

Voter Menu:

1. Log In

2. Sign Up

3. Back

Enter your choice: 1

Enter your user id (alphanumeric code provided at signing up)

>0000

Enter your password (case sensitive)

>asd

finding user....

Access Denied: Incorrect Username or Password

Voter Menu:

1. Log In

2. Sign Up

3. Back

Enter your choice: 1

Enter your user id (alphanumeric code provided at signing up)

>852547

Enter your password (case sensitive)

>sample@project

finding user....

Access Granted

Logged In as Voter:

1. Vote

2. Back

Enter your choice: 1

Please select the candidate number to vote for:

1. Joe Biden (Green House)

Enter your choice: 1

Blockchain Updated:

{'index': 8, 'timestamp': 1726750247, 'transactions_hash': '324944c173e70ae5125bfb225e550f7a91932f96df1525502c0e2f487f0895a2', 'previous_hash': '8ac5d93dd5bed5a05ae0b44e0e0fea5f22572d56d00901d64076c5d9eb12202a', 'hash': '9b22f27298c1cdb7c368c46a3ec63cbc8e670eb83c6d6f8945f7dac0b07d4fb3'}

Logged In as Voter:

1. Vote

2. Back

Enter your choice: 1

You have no votes left.

Logged In as Voter:

1. Vote

2. Back

Enter your choice: 2

Voter Menu:

1. Log In

2. Sign Up

3. Back

Enter your choice: 3

Main Menu:

1. Voter

2. Moderator

3. Show Blockchain

4. Exit

Enter your choice: 2

Moderator Menu:

1. Log In

2. Back

WARNING!: If you close the server you can't run other operations, you need to reopen the program..

Enter your choice: 1

Enter your user id

>2222

Enter your password

>aaas

finding mod....

Access Denied: Incorrect Username or Password

Moderator Menu:

1. Log In

2. Back

WARNING!: If you close the server you can't run other operations, you need to reopen the program..

Enter your choice: 1

Enter your user id

>5316920326

Enter your password

>2hxnc3xndk

finding mod....

Access Granted

Logged In as Moderator:

1. Validate Blockchain

2. Flag and Fix Invalid Block

3. Check Vote Tampering

4. Stop Server

5. Show Votes

6. Back

Enter your choice: 1

Blockchain is valid.

Logged In as Moderator:

1. Validate Blockchain
2. Flag and Fix Invalid Block
3. Check Vote Tampering
4. Stop Server
5. Show Votes
6. Back

Enter your choice: 2

Blockchain is valid.

No invalid blocks found.

Logged In as Moderator:

1. Validate Blockchain
2. Flag and Fix Invalid Block
3. Check Vote Tampering
4. Stop Server
5. Show Votes
6. Back

Enter your choice: 3

Vote tampering detected!

Vote snapshot updated.

Logged In as Moderator:

1. Validate Blockchain
2. Flag and Fix Invalid Block
3. Check Vote Tampering
4. Stop Server
5. Show Votes
6. Back

Enter your choice: 5

1. Joe Biden (Green House) - 5 votes

Logged In as Moderator:

1. Validate Blockchain
2. Flag and Fix Invalid Block
3. Check Vote Tampering
4. Stop Server
5. Show Votes
6. Back

Enter your choice: 4

Data image saved successfully.

Stopping server...

Server stopped and connection closed.

Logged In as Moderator:

1. Validate Blockchain
2. Flag and Fix Invalid Block

3. Check Vote Tampering

4. Stop Server

5. Show Votes

6. Back

Enter your choice: 6

Moderator Menu:

1. Log In

2. Back

WARNING!: If you close the server you can't run other operations, you need to reopen the program..

Enter your choice: 2

Main Menu:

1. Voter

2. Moderator

3. Show Blockchain

4. Exit

Enter your choice: 4

Exiting...

>>>

>>>

Main Menu:

1. Voter

2. Moderator

3. Show Blockchain

4. Exit

Enter your choice: 3

The Blockchain is:

(0, 1725631701,
'89eb0ac031a63d2421cd05a2fbe41f3ea35f5c3712ca839cbf6b85c4e
e07b7a3', '0',
'de717c7183a18bfa452a31b507cb13a52018f715cf9a8cfe162f590693
93a842')

(1, 1725631909,
'04d16fcc728f7c20b84b34f19a44393c5ef69d8167163f65d4b2f877c6
a2b2c2',
'de717c7183a18bfa452a31b507cb13a52018f715cf9a8cfe162f590693
93a842',
'b4bbb5842438725fda9c5b21b02dfc2872f947190aa43831f70999890
74bc992')

(2, 1725631980,
'04d16fcc728f7c20b84b34f19a44393c5ef69d8167163f65d4b2f877c6
a2b2c2',
'b4bbb5842438725fda9c5b21b02dfc2872f947190aa43831f70999890
74bc992',
'6791be01e1329ab51549d3034d6eaee3204196b3a2076dabc4cab92
68da4af70')

(3, 1725632079,
'202cf5a0e54cdafbd0ce2f2b14b94a5708d6d2a2c701017f10a340bccc
12af2a',
'6791be01e1329ab51549d3034d6eaee3204196b3a2076dabc4cab92
68da4af70',
'5bd458bb3966e158bd87849d5458b7a70bbad547f08a44c1184e05c
9e6fd92fa')

(4, 1725632191,
'c24e9f5da0d04b3d5a9caf5e36c54e5f3d7381571ec1f10ce2dcab32a

992752e',
'5bd458bb3966e158bd87849d5458b7a70bbad547f08a44c1184e05c9e6fd92fa',
'3d1f821a3f5fc0ed745d5635bfbb500a30f371f29d3bb1b986f5cfd02912c8b1')

(5, 1725632245,
'8fc9ff9bc87df7b5d1521cb94c0869b86f779625a1479d6fe144d90820d6b1dd',
'3d1f821a3f5fc0ed745d5635bfbb500a30f371f29d3bb1b986f5cfd02912c8b1',
'cd474d08f88c20692c3c61e7fd9bd7f51e4ae9e68abe2974c99402191ea4c9c0')

(6, 1725632544,
'606333613cc3097d3bc52106a0fb0931221fa8cf9367789159ff99962fcabbaa',
'cd474d08f88c20692c3c61e7fd9bd7f51e4ae9e68abe2974c99402191ea4c9c0',
'2b04cc35565bdc06a642201f05318b920d1dc96967857fffe2b3d8a00447fad6')

(7, 1725632624,
'606333613cc3097d3bc52106a0fb0931221fa8cf9367789159ff99962fcabbaa',
'2b04cc35565bdc06a642201f05318b920d1dc96967857fffe2b3d8a00447fad6',
'8ac5d93dd5bed5a05ae0b44e0e0fea5f22572d56d00901d64076c5d9eb12202a')

(8, 1726750247,
'324944c173e70ae5125bfb225e550f7a91932f96df1525502c0e2f487f0895a2',
'8ac5d93dd5bed5a05ae0b44e0e0fea5f22572d56d00901d64076c5d9eb12202a',
'9b22f27298c1cdb7c368c46a3ec63cbc8e670eb83c6d6f8945f7dac0b07d4fb3')

Main Menu:

1. Voter

2. Moderator

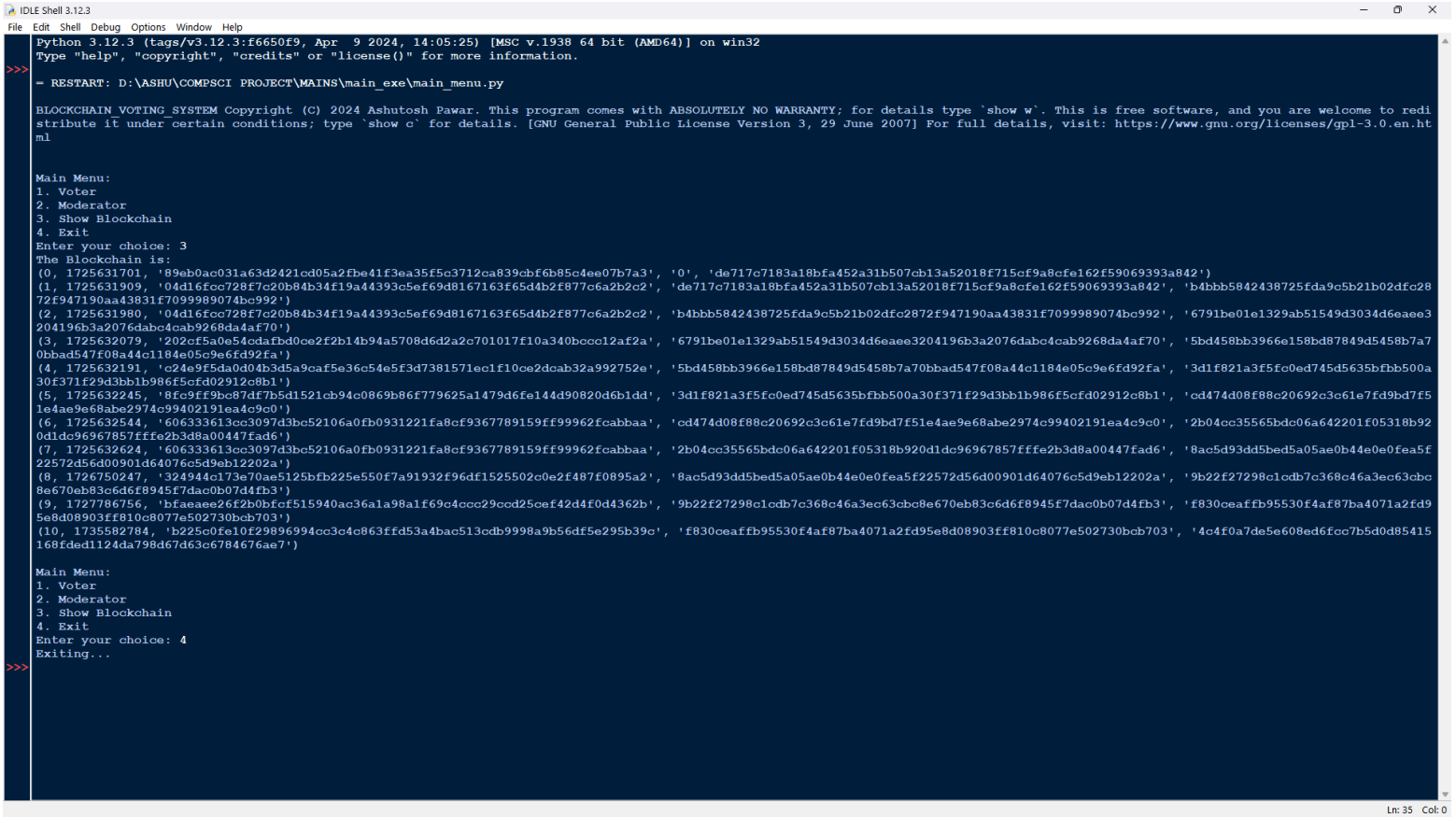
3. Show Blockchain

4. Exit

Enter your choice: 4

Exiting...

>>>



README.md File

BLOCKCHAIN_VOTING_SYSTEM

This is a Blockchain-based Voting System designed to ensure secure, tamper-proof, and transparent elections.

Features:

1. **Voter Module**
   - Login and Sign-up functionality for voters.
   - Cast votes securely using blockchain technology.

2. **Moderator Module**
   - Login functionality for moderators.
   - Validate the integrity of the blockchain.
   - Flag and fix invalid blocks.
   - Monitor for vote tampering and discrepancies.
   - View voting results and manage the system effectively.

3. **Blockchain Integration**
   - Each vote is recorded as a block on the blockchain.
   - Blocks are validated and protected against tampering.
   - Transparent storage of election data.

4. **Database Connectivity**

   - Secure connection to a MySQL database for managing user data and blockchain metadata.

Technical Details:

- **Language:** Python

- **Database:** MySQL

- **License:** GNU General Public License v3.0

Usage:

1. Clone the repository or download the project files.

2. Set up the MySQL database using the provided schema.

3. Run the main Python script to start the system.

4. Choose from the main menu to access voter or moderator functionalities.

Author:

Ashutosh Pawar

2024

License

This project is licensed under the GNU General Public License v3.0. See the LICENSE file for details.

The project is distributed WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. For full license details, visit:

<https://www.gnu.org/licenses/gpl-3.0.en.html>

Application Package Files

blockchain_main.py

main menu.py

main_libs_mods.py

main_libs_voters.py

moderator_tools.py

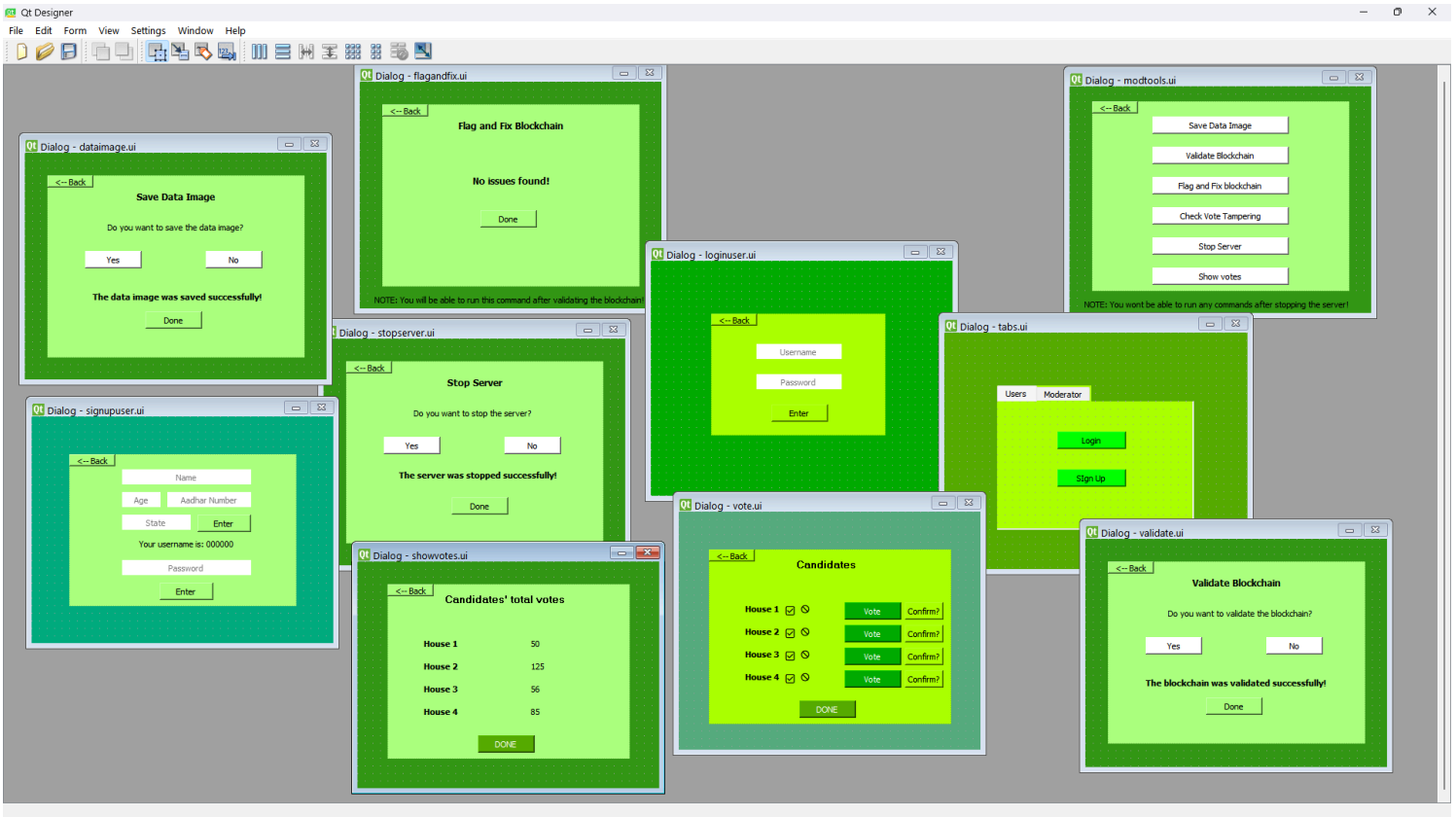
README.md

vote_recorder.py

vote_snapshot.dat

Scope of Improvement

The Blockchain Voting System can be further enhanced by incorporating a modern graphical user interface (GUI) to improve accessibility and usability for voters and moderators. A well-designed GUI would make the system more intuitive, reduce reliance on terminal-based interaction, and provide a visually appealing experience. Features like real-time vote progress tracking, blockchain visualization, and user-friendly navigation could be added. The provided UI images made in QT Designer serve as inspiration and could represent the second version of this code, elevating its functionality and appeal.



Conclusion

The Blockchain Voting System revolutionizes traditional voting by ensuring security, transparency, and tamper-proof election processes. By integrating blockchain technology, it safeguards voter data and guarantees the integrity of every vote. The system's modular design supports secure voter authentication, efficient vote recording, and blockchain validation by moderators. With its robust architecture and user-focused features, this project demonstrates the potential of blockchain in critical applications. Future improvements could include developing a user-friendly graphical interface based on provided UI designs.

Bibliography

1. **Python Official Documentation**
    - https://www.python.org/doc/
2. **Stack Overflow**
    - https://stackoverflow.com/
3. **GNU General Public License (GPL-3.0)**
    - https://www.gnu.org/licenses/gpl-3.0.en.html
4. **MySQL Official Documentation**
    - https://dev.mysql.com/doc/
5. **Blockchain Introduction**
    - https://www.ibm.com/topics/blockchain
6. **Blockchain for Developers**
    - https://www.blockchain.com/learning-portal
7. **Textbook of Computer Science with Python Class XII by VK Pandey**