A DMSL Mini Project Report

On

MANAGEMENT SYSTEM OF DONATION FOR CASUALTY SOLDIERS

By

Karanjkar Soham Sachin (20121004.)
Patil Abhijeet Todarmal (20121023.)
Suralkar Omkar Ravindra (20121031.)
Katare Dhananjay Sachin (20121045.)

Under the guidance of

Mr. Kishor B Sadafale

Department of Computer Engineering

Government College of Engineering and Research, Awasari

SAVITRIBAI PHULE PUNE UNIVERSITY 2022-2023

Government College of Engineering and Research, Awasari

Department of Computer Engineering

Date:

CERTIFICATE

This is to certify that,

Karanjkar Soham Sachin (20121004.)
Patil Abhijeet Todarmal (20121023.)
Suralkar Omkar Ravindra (20121031.)
Katare Dhananjay Sachin (20121045.)

of class T.E Computer Engineering have successfully completed their project work on "MANAGEMENT SYSTEM OF DONATION FOR CASUALTY SOLDIERS" at GOVERNMENT COLLEGE OF ENGINEERING AND RESEARCH, AWASARI in the partial fulfillment of the Graduate Degree course in T.E (Computer Engineering) 2019 Course, in the academic Year 2022-2023 Semester – I as prescribed by the Savitribai Phule Pune University.

Mr. Kishor B Sadafale
Guide

Dr. S. U. Ghumbre
Head of the Department

Acknowledgement

I feel great pleasure in expressing my deepest sense of gratitude and sincere thanks to my guide **Prof. K.B. Sadafale** for their valuable guidance during the Project work, without which it would have been very difficult task. I have no words to express my sincere thanks for valuable guidance, extreme assistance and cooperation extended to all the **Staff Members** of my Department.

This acknowledgement would be incomplete without expressing my special thanks to **Prof. Dr. S. U. Ghumbre** Head of the Department (Computer Engineering) for their support during the work.

I would also like to extend my heartfelt gratitude to my **Principal, Dr. D. R. Pangavane** who provided a lot of valuable support, mostly being behind the veils of college bureaucracy.

Last but not least I would like to thanks all the Teaching, Non- Teaching staff members of my Department, my parent and my colleagues those who helped me directly or indirectly for completing of this Project successfully.

Karanjkar Soham Sachin
Patil Abhijeet Todarmal
Suralkar Omkar Ravindra
Katare Dhananjay Sachin

,

Contents

1. **TITLE OF THE PROJECT**
2. **ABSTRACT**
3. **INTRODUCTION**
    - Problem definition
4. **SCOPE**
5. **SPECIFIC REQUIRMENTS**
    - Hardware Interface
    - Software Interface
6. **THEORY OF SOFTWERE USED**
    - PYTHON FLASK
    - MYSQL
7. **ER DIAGRAM**
8. **DATABASE DESIGN**
9. **OUTPUT SCREEN (GUI)**
10. **SAMPLE CODE**
11. **CONCLUSION**

12. **REFERENCES**

1. Management System of Donation for Casualty Soldiers

2.

ABSTRACT

This Project is all about donation to the Families of Soilders who lost there life in war In database we have given the numbers of families of Soilders so that when user will login to portal he/she will see the names of Soilders and there family members name and UPI number

On website We have created 3 Sections as follows as

1. Army
2. Navy
3. AirForce

This Section will Show the name of Soilders And there Payment details to there family Respectively User will donate to them respectively we have created 2 Databases for the same i.e Soilder and User the Soilder Database will Store Details of Soilders in that table and User Database will Store details of User.

3. INTRODUCTION

3.1 Problem Definition

Need for system:

To support the concern and sentiments of spirited citizens like you Indian Army is operating two accounts viz Army Central Welfare Fund and Army Battle Casualties Welfare Fund, which accepts donation /contribution, for serving and retired soldiers and their families as under:-

(a) **Army Central Welfare Fund**. The contribution received in this fund are utilised to pay financial assistance/ grant to widows of our Soldiers, their Next of Kin, Dependents and needy ExServicemen as a welfare measure. The donations made to this fund are 100% exempted from Income Tax Act vide clause (III hc) of section 80-G of Income Tax Act 1961.

(b) **Army Battle Casualties Welfare Fund**. The contribution received in the fund are utilised to pay financial assistance/ grant to widows of our Battle Casualties, their next of kins and dependents.

4. SCOPE

1. Familiy of Martyr soldier will get an financial support from this project.

2. Country will know the brave strories of the so many unknown or untold stories of soldiers.

3. Making public aware that soldiers family life is not easy is they have a martyr in their house.

4. Upcoming generation will get an inspiration by reading their stories from this project.

5. Students get information about how we get freedom from the britishors.

5. SPECIFIC REQUIRMENTS

The system analysis contains a planning and design phases where a logical design of system is developed and to work accordingly a plan is established. Also the requirements of system are identified and the operating environment is identified.

5.1 Hardware Requirements

o Windows 10 & Windows 7
Operating System.

o 1 GB RAM

o Intel®core2duo processor@3.4GHz

o 200MB memory Space

5.2SoftwareRequirements

oMySQL

oPython Flask

6. THEORY OF SOFTWARE USED

MySQL

MySQL is the world's most popular open source database. According to DB-Engine, MySQL ranks as the second-most-popular database, behind Oracale Database. MySQL powers many of the most accessed applications, including Facebook, Twitter, Netflix, Uber, Airbnb, Shopify, and Booking.com.

Since MySQL is open source, it includes numerous features developed in close cooperation with users over more than 25 years. So it's very likely that your favorite application or programming language is supported by MySQL Database.

Ease of use: Developers can install MySQL in minutes, and the database is easy to manage.

Reliability: MySQL is one of the most mature and widely used databases. It has been tested in a wide variety of scenarios for more than 25 years, including by many of the world's largest companies. Organizations depend on MySQL to run business-critical applications because of its reliability.

Scalability: MySQL scales to meet the demands of the most accessed applications. MySQL's native replication architecture enables organizations such as Facebook to scale applications to support billions of users.

Performance: MySQL HeatWave is faster and less expensive than other database services, as demonstrated by multiple standard industry benchmarks, including TPC-H, TPC-DS, and CH-benCHmark.

High availability: MySQL delivers a complete set of native, fully integrated replication technologies for high availability and disaster recovery. For business-critical applications, and to meet service-level agreement commitments, customers can achieve

- Recovery point objective = 0 (zero data loss)
- Recovery time objective = seconds (automatic failover)

Security: Data security entails protection and compliance with industry and government regulations, including the European Union General Data Protection Regulation, the Payment Card Industry Data Security Standard, the Health Insurance Portability and Accountability Act, and the Defense Information Systems Agency's Security Technical Implementation Guides. MySQL Enterprise Edition provides advanced security features, including authentication/authorization, transparent data encryption, auditing, data masking, and a database firewall.

Flexibility: The MySQL Document Store gives users maximum flexibility in developing traditional SQL and NoSQL schema-free database applications. Developers can mix and match relational data and JSON documents in the same database and application.

CRUD Operations in SQL-

As we know, CRUD operations act as the foundation of any computer programming language or technology. So before taking a deeper dive into any programming language or technology, one must be proficient in working on its CRUD operations. This same rule applies to databases as well.

Let us start with the understanding of CRUD operations in SQL with the help of examples. We will be writing all the queries in the supporting examples using the MySQL database.

1.Create-

In CRUD operations, 'C' is an acronym for create, which *means to add or insert data into the SQL table.* So, firstly we will create a table using CREATE command and then we will use the INSERT INTO command to insert rows in the created table**.**

2.Read:

In CRUD operations, 'R' is an acronym for read, which means *retrieving or fetching the data from the SQL table.* So, we will use the SELECT command to fetch the inserted records from the SQL table. We can retrieve all the records from a table using an asterisk (*) in a SELECT query. There is also an option of retrieving only those records which satisfy a particular condition by using the WHERE clause in a SELECT query.

3.Update:

In CRUD operations, 'U' is an acronym for the update, which *means making updates to the records present in the SQL tables.* So, we will use the UPDATE command to make changes in the data present in tables.

4.Delete:

In CRUD operations, 'D' is an acronym for delete, which means *removing or deleting the records from the SQL tables.* We can delete all the rows from the SQL tables using the DELETE query. There is also an option to remove only the specific records that satisfy a particular condition by using the WHERE clause in a DELETE query.

Python Flask-

Flask is a microWeb FrameWork written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre- existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for Object Relational Mapper, form validation, upload handling, various open authentication technologies and several common framework related tools.

Flask History-

Flask was created by Armin Ronacher of Pocoo, an international group of Python enthusiasts formed in 2004. According to Ronacher, the idea was originally an Aprils Fool's joke that was popular enough to make into a serious application.The name is a play on the earlier Bottle framework.

When Ronacher and Georg Brandl created a bulletin board system written in Python in 2004, the Pocoo projects Werkzeug and jinja were developed.

In April 2016, the Pocoo team was disbanded and development of Flask and related libraries passed to the newly formed Pallets project.Since 2018, Flask-related data and objects can be rendered with BootStrap

Flask has become popular among Python enthusiasts. As of October 2020, it has second most

stars on Github among Python web-development frameworks, only slightly behind Django and was voted the most popular web framework in the Python Developers Survey 2018, 2019, 2020 and 2021

Components of Python Flask

- Werkzeug

Werkzeug (German for "tool") is a utility library for the Python programming language for Web Server Gateway Interface (WSGI) applications. Werkzeug can instantiate objects for request, response, and utility functions. It can be used as the basis for a
custom software framework and supports Python 2.7 and 3.5 and later. Jinja

Jinja, also by Ronacher, is a template engine for the Python programming language. Similar to the Django web framework, it handles templates in a sandbox.

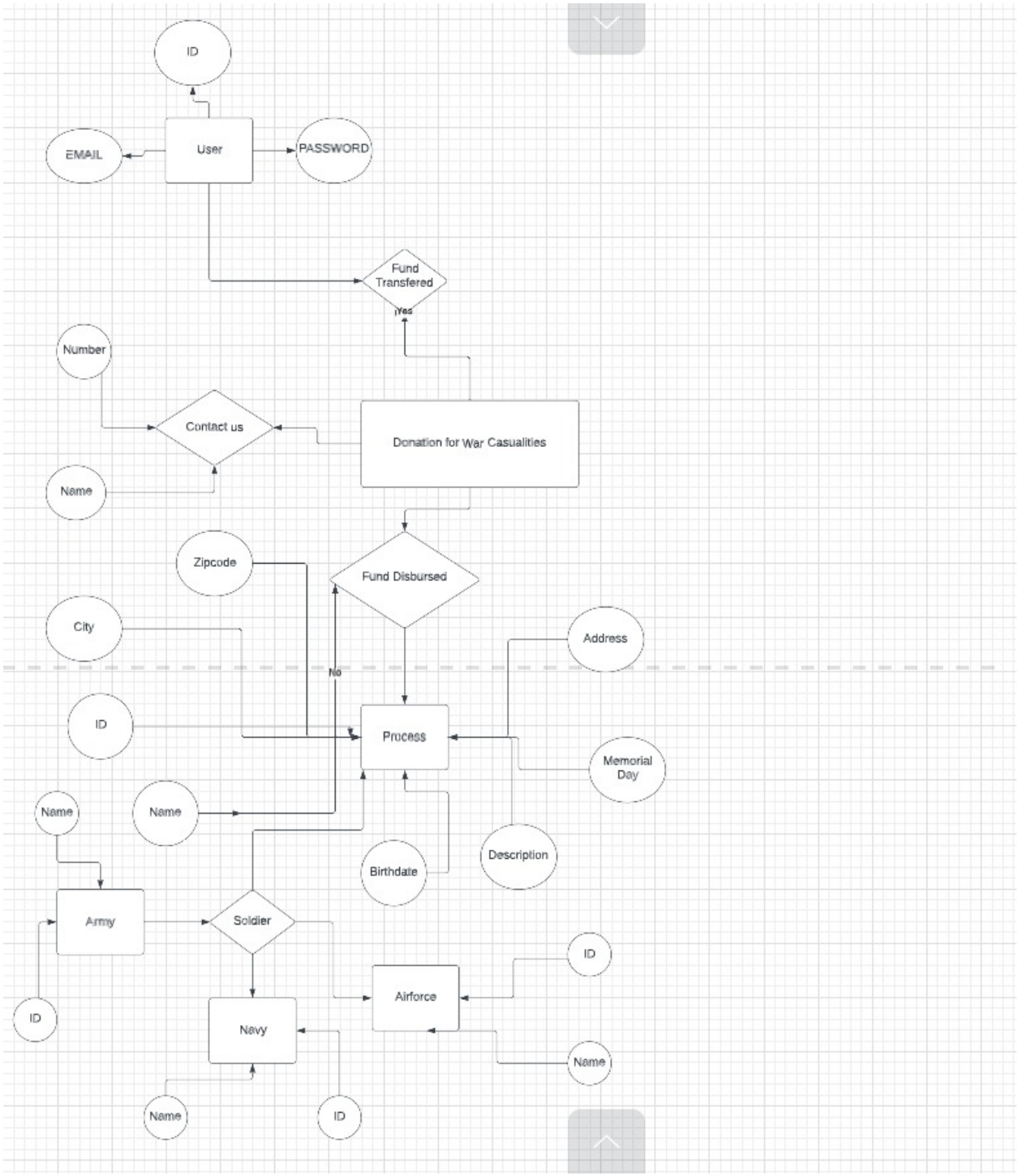
MarkupSafe

MarkupSafe is a string handling library for the Python programming language. The eponymous MarkupSafe type extends the Python string type and marks its contents as "safe"; combining MarkupSafe with regular strings automatically escapes the unmarked strings, while avoiding double escaping of already marked strings.

ItsDangerous

ItsDangerous is a safe data serialization library for the Python programming language. It is used to store the session of a Flask application in a cookie without allowing users to tamper with the session contents.

7. ER DIAGRAM


ID
User
EMAIL
PASSWORD
Fund Transfered
Yes
Number
Contact us
Donation for War Casualities
Name
Zipcode
Fund Disbursed
City
Address
No
ID
Process
Memorial Day
Name
Name
Birthdate
Description
Army
Soldier
ID
Airforce
ID
Navy
Name
Name
ID

8.DATABASE DESIGN

```
from flask import Flask,render_template,request,session,redirect
from flask_sqlalchemy import SQLAlchemy # This is to connect Database to front-end
from flask_login import UserMixin,login_user,logout_user,login_manager
from werkzeug.security import generate_password_hash, check_password_hash
from flask_login import login_required , current_user
from flask_login.login_manager import LoginManager
from flask import url_for
from flask import Flask, send_from_directory
flag = 0
local_server = True
app = Flask(__name__)
app.secret_key = 'dbmsproject'
#  For images
# @app.route('/<path:path>', methods=['GET'])
# def static_proxy(path):
#   return send_from_directory('./images', path)

# picFolder = os.path.join('templates','images')
# app.config(['UPLOAD_FOLDER']) = picFolder

app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql://root:@localhost/army'
db=SQLAlchemy(app)
```

```
login_manager = LoginManager(app)
login_manager.login_view = 'signin'


class Signup(UserMixin,db.Model):
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(30))
    password = db.Column(db.String(100))

class Soldier(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100))
    birthdate = db.Column(db.String(100))
    memorialday = db.Column(db.String(100))
    decription = db.Column(db.String(100))
    types = db.Column(db.String(100))
    address = db.Column(db.String(100))
    city = db.Column(db.String(30))
    zipcode = db.Column(db.Integer)




@login_manager.user_loader
def load_user(signup_id):
    return Signup.query.get(int(signup_id))


@app.route("/")
def index():
    return render_template('index.html')
```

```
# This id for soldier


@app.route("/addsoldier" , methods=['POST','GET'])
def addsoldier():
    if request.method=="POST":
        flag = 2
        name=request.form.get('name')
        email=request.form.get('email')
        birthdate = request.form.get('birthdate')
        memorialday = request.form.get('memorialday')
        type = request.form.get('type')
        desc = request.form.get('desc')
        address = request.form.get('address')
        city = request.form.get('city')
        zipcode = request.form.get('zipcode')
            new_user = db.engine.execute(f"INSERT INTO `soldier` ( `name`, `email`,`birthdate`, `memorialday`, `description`, `type`, `addresss`, `city`, `zipcode`) VALUES ( '{name}','{email}', '{birthdate}', '{memorialday}', '{desc}', '{type}', '{address}', '{city}', '{zipcode}');")


        print("This is  Post Buddy")



    if request.method=="GET":
        flag = 1
        print("This is get method")


    return render_template('addsoldier.html')


@app.route("/test")
def test():
    try:
```

```
        Notes.query.all()
        return 'Success'
    except:
        return 'Fail to connect'



@app.route("/contactus")
def contactus():
    return render_template('contactus.html')



@app.route("/signup" , methods=['POST','GET'])
def signup():
    if request.method=="POST":
        email=request.form.get('email')
        password=request.form.get('password')
        print("email =" , email)
        print("Password = " , password)

        user=Signup.query.filter_by(email=email).first()
        if user:
            print("Email already existed")
            return render_template('/signup.html')


        else:
            encpassord = generate_password_hash(password)
            print("Stm 1")
              # new_user = db.engine.execute(f" INSERT INTO 'user' ('username','email','password') VALUES ('{username}' , '{email}' , '{encpassord}');")
                    new_user = db.engine.execute(f"INSERT INTO `signup` (`email`, `password`) VALUES ('{email}', '{password}');")
            print("Stm 2")
```

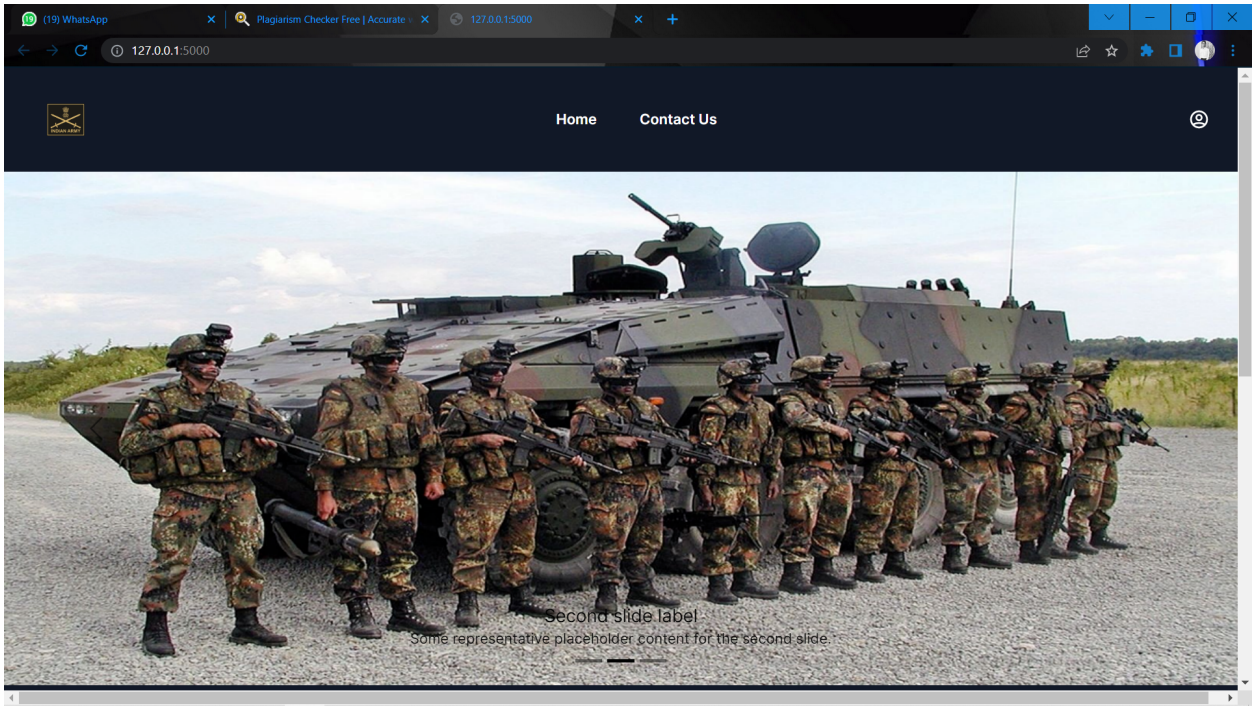
```
            return render_template('signin.html')
    # print("Method = ")
    # print(request.method)


    return render_template('signup.html')


@app.route("/new_index",methods=['POST','GET'])
@login_required
def new_index():
    # print("Catagory ",catagory)
    if request.method=="POST":
        print(" This is POST")
        return render_template('new_index.html')
    else:
        print(" This is GET")

        return render_template('new_index.html')


@app.route("/logout")
@login_required
def logout():
    flag = 0
    logout_user()
    return redirect(url_for('signin'))



@app.route("/showarmy")
def showarmy():
    query = db.engine.execute("SELECT * FROM `soldier` WHERE type='army'")
```

```
    return render_template('showarmy.html', query=query)

@app.route("/shownavy")
def shownavy():
    query = db.engine.execute("SELECT * FROM `soldier` WHERE type='navy'")
    return render_template('shownavy.html', query=query)

@app.route("/showairforce")
def showairforce():
    query = db.engine.execute("SELECT * FROM `soldier` WHERE type='airforce'")
    return render_template('showairforce.html', query=query)



@app.route("/signin", methods=['POST','GET'])
def signin():
    if request.method=="POST":
        email=request.form.get('email')
        password=request.form.get('password')
        user = Signup.query.filter_by(email=email).first()
        if user and user.password==password:
            print(" You can proceed :) ")
            login_user(user)
            return redirect(url_for('new_index'))
        else:
            print("Invalid details")
            return render_template('signin.html')

        print(email , password)
        return render_template('signin.html')
    return render_template('signin.html')

    return render_template('signin.html')
```

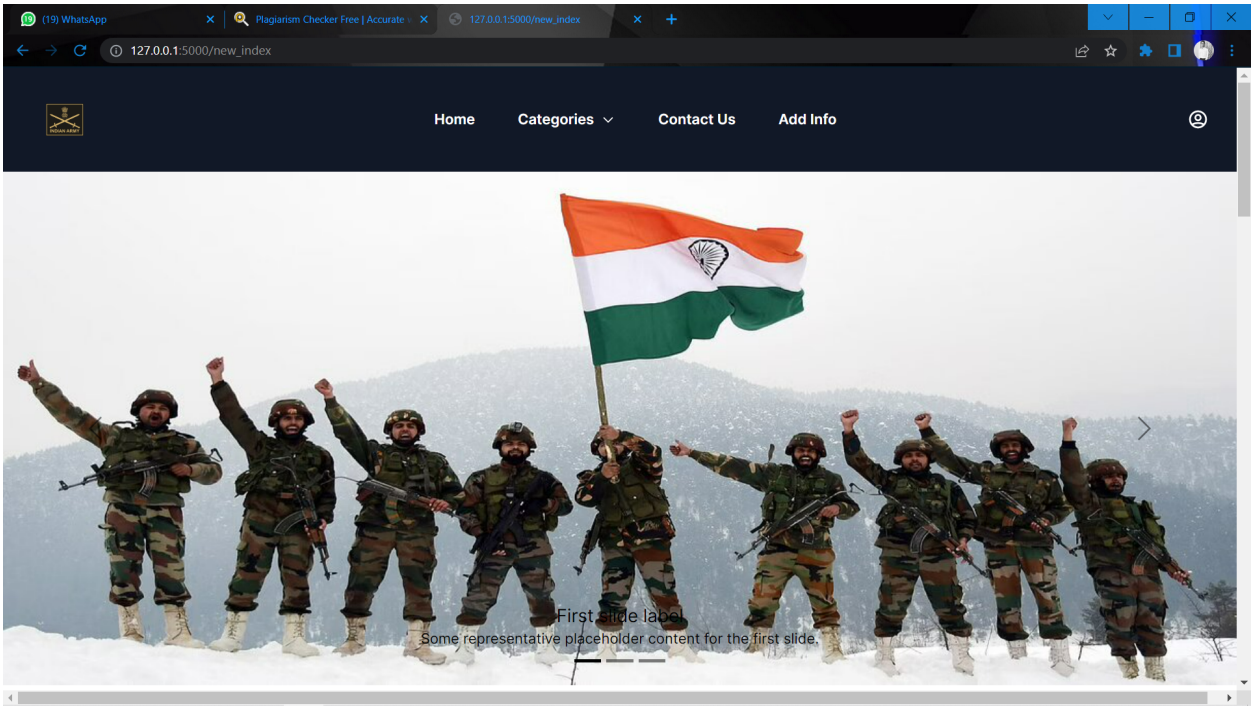
```
# @app.route("/signup")
# def signup():
#     return render_template('signup.html')
print("flag = ",flag)

app.run(debug=True , threaded=True)
```

9. OUTPUT SCREEN (GUI)

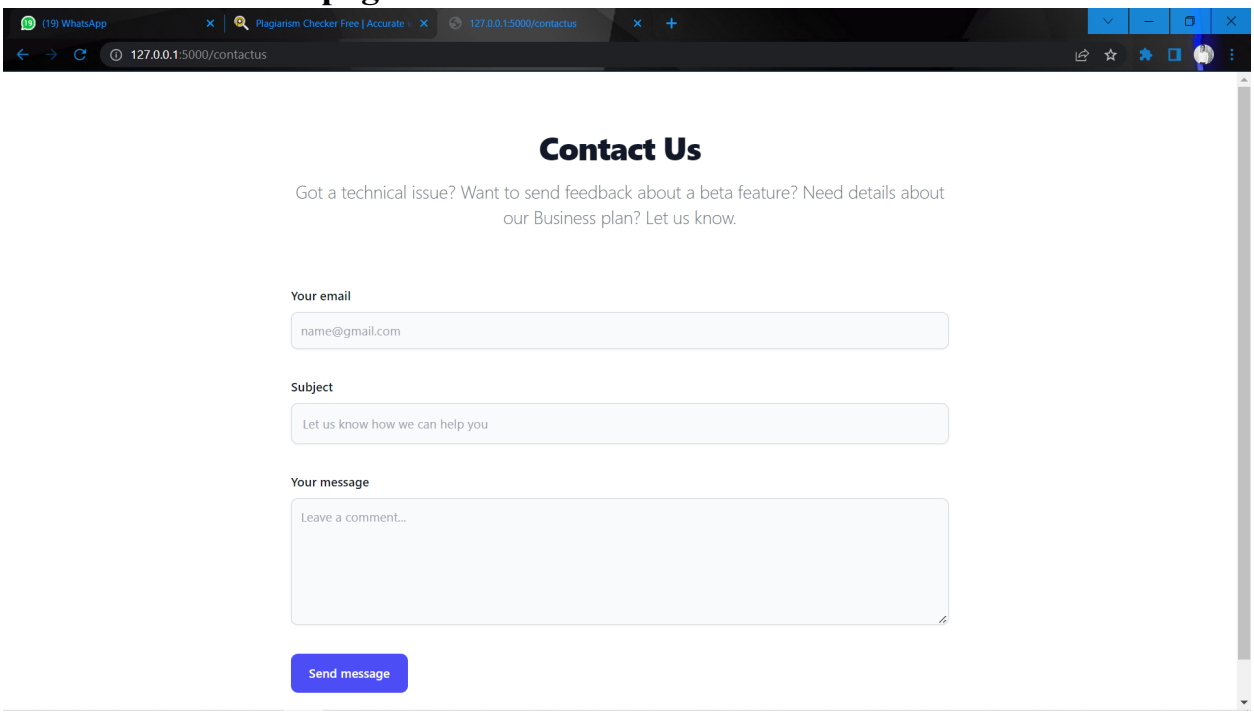
1.Home Page



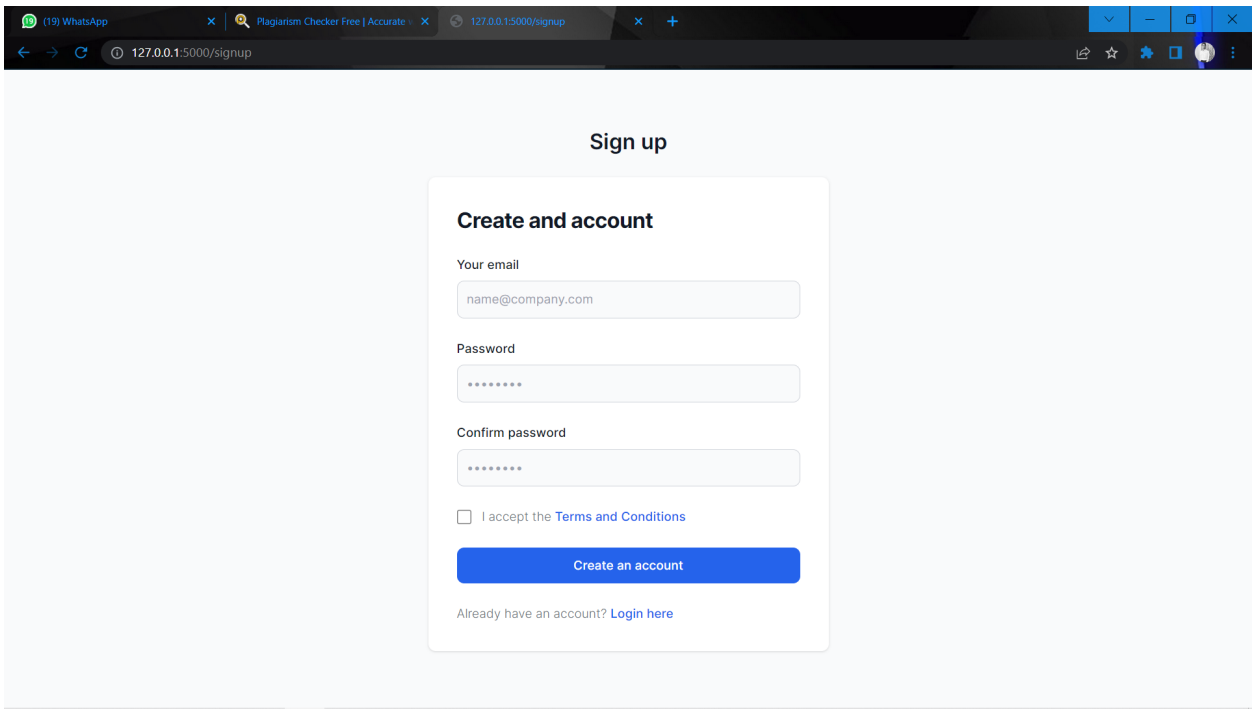
2. Detailed View of Home Page



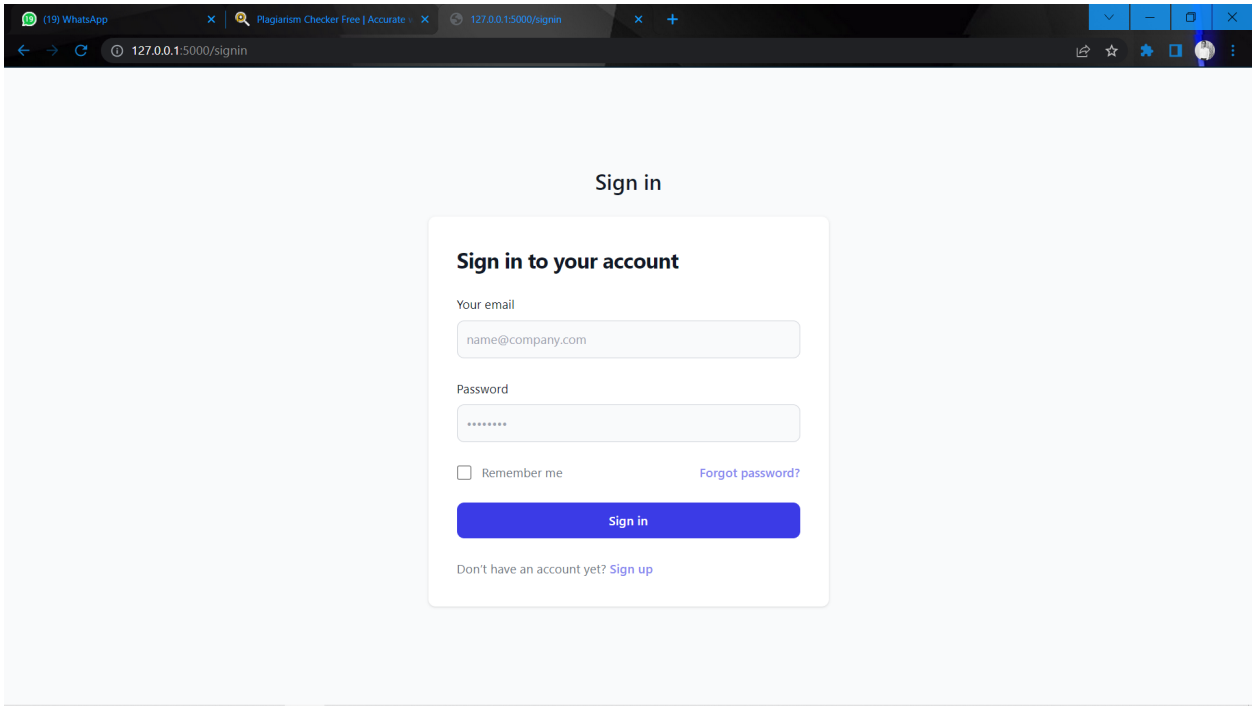
3.Contact to Admin page



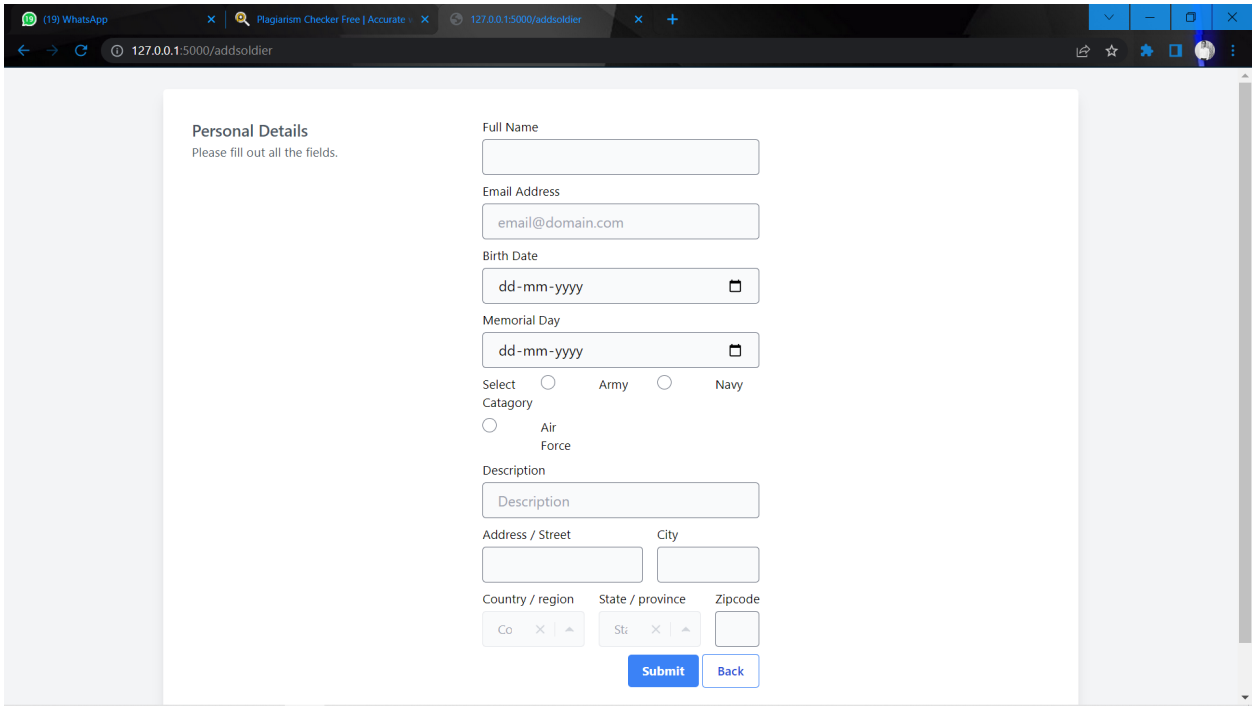
4.Dashboard where users can sign up.



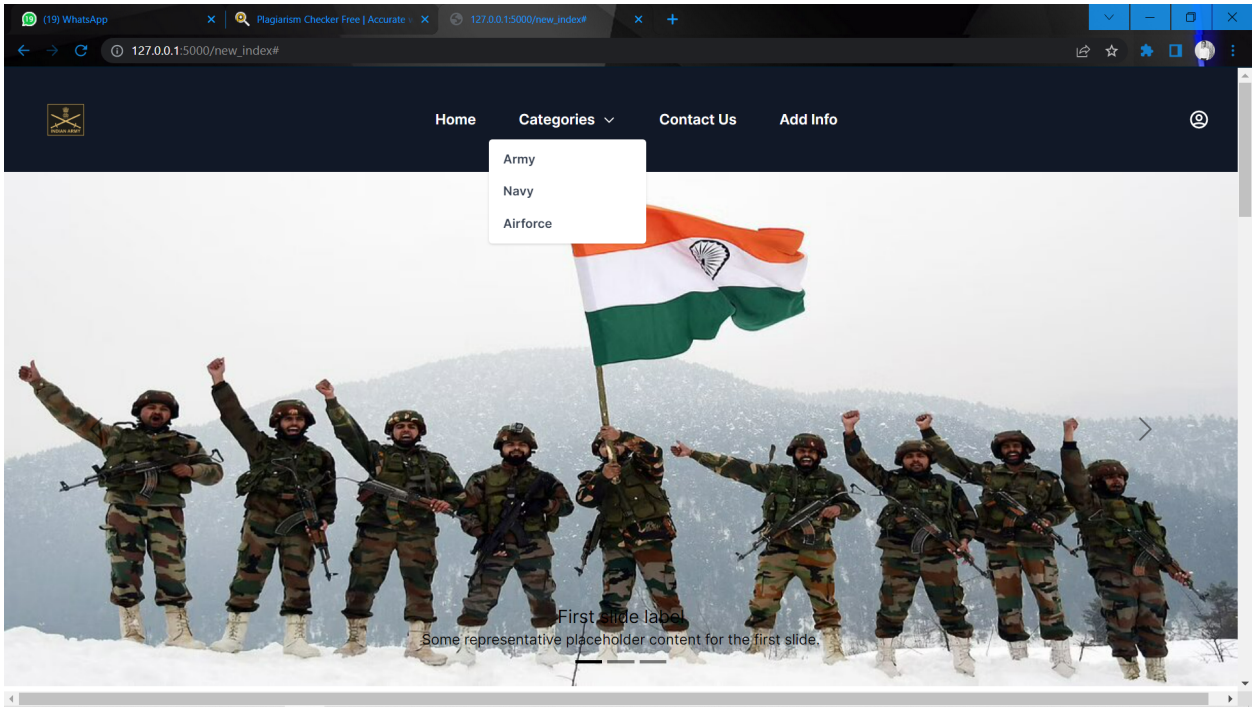
5.Registration page for users



6.Page where user can fill details for registration (Registration Form).



7.Categories Section for Users.



10. SAMPLE CODE

```
from flask import Flask,render_template,request,session,redirect
from flask_sqlalchemy import SQLAlchemy # This is to connect Database to front-end
from flask_login import UserMixin,login_user,logout_user,login_manager
from werkzeug.security import generate_password_hash, check_password_hash
from flask_login import login_required , current_user
from flask_login.login_manager import LoginManager
from flask import url_for
from flask import Flask, send_from_directory
flag = 0
local_server = True
app = Flask(__name__)
app.secret_key = 'dbmsproject'
#  For images
# @app.route('/<path:path>', methods=['GET'])
# def static_proxy(path):
#   return send_from_directory('./images', path)

# picFolder = os.path.join('templates','images')
# app.config(['UPLOAD_FOLDER']) = picFolder

app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql://root:@localhost/army'
db=SQLAlchemy(app)


login_manager = LoginManager(app)
login_manager.login_view = 'signin'


class Signup(UserMixin,db.Model):
```

```
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(30))
    password = db.Column(db.String(100))

class Soldier(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100))
    birthdate = db.Column(db.String(100))
    memorialday = db.Column(db.String(100))
    decription = db.Column(db.String(100))
    types = db.Column(db.String(100))
    address = db.Column(db.String(100))
    city = db.Column(db.String(30))
    zipcode = db.Column(db.Integer)




@login_manager.user_loader
def load_user(signup_id):
    return Signup.query.get(int(signup_id))



@app.route("/")
def index():
    return render_template('index.html')

# This id for soldier

@app.route("/addsoldier" , methods=['POST','GET'])
def addsoldier():
```

```
    if request.method=="POST":
        flag = 2
        name=request.form.get('name')
        email=request.form.get('email')
        birthdate = request.form.get('birthdate')
        memorialday = request.form.get('memorialday')
        type = request.form.get('type')
        desc = request.form.get('desc')
        address = request.form.get('address')
        city = request.form.get('city')
        zipcode = request.form.get('zipcode')
            new_user = db.engine.execute(f"INSERT INTO `soldier` ( `name`, `email`,`birthdate`, `memorialday`, `description`, `type`, `addresss`, `city`, `zipcode`) VALUES ( '{name}','{email}', '{birthdate}', '{memorialday}', '{desc}', '{type}', '{address}', '{city}', '{zipcode}');")


        print("This is  Post Buddy")



    if request.method=="GET":
        flag = 1
        print("This is get method")


    return render_template('addsoldier.html')


@app.route("/test")
def test():
    try:
        Notes.query.all()
        return 'Success'
    except:
        return 'Fail to connect'
```

```
@app.route("/contactus")
def contactus():
    return render_template('contactus.html')



@app.route("/signup" , methods=['POST','GET'])
def signup():
    if request.method=="POST":
        email=request.form.get('email')
        password=request.form.get('password')
        print("email =" , email)
        print("Password = " , password)

        user=Signup.query.filter_by(email=email).first()
        if user:
            print("Email already existed")
            return render_template('/signup.html')


        else:
            encpassord = generate_password_hash(password)
            print("Stm 1")
              # new_user = db.engine.execute(f" INSERT INTO 'user' ('username','email','password')
VALUES ('{username}' , '{email}' , '{encpassord}');")
                    new_user = db.engine.execute(f"INSERT INTO `signup` (`email`, `password`)
VALUES ('{email}', '{password}');")
            print("Stm 2")
            return render_template('signin.html')
    # print("Method = ")
    # print(request.method)
```

```
    return render_template('signup.html')


@app.route("/new_index",methods=['POST','GET'])
@login_required
def new_index():
    # print("Catagory ",catagory)
    if request.method=="POST":
        print(" This is POST")
        return render_template('new_index.html')
    else:
        print(" This is GET")

        return render_template('new_index.html')


@app.route("/logout")
@login_required
def logout():
    flag = 0
    logout_user()
    return redirect(url_for('signin'))



@app.route("/showarmy")
def showarmy():
    query = db.engine.execute("SELECT * FROM `soldier` WHERE type='army'")
    return render_template('showarmy.html', query=query)

@app.route("/shownavy")
def shownavy():
    query = db.engine.execute("SELECT * FROM `soldier` WHERE type='navy'")
```

```
    return render_template('shownavy.html', query=query)

@app.route("/showairforce")
def showairforce():
    query = db.engine.execute("SELECT * FROM `soldier` WHERE type='airforce'")
    return render_template('showairforce.html', query=query)



@app.route("/signin", methods=['POST','GET'])
def signin():
    if request.method=="POST":
        email=request.form.get('email')
        password=request.form.get('password')
        user = Signup.query.filter_by(email=email).first()
        if user and user.password==password:
            print(" You can proceed :) ")
            login_user(user)
            return redirect(url_for('new_index'))
        else:
            print("Invalid details")
            return render_template('signin.html')

        print(email , password)
        return render_template('signin.html')
    return render_template('signin.html')

    return render_template('signin.html')

# @app.route("/signup")
# def signup():
#     return render_template('signup.html')
print("flag = ",flag)
```

```
app.run(debug=True , threaded=True)
```

11.CONCLUSION

By this project we are able to fine untold stories martyr soldiers. This will be the inspiration for upcoming generation.

By this project also learned the CRUD operations in DBMS, connectivity of front end with backend and performing various operations.

12. REFERENCES

Online References

1) www.mongodb.org

2) **stackoverflow**.com

3) www.oracle.com (Java software download)

4)https://www.python.org/

Books Reference

1)**Python Phrasebook**