



EVOTING SYSTEM USING BLOCK CHAIN & FACE RECOGNITION CSE3002 – INTERNET & WEB PROGRAMMING

J COMPONENT PROJECT DOCUMENT

FALL 2019 - 2020

INDIVIDUAL PROJECT

1. 17BCE2044 SIDHANTHA PODDAR

UNDER THE GUIDENCE OF

PROF. NAVEEN KUMAR N

SCOPE



School of Computer Science and Engineering

DECLARATION

I/We hereby declare that the J Component project entitled **“Title of the project”** submitted by me/us to the School of Computer Science and Engineering, VIT University, Vellore-14 in partial fulfillment of the requirements for the **Internet and Web Programming (CSE3002)** course, is a record of bonafide work carried out by me/us under the supervision of **Naveen Kumar N, Assistant Professor (Sr)**. I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma of this institute or of any other institute or university.

Reg.No: 17BCE2044

Name: SIDHANTHA PODDAR

Chapter Title	Page
Title Page	i
Declaration	ii
Table of Contents	iii
List of Tables	iv
List of Figures	v
List of Abbreviations	vi
Abstract	vii

1 TABLE OF CONTENTS

2	Introduction	5
2.1	AIM.....	5
2.2	Objective	5
3	System Requirements	5
3.1	block chain model	5
3.2	database for voting and vote login webpage	6
3.3	main webpage and landing page	6
4	Solution for existing problem	7
4.1	Data – Vote integrity.....	7
4.1.1	Problem description.....	7
4.1.2	Sample code.....	7
4.1.3	Screenshot	8
4.2	Security	9
4.2.1	Problem description.....	9
4.2.2	Sample code.....	9
4.2.3	Screenshot	9
4.3	Recovery.....	10
4.3.1	Problem description.....	10
4.3.2	Sample code.....	10
4.3.3	Screenshot	10
4.4	Information access.....	11
4.4.1	Problem description.....	11

4.4.2	Sample code.....	11
4.4.3	Screenshot	11
4.5	Privacy.....	13
4.5.1	Problem description.....	13
4.5.2	Sample code.....	13
4.5.3	Screenshot	13
5	Recent technologies.....	14
5.1	Blockchain	14
5.1.1	Importance of the technology	14
5.1.2	Sample code(1/10 th).....	14
5.1.3	Sample screenshot.....	15
5.2	Face Recognition.....	17
5.2.1	Importance of the technology	17
5.2.2	Sample code.....	17
5.2.3	Sample screenshot.....	18
5.3	Flask	19
5.3.1	Importance of the technology	19
5.3.2	Sample code.....	19
5.3.3	Sample screenshot.....	20
6	Screenshot	21
7	Conclusion.....	22

2 INTRODUCTION

2.1 AIM

Building an electronic voting system that satisfies the legal requirements of legislators has been a challenge for a long time. Distributed ledger technologies is an exciting technological advancement in the information technology world. Blockchain technologies offer an infinite range of applications benefiting from sharing economies. This project aims to evaluate the application of blockchain as service to implement distributed electronic voting systems. This project elucidates the requirements of building electronic voting systems and identifies the legal and technological limitations of using blockchain as a service for realizing such systems.

This project also includes some security features and other services like face recognition which decreased the chance of potential attack and makes online voting more secure and reliable.

2.2 OBJECTIVE

This objective of this project is to

- Create an E-voting system using blockchain.
- Make a secure portal for login using face recognition.
- Create an information website where admin can add information.
- Add user profile.
- Update user profile.
- Password recovery using email

3 SYSTEM REQUIREMENTS

3.1 BLOCK CHAIN MODEL

- Ethereum Wallet
- Meta-Mask
- Gnache
- Truffle – framework
- Solidity
- Node-js packages
- Web Browser which supports Metamask

3.2 DATABASE FOR VOTING AND VOTE LOGIN WEBPAGE

- PHP
- MYSQL
- XAMPP
- JQuery

3.3 MAIN WEBPAGE AND LANDING PAGE

- Pickle
- OS module
- Flask
- Flask_sql_alchemy
- OpenCV
- Numpy
- Pandas
- Har cascade
- MYSQLI
- Other flask dependencies

4 SOLUTION FOR EXISTING PROBLEM

4.1 DATA – VOTE INTEGRITY

4.1.1 Problem description

In today's world data integrity is one of the major issues. Vote being a very sensitive data is vulnerable to attack thus we need to ensure that it is not manipulated, and its integrity is maintained. It can be solved using new technologies like block chain

4.1.2 Sample code

```
var Election = artifacts.require("./Election.sol");

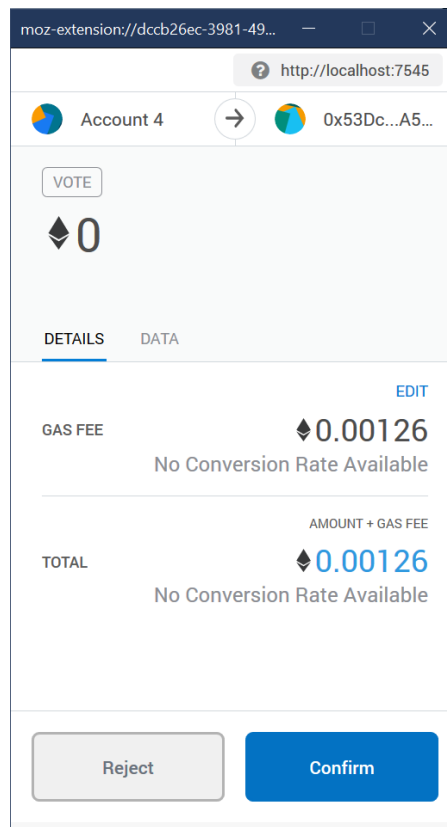
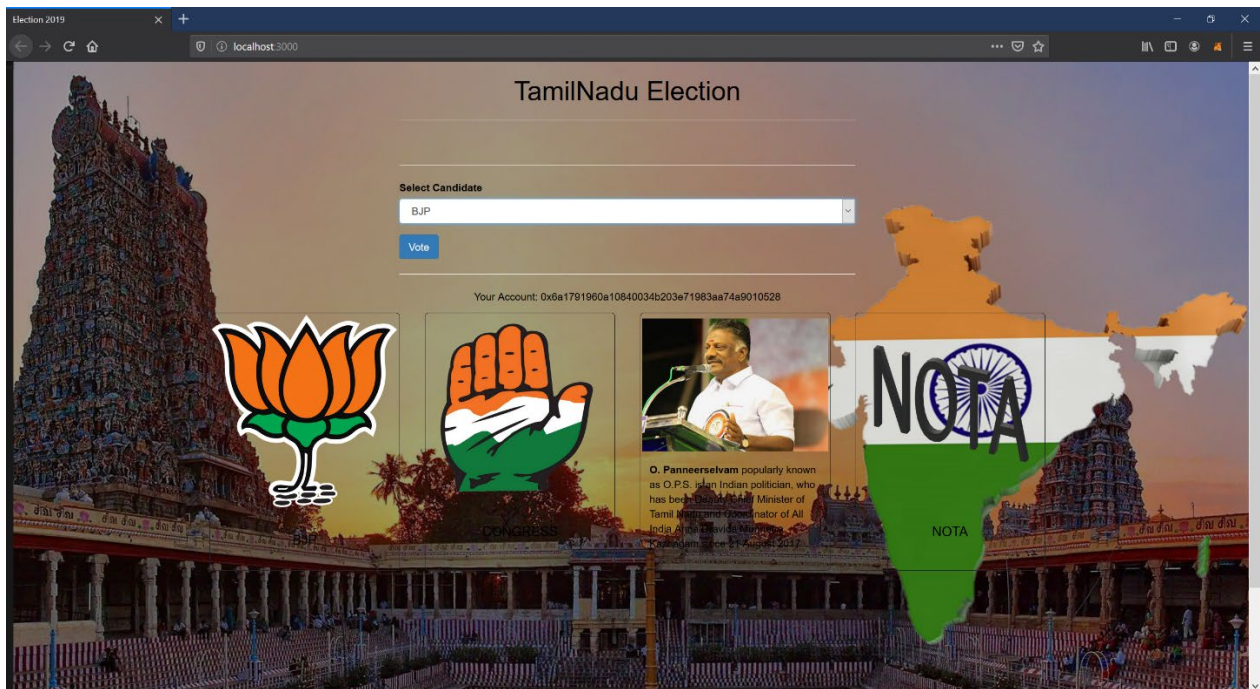
contract("Election", function(accounts){

    var electionInstance;

    it("initializes with 3 candidates", function(){
        return
        Election.deployed().then(function(instance){
            return
            instance.candidatesCount();
        }).then(function(count){

            assert.equal(count,4);
        });
    });
    it("it initializes the candidates with the correct
values", function() {
        return Election.deployed().then(function(instance) {
            electionInstance = instance;
            return electionInstance.candidates(1);
        }).then(function(candidate) {
            assert.equal(candidate[0], 1, "contains the correct id");
            assert.equal(candidate[1], "BJP",
"contains the correct name");
            assert.equal(candidate[2], 0, "contains the correct votes
count");
            return electionInstance.candidates(2);
        }).then(function(candidate) {
            assert.equal(candidate[0], 2, "contains the correct id");
            assert.equal(candidate[1], "Congress", "contains the
correct name");
            assert.equal(candidate[2], 0, "contains the correct votes
count");
            return electionInstance.candidates(3);
        }).then(function(candidate) {
            assert.equal(candidate[0], 3, "contains the correct id");
            assert.equal(candidate[1], "AAP", "contains the correct
name");
            assert.equal(candidate[2], 0, "contains the correct votes
count");
            return electionInstance.candidates(4);
        }).then(function(candidate) {
            assert.equal(candidate[0], 4, "contains the correct id");
            assert.equal(candidate[1], "NOTA", "contains the correct
name");
            assert.equal(candidate[2], 0, "contains the correct votes
count");
        });
    });
});
```

4.1.3 Screenshot



4.2 SECURITY

4.2.1 Problem description

In today's world alphanumeric passwords are not enough as due to raw computing power and advanced technology passwords can be brute forced and phishing is a major issue. To tackle this problem, we have used biometrics in form of face recognition to allow voting

4.2.2 Sample code

```
class RegistrationForm(FlaskForm):
    username = StringField('Username',
        validators=[DataRequired(), Length(min=2,
max=20)])
    email = StringField('Email',
        validators=[DataRequired(), Email()])
    password = PasswordField('Password',
        validators=[DataRequired()])
    confirm_password = PasswordField('Confirm Password',
        validators=[DataRequired(),
            EqualTo('password')])
    submit = SubmitField('Sign Up')



    def validate_username(self, username):
        user =
            User.query.filter_by(username=username.data).first()

        if user:
            raise ValidationError('That username is taken. Please
            choose a different one.')

    def validate_email(self, email):
        user = User.query.filter_by(email=email.data).first()
        if user:
            raise ValidationError('That email is taken. Please
            choose a different one.')

class LoginForm(FlaskForm):
    email = StringField('Email',
        validators=[DataRequired(), Email()])
    password = PasswordField('Password',
        validators=[DataRequired()])
    remember = BooleanField('Remember Me')
    submit = SubmitField('Login')
```

4.2.3 Screenshot

 naveenkumar.n@vit.ac.in	01-11-2019 14:31	File folder
 sidhanthapoddar99@gmail.com	30-10-2019 03:05	File folder



4.3 RECOVERY

4.3.1 Problem description

Recover of lost data is one of the major problems thus forgot password option is added to the project for safe recovery of password

4.3.2 Sample code

```
class RequestResetForm(FlaskForm):
    email = StringField('Email',
                        validators=[DataRequired(), Email()])
    submit = SubmitField('Request Password Reset')

    def validate_email(self, email):
        user = User.query.filter_by(email=email.data).first()
        if user is None:
            raise ValidationError('There is no account with that
            email. You must register first.')

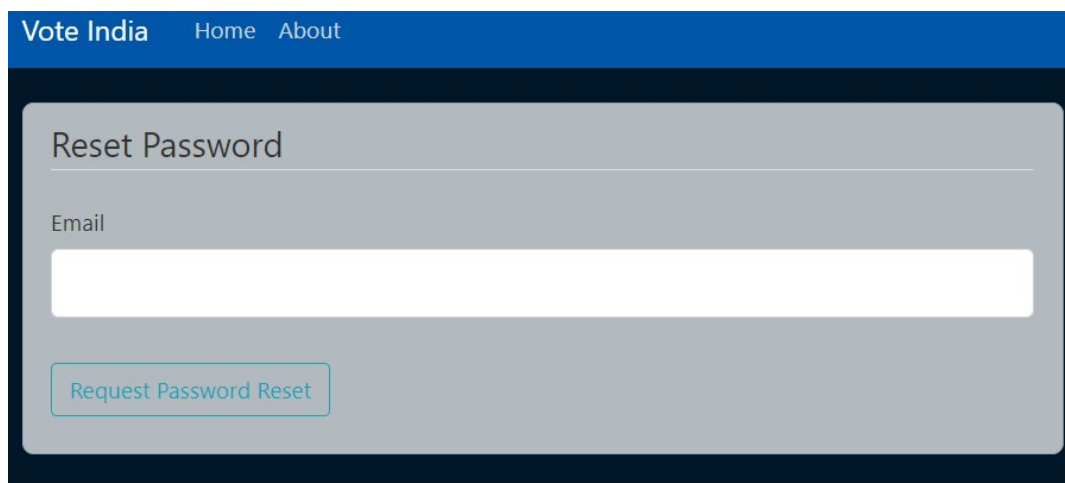
class ResetPasswordForm(FlaskForm):
    password = PasswordField('Password',
    validators=[DataRequired()])
    confirm_password = PasswordField('Confirm Password',
    validators=[DataRequired(),
    EqualTo('password')])
    submit = SubmitField('Reset Password')

def send_reset_email(user):
    token = user.get_reset_token()
    msg = Message('Password Reset Request',
        sender='noreply@demo.com',
        recipients=[user.email])
    msg.body = f'''To reset your password, visit the following
    link:
    {url_for('reset_token', token=token, _external=True)}

    If you did not make this request then simply ignore this email
    and no changes will be made.
    '''
    mail.send(msg)

@app.route("/reset_password", methods=['GET', 'POST'])
def reset_request():
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    form = RequestResetForm()
    if form.validate_on_submit():
        user = User.query.filter_by(email=form.email.data).first()
        send_reset_email(user)
        flash('An email has been sent with instructions to reset
        your password.', 'info')
        return redirect(url_for('login'))
    return render_template('reset_request.html', title='Reset
    Password', form=form)
```

4.3.3 Screenshot



The screenshot shows a web application interface for a 'Reset Password' form. At the top, there is a blue header bar with the text 'Vote India' and two links: 'Home' and 'About'. Below the header, the main content area has a light gray background. The title 'Reset Password' is displayed in a dark font. Underneath the title, there is a label 'Email' followed by a large, empty white input field. At the bottom of the form, there is a button with a blue border and the text 'Request Password Reset'.

4.4 INFORMATION ACCESS

4.4.1 Problem description

In any voting platform data like data and current news feeds are a very important part of the procedure which guides the users through the portal and helps the selecting the right candidate for voting. Thus, mini blogs and new blogs are used to spread information.

4.4.2 Sample code

```
@app.route("/user/<string:username>")
def user_posts(username):
    page = request.args.get('page', 1, type=int)
    user =
    User.query.filter_by(username=username).first_or_404()
    posts = Post.query.filter_by(author=user)\
        .order_by(Post.date_posted.desc())\
        .paginate(page=page, per_page=5)
    return render_template('user_posts.html', posts=posts,
user=user)

@app.route("/post/<int:post_id>/delete", methods=['POST'])
@login_required
def delete_post(post_id):
    post = Post.query.get_or_404(post_id)
    if post.author != current_user:
        abort(403)
    db.session.delete(post)
    db.session.commit()
    flash('Your post has been deleted!', 'success')
    return redirect(url_for('home'))

@app.route("/post/<int:post_id>/update",
methods=['GET', 'POST'])
@login_required
def update_post(post_id):
    post = Post.query.get_or_404(post_id)
    if post.author != current_user:
        abort(403)
    form = PostForm()
    if form.validate_on_submit():
        post.title = form.title.data
        post.content = form.content.data
        db.session.commit()
        flash('Your post has been updated!', 'success')
        return redirect(url_for('post', post_id=post.id))
    elif request.method == 'GET':
        form.title.data = post.title
        form.content.data = post.content
    return render_template('create_post.html', title='Update
Post',
form=form, legend='Update Post')
```

4.4.3 Screenshot



The screenshot shows a web application interface for 'Vote India'. At the top, there is a blue navigation bar with the text 'Vote India' and links for 'Home' and 'About'. Below the navigation bar, the main content area has a dark blue background. A light gray box titled 'New Post' is centered on the page. Inside this box, there is a 'Title' label followed by a white text input field. Below the title field is a 'Content' label followed by a larger white text area with a small double-slash icon at the bottom right. At the bottom left of the 'New Post' box, there is a button labeled 'Post' with a teal border.

[Sidhantha Poddar](#) 2019-10-29

Update

Delete

Legislative assembly elections

Assembly elections of Andhra Pradesh, Arunachal Pradesh, Odisha, Sikkim, were held simultaneously with the general elections

Legislative assembly elections of Haryana, Maharashtra held on 21st October 2019.

Legislative assembly elections of Jharkhand will be held later this year.

Posts by Sidhantha Poddar (4)

[Sidhantha Poddar](#) 2019-11-01

voting starts now

start today

[Sidhantha Poddar](#) 2019-10-29

Link for future election dates

https://en.m.wikipedia.org/wiki/2019_elections_in_India

[Sidhantha Poddar](#) 2019-10-29

Legislative assembly elections

Assembly elections of Andhra Pradesh, Arunachal Pradesh, Odisha, Sikkim, were held simultaneously with the general elections

Legislative assembly elections of Haryana, Maharashtra held on 21st October 2019.

Legislative assembly elections of Jharkhand will be held later this year.

[Sidhantha Poddar](#) 2019-10-29

General elections

General elections were held in India in April to May 2019 to constitute the 17th Lok Sabha. After the fierce fight in the election, the results were declared on 23 May, early exit polls suggest a win for the BJP led NDA.

The phase-wise schedule, the number of seats in each phase and their State-wise break-up:

4.5 PRIVACY

4.5.1 Problem description

Voting is the fundamental right of all citizens of India which is directly provided by the constitution. Vote should be private and must not be linked with the voter to avoid vote manipulation. Thus, to solve this problem different databases are formed one to maintain a vote count and other containing the people count. Thus, this ensures privacy of votes

4.5.2 Sample code

```
def get_rows():
    em=current_user.email
    connection =
    pymysql.connect(host="127.0.0.1",user="root",passwd="",
    database="voter" )
    cursor = connection.cursor()
    retrieve = "Select * from vote where ID='"+em+"';"
    #executing the queries
    cursor.execute(retrieve)
    rows = cursor.fetchall()
    connection.commit()
    connection.close()
    return rows

def insert_into_db(ID,KEYCODE):
    connection =
    pymysql.connect(host="127.0.0.1",user="root",passwd="",
    database="voter" )
    cursor = connection.cursor()
    insert1 = "INSERT INTO vote
VALUES('"+ID+"','"+KEYCODE+"',1);"
    cursor.execute(insert1)
    connection.commit()
    connection.close()

def get_code():
    em=current_user.email
    rows=get_rows()
    if(len(rows)==0):

KEYCODE=''.join([random.choice("12345667890ABCDEF12
345667890GHIJKLMNOPQRST12345667890") for i in
range(20) ])
    insert_into_db(em,KEYCODE)

    return KEYCODE

def gen(code="Not Generated"):
    #code="123123"
    return render_template('cgen.html',
    title='Account',code=code)

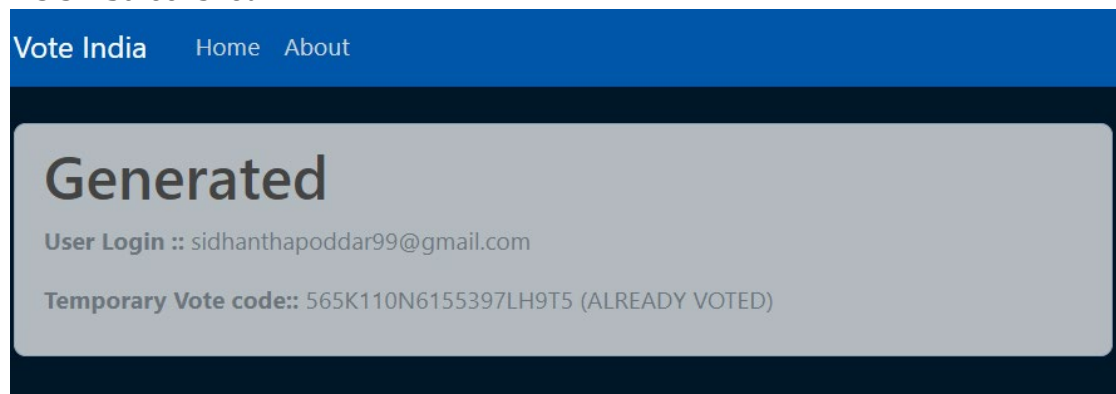
@app.route("/gene",methods=["POST","GET"])
@login_required
def gene():

    if len(get_rows())!=0:
        row=get_rows()[0]
        code=row[1]
        voted=row[2]
        if voted==1:
            code+=' (CODE GENERATED PREVIOUSLY)'
        else:
            code+=' (ALREADY VOTED)'
        return gen(code)

    form = GenForm()
    if form.validate_on_submit():
        if form.etype.data=="I ACCEPT" :
            next_page = request.args.get('next')
            code=get_code()
            return gen(code)
            #return redirect(next_page) if next_page else
            redirect(url_for('home'))
        else:
            flash('Enter the correct phase', 'danger')
            return render_template('generate.html', title='Login',
            form=form)

1.1.1 Screenshot
```

4.5.3 Screenshot



5 RECENT TECHNOLOGIES

5.1 BLOCKCHAIN

5.1.1 Importance of the technology

Traditional databases are maintained by a single organization, and that organization has complete control of the database, including the ability to tamper with the stored data, to censor otherwise valid changes to the data, or to add data fraudulently. For most use cases, this is not a problem since the organization which maintains the database does so for its own benefit, and therefore has no motive to falsify the database's contents; however, there are other use cases, such as a financial network, where the data being stored is too sensitive and the motive to manipulate it is too enticing to allow any single organization to have total control over the database. Even if it could be guaranteed that the responsible organization would never enact a fraudulent change to the database (an assumption which, for many people, is already too much to ask), there is still the possibility that a hacker could break in and manipulate the database to their own ends.

The most obvious way to ensure that no single entity can manipulate the database is to make the database public, and allow anyone to store a redundant copy of the database. In this way, everyone can be assured that their copy of the database is intact, simply by comparing it with everyone else's. This is sufficient as long as the database is static; however, if changes must be made to the database after it has been distributed, a problem of consensus arises: which of the entities keeping a copy of the database decides which changes are allowed and what order those changes occurred in? If any of the entities can make changes at any time, the redundant copies of the database will quickly get out of sync, and there will be no consensus as to which copy is correct. If all of the entities agree on a certain one who makes changes first, and the others all copy from it, then that one has the power to censor changes it doesn't like. Furthermore, if that one entity disappears, the database is stuck until all of the others can organize to choose a replacement. All of the entities may agree to take turns making changes and all the others copy changes from the one whose turn it is, but this opens the question of who decides who gets a turn when.

5.1.2 Sample code(1/10th)

```
App = {
  web3Provider: null,
  contracts: {},
  account: '0x0',
  hasVoted: false,

  init: function() {
    return App.initWeb3();
  },

  initWeb3: function() {
    // TODO: refactor conditional
    if (typeof web3 !== 'undefined') {
      // If a web3 instance is already provided by Meta Mask.
      App.web3Provider = web3.currentProvider;
      web3 = new Web3(web3.currentProvider);
    } else {
      // Specify default instance if no web3 instance provided
      App.web3Provider = new
Web3.providers.HttpProvider('http://localhost:7545');
      web3 = new Web3(App.web3Provider);
    }

    App.listenForEvents();

    return App.render();
  }
};

// Listen for events emitted from the contract
listenForEvents: function() {
  App.contracts.Election.deployed().then(function(instance) {
    // Restart Chrome if you are unable to receive this event
    // This is a known issue with Metamask
    // https://github.com/MetaMask/metamask-extension/issues/2393
    instance.votedEvent({}, {
      fromBlock: 0,
      toBlock: 'latest'
    }).watch(function(error, event) {
      console.log("event triggered", event)
      // Reload when a new vote is recorded
      App.render();
    });
  });
}
```

```

return App.initContract();
},

initContract: function() {
$.getJSON("Election.json", function(election) {
// Instantiate a new truffle contract from the artefact
App.contracts.Election = TruffleContract(election);
// Connect provider to interact with contract
App.contracts.Election.setProvider(App.web3Provider);

```

```

render: function() {
var electionInstance;
var loader = $("#loader");
var content = $("#content");

loader.show();
content.hide();

```

5.1.3 Sample screenshot

Ganache

ACCOUNTS BLOCKS TRANSACTIONS CONTRACTS EVENTS LOGS UPDATE AVAILABLE SEARCH FOR BLOCK NUMBERS OR TX HASH

CURRENT BLOCK 0 GAS PRICE 20000000000 GAS LIMIT 6721975 HARDFORK PETERSBURG NETWORK ID 5777 RPC SERVER HTTP://127.0.0.1:7545 MINING STATUS AUTOMINING WORKSPACE QUICKSTART SAVE SWITCH

MNEMONIC layer phone virus avocado dinner festival arrange increase frog word bring caution **HD PATH** m/44'/60'/0'/0/account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0x23d898ECA9e9D0B51C7AD42AcF82FF83DC7cebEF	100.00 ETH	0	0
0x75C29Ebf31DB5814BBaC25471D715EF97bf97de7	100.00 ETH	0	1
0x6a1791960A10840034b203e71983AA74A9010528	100.00 ETH	0	2
0x0C854e521d1764B7954c3E3bDCd1CF53117A0070	100.00 ETH	0	3
0x52049281A4931e1Ef0c195243cBDb12F1D7559Ed	100.00 ETH	0	4
0xB7483Fe4AC12b5e1Be94d22882AF705ccaCC3766	100.00 ETH	0	5
0x953bD659Bcc6E5Ae81819d2323955775DC492b32	100.00 ETH	0	6

5.2 FACE RECOGNITION

5.2.1 Importance of the technology

As we see that existing voting system has many defects such as lengthy process, time taking, not secure, bogus voting, no security level but now we can say that our approach is more useful and secure from the existing system. Since, we are using three level of security in this proposed system the false voters can be easily identified. The facial authentication technique is very much useful in identifying the fraud voters, so we can avoid the bogus votes during election commission. The voters can cast their voting from anywhere by login to our proposed smart voting system through internet. As every operation is performed through internet connectivity so, it is one-time investment for government. Voters' location is not important, but their voting is important. As data is stored in centralized repository so, data is accessible at any time as well as backup of the data is possible. Smart voting system provides updated result at each minute. Also requires less manpower and resources.

5.2.2 Sample code

```
def getface(valtbc):
    #print("1")
    face_cascade =
cv2.CascadeClassifier(pathss('face\\HCTrainingImages\\haarcascade_frontalface
_default.xml'))

    recognizer = cv2.face.LBPHFaceRecognizer_create()
    recognizer.read(pathss('face\\training.yml'))
    iddict = pickle.load(open(pathss('face\\pickle\\iddict.pkl'),'rb'))
    font = cv2.FONT_HERSHEY_SIMPLEX
    cap = cv2.VideoCapture(0)
    start_time = time.asctime(time.localtime(time.time()))

    att_stud = {}
    init_faculty = {}
    facenum = 0
    arrn=[]
    #print("2")
    while facenum<50:

        _, img = cap.read()
        grey = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(grey, 1.3, 5)

        for (x,y,w,h) in faces:
            roi_grey = grey[y:y+h, x:x+w]
            pred_id, config = recognizer.predict(roi_grey)

            facenum+=1

            for regno,[id,desig] in iddict.items():
                if id == pred_id:
                    pred_regno = regno
                    arrn+=[pred_regno]
                    #print(pred_regno)
                    colour_desig = tuple([255 if desig == 'student' else 0, 255 if desig ==
'faculty' else 0,255 if desig == 'admin' else 0])
                    cv2.putText(img, str(pred_regno), (x,y), font , 1, colour_desig, 2)
                    #cv2.putText(img, str(desig), (x,y+h), font , 1, colour_desig, 2)
                    #cv2.rectangle(img, (x,y), (x+w, y+h), (255,0,0), 2)
                    cv2.circle(img, (int(x+w/2),int(y+h/2)), int(h/2), (0,255,0), 2)

                    """"if regno not in att_stud and desig == 'student':
                        att_stud[regno] = [desig,
time.asctime(time.localtime(time.time()))]

                    if regno not in init_faculty and desig == 'faculty':

def register():

    if current_user.is_authenticated:

        return redirect(url_for('home'))

    form = RegistrationForm()

    if form.validate_on_submit():

        hashed_password =
bcrypt.generate_password_hash(form.password.data).d
ecode('utf-8')

        user = User(username=form.username.data,
email=form.email.data, password=hashed_password)

        newUser(form.email.data)

        train()

        db.session.add(user)

        db.session.commit()

        flash('Your account has been created! You are now
able to log in', 'success')

        return redirect(url_for('login'))

    return render_template('register.html', title='Register',
form=form)

@app.route("/login", methods=['GET', 'POST'])

def login():

    print("0")

    if current_user.is_authenticated:

        return redirect(url_for('home'))

    form = LoginForm()

    print("00")
```

```

        init_faculty[regno] = [desig,
time.asctime(time.localtime(time.time()))]"""

    cv2.imshow('img', img)
    k = cv2.waitKey(30) & 0xFF
    if k==27:
        break

    cap.release()
    cv2.destroyAllWindows()
    end_time = time.asctime(time.localtime(time.time()))

    all_att = []
    if os.path.exists('face\\pickle\\all_att.pkl'):
        all_att = pickle.load(open(pathss('face\\pickle\\all_att.pkl'), 'rb'))

    all_att.append([att_stud, init_faculty, start_time, end_time])
    pickle.dump(all_att,open(pathss('face\\pickle\\all_att.pkl'),'wb'))
    #print("3")
    return arrn.count(valtbc)

```

```

    if form.validate_on_submit():

        user =
        User.query.filter_by(email=form.email.data).first()

        if getface(form.email.data)>40 and user and
        bcrypt.check_password_hash(user.password,
        form.password.data):

            login_user(user, remember=form.remember.data)

            next_page = request.args.get('next')

            return redirect(next_page) if next_page else
            redirect(url_for('home'))

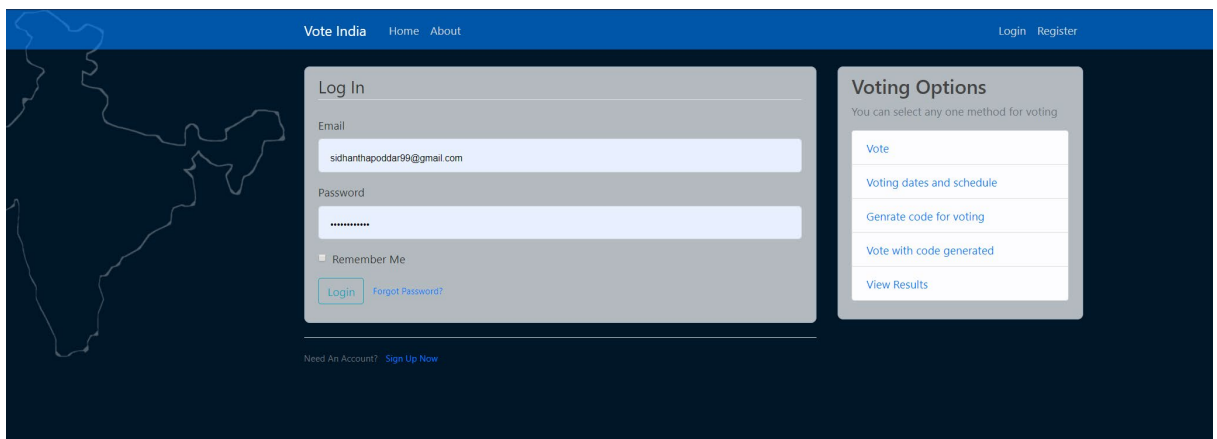
        else:

            flash('Login Unsuccessful. Please check email and
            password', 'danger')

            return render_template('login.html', title='Login',
            form=form)

```

5.2.3 Sample screenshot



5.3 FLASK

5.3.1 Importance of the technology

Flask comes with all the benefits of fast templates, strong WSGI features, thorough unit testability at the web application and library level, extensive documentation. So next time you are starting a new project where you need some good features and a vast number of extensions

5.3.2 Sample code

```
def account():
    form = UpdateAccountForm()
    if form.validate_on_submit():
        if form.picture.data:
            picture_file = save_picture(form.picture.data)
            current_user.image_file = picture_file
        current_user.username = form.username.data
        current_user.email = form.email.data
        db.session.commit()
        flash('Your account has been updated!', 'success')
        return redirect(url_for('account'))
    elif request.method == 'GET':
        form.username.data = current_user.username
        form.email.data = current_user.email
        image_file = url_for('static', filename='profile_pics/' +
                             current_user.image_file)
        return render_template('account.html', title='Account',
                               image_file=image_file, form=form)

@app.route("/post/new", methods=['GET', 'POST'])
@login_required
def new_post():
    form = PostForm()
    if form.validate_on_submit():
        post = Post(title=form.title.data,
                    content=form.content.data, author=current_user)
        db.session.add(post)
        db.session.commit()
        flash('Your post has been created!', 'success')
        return redirect(url_for('home'))
    return render_template('create_post.html', title='New
Post',
                           form=form, legend='New Post')

app.route("/post/<int:post_id>/update", methods=['GET',
'POST'])
@login_required
def update_post(post_id):
    post = Post.query.get_or_404(post_id)
    if post.author != current_user:
        abort(403)
    form = PostForm()
    if form.validate_on_submit():
        post.title = form.title.data
        post.content = form.content.data
        db.session.commit()
        flash('Your post has been updated!', 'success')
        return redirect(url_for('post', post_id=post.id))
    elif request.method == 'GET':
        form.title.data = post.title
        form.content.data = post.content
    return render_template('create_post.html', title='Update
Post',
                           form=form, legend='Update Post')

@app.route("/post/<int:post_id>/delete",
methods=['POST'])
@login_required
def delete_post(post_id):
    post = Post.query.get_or_404(post_id)
    if post.author != current_user:
        abort(403)
    db.session.delete(post)
    db.session.commit()
    flash('Your post has been deleted!', 'success')
    return redirect(url_for('home'))
```

5.3.3 Sample screenshot

```
Anaconda Prompt - python run.py

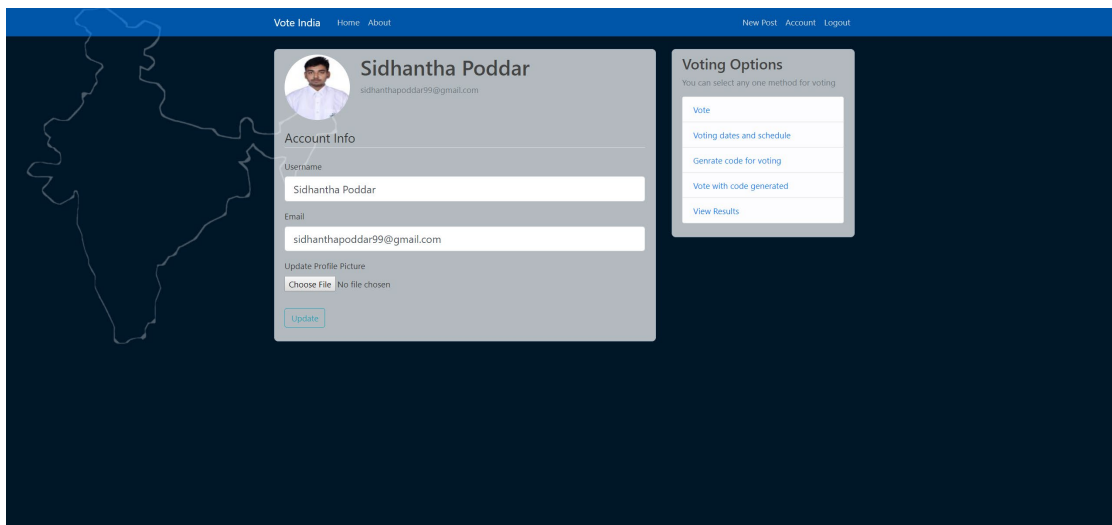
(base) C:\Users\sidha>activate flask

(flask) C:\Users\sidha>cd C:\Users\sidha\Desktop\iwp PROJ\TEST-AREA\vote-blog
The filename, directory name, or volume label syntax is incorrect.

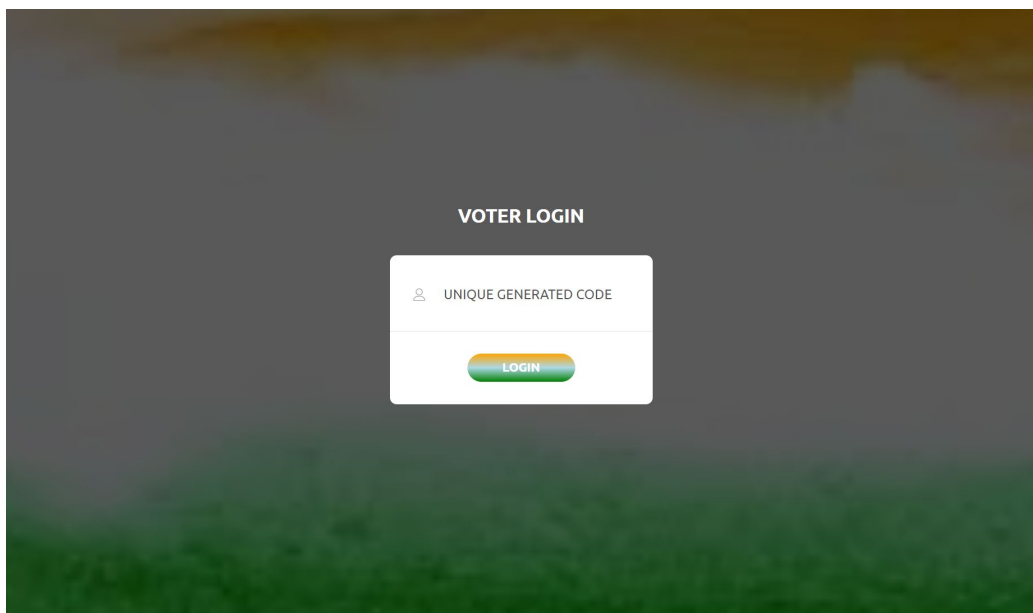
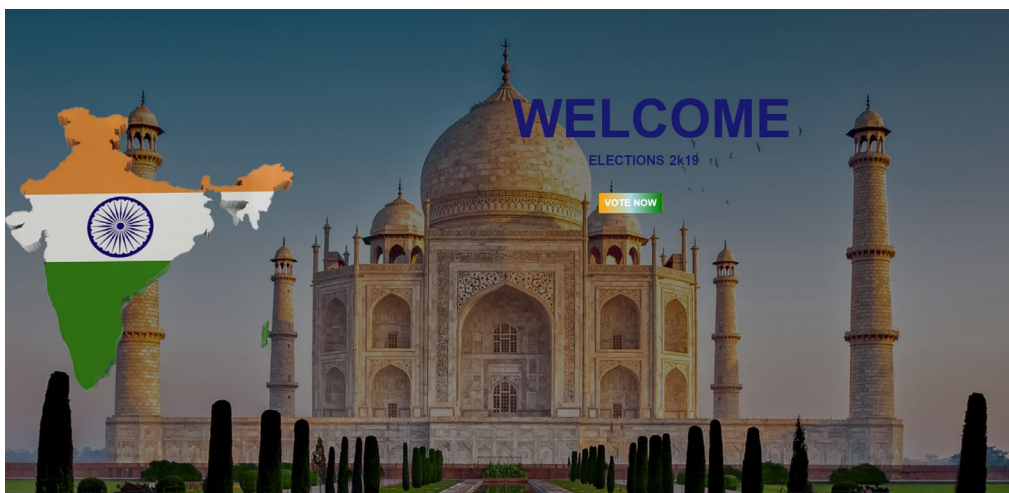
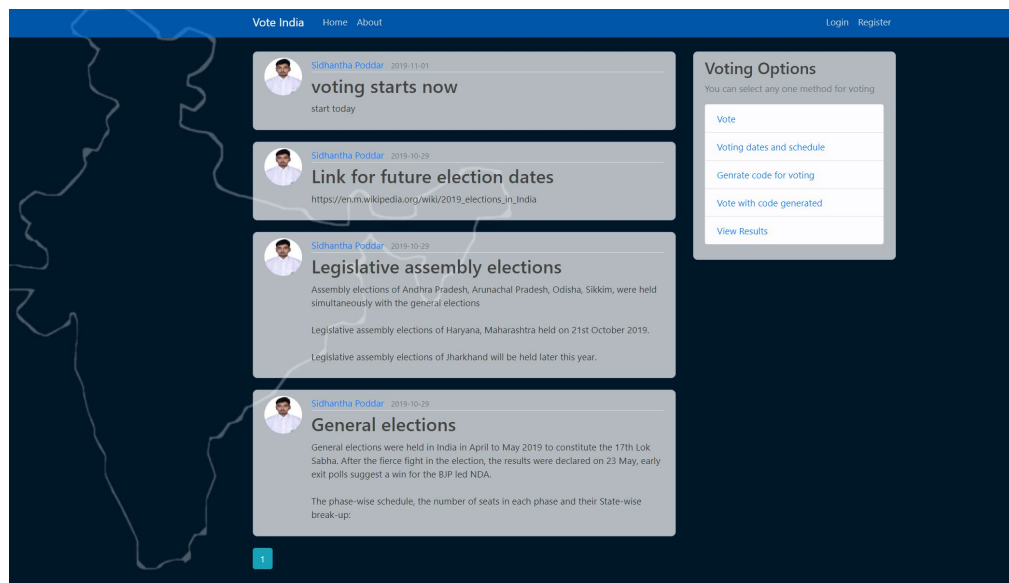
(flask) C:\Users\sidha>cd C:\Users\sidha\Desktop\iwp PROJ\TEST-AREA\vote-blog

(flask) C:\Users\sidha\Desktop\iwp PROJ\TEST-AREA\vote-blog>python reun.py
python: can't open file 'reun.py': [Errno 2] No such file or directory

(flask) C:\Users\sidha\Desktop\iwp PROJ\TEST-AREA\vote-blog>python run.py
C:\Users\sidha\conda\envs\flask\lib\site-packages\flask_sqlalchemy\__init__.py:835: FSADeprecationWarning: SQLAlchemy_TRACK_MODIFICATIONS adds significant overhead and will be disabled by default in the future. Set it to True or False to suppress this warning.
  'SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and '
* Serving Flask app "flaskblog" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
C:\Users\sidha\conda\envs\flask\lib\site-packages\flask_sqlalchemy\__init__.py:835: FSADeprecationWarning: SQLAlchemy_TRACK_MODIFICATIONS adds significant overhead and will be disabled by default in the future. Set it to True or False to suppress this warning.
  'SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and '
* Debugger is active!
* Debugger PIN: 271-488-838
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```



6 SCREENSHOT



7 CONCLUSION

In its most basic form, blockchain is a digital ledger. The technology draws its power from the peers – or nodes – on its network to verify, process, and record all transactions across the system. This ledger is never stored, but rather exists on the “chain” supported by millions of nodes simultaneously. Thanks to encryption and decentralization, blockchain’s database of transactions is incorruptible, and each record is easily verifiable. The network cannot be taken down or influenced by a single party because it doesn’t exist in one place.

It’s not only financial transactions that work with blockchain, but any type of data transmission. This kind of system infrastructure is extremely useful for voting because a vote is a small piece of high-value data. Out of necessity, modern voting systems are largely stuck in the last century, and those that want to vote must leave their homes and submit paper ballots to a local authority. Some have tried, but it has proven difficult to put faith in the results due to large gaps in security.

Blockchain can solve the many problems discovered in these early attempts at online voting. A blockchain-based voting application does not concern itself with the security of its internet connection, because any hacker with access to the terminal will not be able to affect other nodes. Voters can effectively submit their vote without revealing their identity or political preferences to the public. Officials can count votes with absolute certainty, knowing that each ID can be attributed to one vote, no fakes can be created, and that tampering is impossible.