Blood-Connect: Bridging Lives Through Blood Donation

PROJECT SUBMITTED TO JAIN (DEEMED-TO-BE-UNIVERSITY), KOCHI FOR MSc DATA SCIENCE & ANALYTICS

DEPARTMENT OF DATA ANALYTICS AND MATHEMATICAL SCIENCES, JAIN DEEMED TO BE UNIVERSITY, NIRMAL INFOPARK, KAKKANAD, KOCHI – 682042

Telephone: 0484 435 5555

Email: enquiry.kochi@jainuniversity.ac.in

Certificate

Certified that this is the Bonafide record of work done in the Artificial Intelligence Laboratory as part of the Group Project titled Blood-Connect: Bridging Lives Through Blood Donation" by the following students: ADHIL NAZIM NAZEER (24MSKR0002), HIMA PRABHAKAR K(24MSKR0013), LAKSHMI NANDANA M S (24MSKR0016), SHALU AJAYAN (24MSKR0019) and SONA VIMAL (24MSKR0021) of the First Semester MSc Data Science and Analytics Degree Course under the School of Data Analytics & Mathematical Sciences in JAIN Deemed to be University Kochi, during the academic year 2024 - 25.

Head of the Department	Faculty In-Charge
Dr. Sajithamony T M	Dr. Meenu Suresh
Assistant Professor	Teaching Assistant

Acknowledgement

I would like to express my sincere gratitude to all those who supported and guided me throughout the development of this project.

Firstly, I would like to thank DR. MEENU SURESH, my project supervisor, for their invaluable advice, encouragement, and continuous support. Their guidance helped me navigate through the challenges and stay focused on achieving the objectives of this project.

I am also grateful to JAIN DEEMED TO BE UNIVERSITY for providing the resources and platform necessary for this project. The learning environment and facilities were instrumental in enabling me to design and implement the student database system effectively.

Finally, I am indebted to the faculty members and fellow students who offered constructive feedback and suggestions to refine and enhance the project. Their input has been invaluable in shaping this project into its final form. Thank you all for your support and encouragement.

Abstract

Blood-Connect is an innovative application designed to revolutionize the blood donation process by creating an efficient and user-friendly platform. It bridges the gap between blood donors, recipients, and healthcare institutions by streamlining the registration, management, and search for vital blood donation resources. The platform supports:

- **Donors:** A simple registration process to input essential details, enabling quick identification of compatible recipients.
- **Receivers:** Easy access to a database of potential donors to facilitate lifesaving matches during emergencies.
- **Hospitals:** Tools to organize and manage blood donation campaigns and access donor and recipient data seamlessly.

By integrating advanced machine learning algorithms, Blood-Connect ensures compatibility checks and prioritization of matches based on proximity, urgency, and medical needs. The platform also employs robust database management to store and retrieve critical data efficiently, providing a reliable solution during time-sensitive scenarios. Blood-Connect fosters a community-driven approach to saving lives by making the blood donation process transparent, accessible, and efficient for all stakeholders.

Table of Contents

Introduction

Blood-Connect addresses the critical need for an efficient and responsive blood donation system. In many emergencies, the availability of compatible blood can make the difference between life and death. Recognizing this urgent necessity, Blood-Connect has been designed to bring donors, receivers, and hospitals onto a single platform to facilitate seamless coordination.

The application offers a comprehensive set of features tailored to streamline the process of finding and managing blood resources. Donors can register their details, including blood type and location, making it easier to match them with those in need. Receivers can search for compatible donors nearby, ensuring timely access to life-saving donations. Hospitals can leverage the platform to organize blood donation drives, manage their donor database, and coordinate with patients efficiently.

Blood-Connect utilizes cutting-edge technologies, such as machine learning algorithms, to enhance its donor-receiver matching capabilities. These algorithms consider factors like blood group compatibility and geographic proximity, ensuring that matches are both medically suitable and logistically feasible. Additionally, the system's robust database architecture guarantees transparency, ease of access, and reliability, even in high-pressure emergency scenarios.

By fostering a collaborative and community-driven approach, Blood-Connect not only simplifies the process of blood donation but also strengthens the network of support between donors, patients, and healthcare providers. This initiative is a step forward in addressing the global challenge of blood shortages and ensuring that life-saving resources are always within reach.

Objectives

1. **Streamline the Registration Process**: Simplify the process of registering blood donors, receivers, and hospitals by providing an intuitive and user-friendly interface. This ensures quick and hassle-free registration for all users, reducing barriers to participation.
2. **Implement a Reliable Database System:** Develop a robust database infrastructure for secure storage and retrieval of critical data. This database will store information about donors, receivers, hospitals, and past transactions, ensuring data consistency and integrity.
3. **Provide Search Functionality**: Enable efficient search capabilities for users to find blood donors based on specific criteria such as blood group and geographic location. This functionality facilitates timely responses in emergency situations.
4. **Ensure Compatibility Checks**: Integrate blood compatibility algorithms to verify safe matches between donors and receivers. This ensures the safety and effectiveness of every blood transfusion conducted through the platform.
5. **Develop a User-Friendly Interface:** Utilize Streamlit to create an accessible and visually appealing interface. The interface will guide users through various features and processes, making the platform easy to navigate for users of all technical backgrounds.
6. **Incorporate Advanced Features**: Implement machine learning techniques such as K-Nearest Neighbors (KNN) to rank donors based on historical data. This includes factors like proximity, blood compatibility, and donation frequency to provide the most relevant matches.
7. **Foster a Community Approach:** Encourage collaboration between donors, receivers, and hospitals to build a supportive network that prioritizes saving lives. The platform's features are designed to strengthen these connections and promote awareness about the importance of blood donation.

Algorithm

1. **Initialize Database:**
    - Create tables for donors, receivers, and hospitals.
    - Ensure tables have appropriate fields like ID, name, phone number, location, and blood group.

2. **Add Records:**
    - Use forms in Streamlit to add donors, receivers, and hospitals.
    - Validate inputs and commit to the database.

3. **Search Functionality:**
    - Fetch data based on blood group and location.
    - Use blood group compatibility logic for matching.

4. **Donor Ranking (KNN):**
    - Train a KNN model using historical data.
    - Use features like geographic distance, blood compatibility, and donation frequency.

5. **Data Display and Management:**

    - Provide options to view, delete, and update records.

Scope

1. **Simplifies the Connection Between Stakeholders**: Blood-Connect bridges the gap between donors, receivers, and hospitals, creating an integrated system that fosters efficient communication and collaboration.

2. **Provides Real-Time Data Access**: The platform enables stakeholders to access real-time data about available donors, urgent requirements, and hospital inventories, facilitating prompt decision-making and response.

3. **Facilitates Emergency Responses**: During crises, the system's robust search and compatibility features expedite the matching process, ensuring timely blood transfusions and minimizing delays.

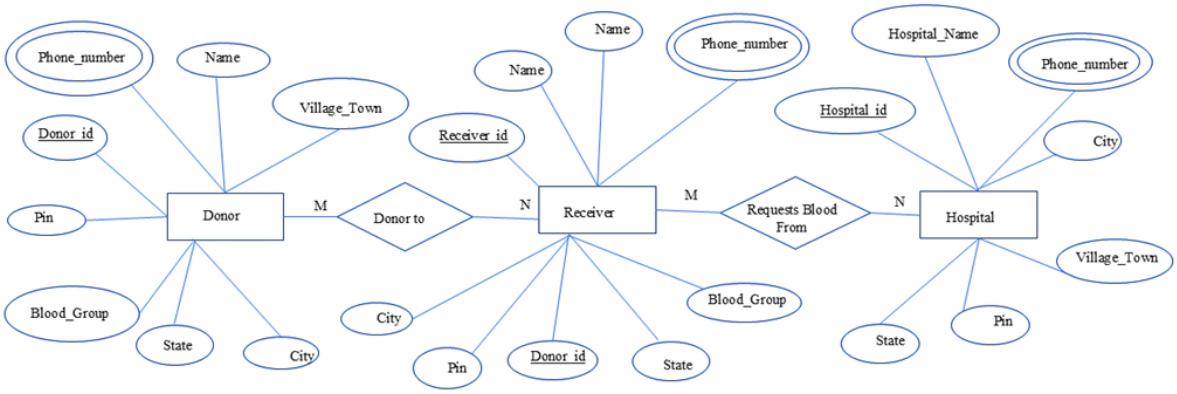
4. **Ensures Accurate Matching with Machine Learning**: By leveraging machine learning techniques like KNN, Blood-Connect prioritizes donor matches based on compatibility, availability, and proximity, enhancing the reliability of its recommendations.

5. **Supports Future Enhancements**: Blood-Connect is designed with scalability in mind, paving the way for advanced features such as automated scheduling, push notifications for donation drives, and integration with healthcare systems. These additions will further enrich its capabilities and impact.

6. **Promotes Community Engagement**: The application encourages users to participate in blood donation drives and educates them about the importance of regular donations. This cultivates a sense of community responsibility and collective effort to save lives.

Entity-Relationship Diagram

- Entities: Donors, Receivers, Hospitals
- Relationships:
    - Donors and Receivers connected through blood compatibility.
    - Hospitals manage campaigns and donor-receiver coordination.



Database Design

The system is structured with the following tables:

Table 1:

1. **Donors:**
    - ID (Primary Key), Name, Phone Number, Village/Town, City, State, PIN, Blood Group

Name	Phone Number	Village _Town	City	State	Pin	Blood Group
Anu	8899901267	Pine hollow	Kannur	Kerala	680569	A^+
Ahdal	9847123779	Riverstone	Palakkakd	Kerala	110001	B^-
Malu	8891973779	Ashford	Trivandrum	Kerala	560001	A^-
Akhila	7012804351	Redhill	Malappuram	Kerala	122001	AB^+
Amal	8606206796	Kingsbridge	Alappuzha	Kerala	654390	0^-

Table 2:

2. **Receivers:**
    - ID (Primary Key), Name, Phone Number, Village/Town, City, State, PIN, Blood Group

Name	Phone Number	Village_Town	City	State	Pin	Blood Group
Midhun	9867444002	Oakwood	Kollam	Kerala	400001	0–
Akshaya	8876900021	Ashford	Kottayam	Kerala	564321	A^-
Sreya	9723600671	Brackenbury	Idukki	Kerala	685743	AB^+

Adith	8894467551	Redhill	Kannur	Kerala	680569	B^+
Aswin	7644300871	Eversham	wayanad	Kerala	234667	A^+

Table 3:

3. **Hospitals:**
    - ID (Primary Key), Hospital Name, Phone Number, Village/Town, City, State, PIN

Hospital_Name	Village_Town	City	State	Pin	Phone_Number
Amritha Hospital	Kingsbridge	Kannur	Kerala	685743	8899901267
Amala Hospital	Redhill	Idukki	Kerala	680569	9847123779
Daya Hospital	Ashford	Kollam	Kerala	234667	9867444002
Mother Hospital	Pine hollow	Trivandrum	Kerala	122001	8876900021
Jilla mission Hospital	Oakwood	Alappuzha	Kerala	654390	9723600671

Implementation

```
#working final
import streamlit as st
import pymysql
from streamlit_extras.let_it_rain import rain
import pandas as pd
from sklearn.neighbors import NearestNeighbors
from datetime import datetime, timedelta
import numpy as np
from geopy.distance import geodesic

# Database connection setup
def get_connection():
    return pymysql.connect(
        host='localhost',
        user='root',
        password='',  # Replace with your MySQL password
        database='blood_donation',
        cursorclass=pymysql.cursors.DictCursor
    )

# Initialize database tables
def initialize_tables():
    connection = get_connection()
    try:
        with connection.cursor() as cursor:
            # Donor table
            cursor.execute('''CREATE TABLE IF NOT EXISTS donors (
                                id INT AUTO_INCREMENT PRIMARY KEY,
                                name VARCHAR(255),
                                phone_number VARCHAR(20),
                                village_town VARCHAR(255),
                                city VARCHAR(255),
                                state VARCHAR(255),
                                pin VARCHAR(10),
                                blood_group VARCHAR(5)
                            )''')
```

```
        # Receiver (Patient) table
        cursor.execute('''CREATE TABLE IF NOT EXISTS receivers (
                          id INT AUTO_INCREMENT PRIMARY KEY,
                          name VARCHAR(255),
                          phone_number VARCHAR(20),
                          village_town VARCHAR(255),
                          city VARCHAR(255),
                          state VARCHAR(255),
                          pin VARCHAR(10),
                          blood_group VARCHAR(5)
                        )''')

        # Hospital table
        cursor.execute('''CREATE TABLE IF NOT EXISTS hospitals (
                          id INT AUTO_INCREMENT PRIMARY KEY,
                          hospital_name VARCHAR(255),
                          phone_number VARCHAR(20),
                          village_town VARCHAR(255),
                          city VARCHAR(255),
                          state VARCHAR(255),
                          pin VARCHAR(10)
                        )''')
    connection.commit()
  finally:
    connection.close()


# Add a donor
def add_donor(name, phone_number, village_town, city, state, pin, blood_group):
  connection = get_connection()
  try:
    with connection.cursor() as cursor:
      cursor.execute('''INSERT INTO donors (name, phone_number, village_town, city, state, pin,
blood_group)
                        VALUES (%s, %s, %s, %s, %s, %s, %s)''',
                     (name, phone_number, village_town, city, state, pin, blood_group))
    connection.commit()
  finally:
    connection.close()
```

```python
# Get list of donors

def get_donors():
    connection = get_connection()
    try:
        with connection.cursor() as cursor:
            cursor.execute('SELECT * FROM donors')
            donors = cursor.fetchall()  # Returns a list of dictionaries

                # Convert the list of dictionaries to a pandas DataFrame
            df = pd.DataFrame(donors)
            st.write("Donor Information")
            #st.dataframe(df)  # Display the DataFrame in Streamlit

            return df
    except Exception as e:
        st.write(f"An error occurred: {e}")
        return []
    finally:
        connection.close()


# Delete a donor
def delete_donor(donor_id):
    connection = get_connection()
    try:
        with connection.cursor() as cursor:
            cursor.execute('DELETE FROM donors WHERE id = %s', (donor_id,))
        connection.commit()
    finally:
        connection.close()

# Add a receiver
def add_receiver(name, phone_number, village_town, city, state, pin, blood_group):
    connection = get_connection()
    try:
        with connection.cursor() as cursor:
```

```
        cursor.execute('''INSERT INTO receivers (name, phone_number, village_town, city, state, pin,
blood_group)
                          VALUES (%s, %s, %s, %s, %s, %s, %s)''',
                       (name, phone_number, village_town, city, state, pin, blood_group))
        connection.commit()
    finally:
        connection.close()

# Get list of receivers
def get_receivers():
    connection = get_connection()
    try:
        with connection.cursor() as cursor:
            cursor.execute('SELECT * FROM receivers')
            receiver = cursor.fetchall()
            df = pd.DataFrame(receiver)
            st.write("Receiver Information")
            return df
    finally:
        connection.close()

# Delete a receiver
def delete_receiver(receiver_id):
    connection = get_connection()
    try:
        with connection.cursor() as cursor:
            cursor.execute('DELETE FROM receivers WHERE id = %s', (receiver_id,))
        connection.commit()
    finally:
        connection.close()

# Add a hospital
def add_hospital(hospital_name, phone_number, village_town, city, state, pin):
    connection = get_connection()
    try:
        with connection.cursor() as cursor:
            cursor.execute('''INSERT INTO hospitals (hospital_name, phone_number, village_town, city, state, pin)
                          VALUES (%s, %s, %s, %s, %s, %s)''',
                       (hospital_name, phone_number, village_town, city, state, pin))
```

```
        connection.commit()
    finally:
        connection.close()

# Get list of receivers
def get_hosptials():
    connection = get_connection()
    try:
        with connection.cursor() as cursor:
            cursor.execute('SELECT * FROM hospitals')
            hospital = cursor.fetchall()
            df = pd.DataFrame(hospital)
            st.write("Hospital Information")
            return df
    finally:
        connection.close()

# Delete a hospital
def delete_hospital(hospital_id):
    connection = get_connection()
    try:
        with connection.cursor() as cursor:
            cursor.execute('DELETE FROM hospitals WHERE id = %s', (hospital_id,))
        connection.commit()
    finally:
        connection.close()

# Search donors by blood group and city
def search_donors_by_blood_group_and_city(blood_group, city):
    connection = get_connection()
    try:
        with connection.cursor() as cursor:
            cursor.execute('SELECT * FROM donors WHERE blood_group = %s OR city = %s', (blood_group,
city))
            return cursor.fetchall()
    finally:
        connection.close()
```

```
# Fetch Compatible Donors
def get_compatible_donors(receiver, donors_data):
    donors = pd.DataFrame(donors_data)  # Convert to DataFrame
    compatible_blood_groups = blood_compatibility(receiver['blood_group'])
    compatible_donors = donors[
        (donors['blood_group'].isin(compatible_blood_groups)) & (donors['city'] == receiver['city'])
    ]
    return compatible_donors


def train_knn_model(data):
    """
    Trains a KNN model to rank donors based on historical matches.

    Parameters:
        data (pd.DataFrame): Historical data including features like distance, blood compatibility, and frequency.

    Returns:
        NearestNeighbors: Trained KNN model.
    """
    features = data[['distance', 'blood_compatibility', 'frequency']]
    model = NearestNeighbors(n_neighbors=5, metric='euclidean')
    model.fit(features)
    return model

def predict_matches(receiver, donors, model):
    """
    Predicts the top matches for a receiver using a trained KNN model.

    Parameters:
        receiver (dict): Receiver's details.
        donors (pd.DataFrame): Donors' data.
        model (NearestNeighbors): Trained KNN model.

    Returns:
        pd.DataFrame: Top matched donors.
    """
    receiver_features = pd.DataFrame([{
        'distance': geodesic((receiver['latitude'], receiver['longitude']),
```

```
                        (row['latitude'], row['longitude'])).km,
        'blood_compatibility': 1 if row['blood_group'] in blood_compatibility[receiver['blood_group']] else 0,
        'frequency': (datetime.now() - pd.to_datetime(row['last_donation_date'])).days
    } for _, row in donors.iterrows()])

    distances, indices = model.kneighbors(receiver_features)
    return donors.iloc[indices.flatten()]

def blood_compatibility(donor_blood_group, receiver_blood_group):
    # Define compatibility logic
    compatibility = {
        "O-": ["O-"],
        "O+": ["O-", "O+"],
        "A-": ["O-", "A-"],
        "A+": ["O-", "O+", "A-", "A+"],
        "B-": ["O-", "B-"],
        "B+": ["O-", "O+", "B-", "B+"],
        "AB-": ["O-", "A-", "B-", "AB-"],
        "AB+": ["O-", "O+", "A-", "A+", "B-", "B+", "AB-", "AB+"],
    }
    return receiver_blood_group in compatibility.get(donor_blood_group, [])

# Streamlit UI
st.title("Blood Donation")
st.header("Blood-Connect: Bridging Lives Through Blood Donation")
st.subheader("Simplifying the Connection Between Donors, Patients, and Hospitals for Lifesaving Support")


# Display Images

# import Image from pillow to open images
from PIL import Image
img = Image.open("C:/Users/HP/OneDrive/Desktop/AI Project/blood donate.png")

# display image using streamlit
# width is used to set the width of an image
st.image(img, width=200)

initialize_tables()
```

```
menu = ["Home","Donor", "Receiver", "Hospital", "Search"]
choice = st.sidebar.selectbox("Menu", menu)

if choice == "Home":
    st.subheader("About")
    # Raining Emoji
    rain(
        emoji="🩸",
        font_size=10,  # the size of emoji
        falling_speed=10,  # speed of raining
        animation_length="infinite",  # for how much time the animation will happen
        )
    about="""\n\nBlood-Connect is a streamlined web-based application designed to facilitate blood donation management by connecting donors, receivers, and hospitals. With the increasing demand for efficient blood donation services, this platform simplifies the process of registering donors and receivers, while allowing hospitals to stay organized and accessible. Blood-Connect provides an easy-to-use interface for users to add, view, and manage critical data, ensuring quick and accurate access during emergencies.
\n\nFor blood donors, the platform enables seamless registration with essential details like name, blood group, and location, allowing receivers and hospitals to identify compatible donors swiftly. Similarly, patients or their families can register themselves as receivers, making it easier to match them with available donors. Hospitals, on the other hand, can use the system to organize their blood donation campaigns, connect with donors, and schedule donations efficiently. By centralizing these features, Blood-Connect ensures that lifesaving resources are always within reach.
\n\nThe app also includes a robust search feature that helps users find donors based on blood group and city. This is especially helpful during urgent situations when time is of the essence. Blood-Connect bridges the gap between those in need and those willing to help, fostering a community-driven approach to saving lives through blood donation. Whether you are a donor, a patient, or a hospital, Blood-Connect is here to make blood donation simple, transparent, and impactful.
"""
    st.text(about)

elif choice == "Donor":
    donor_action = st.selectbox("Choose an action", ["Add Donor", "Donors List", "Delete Donor"])

    if donor_action == "Add Donor":
        st.subheader("Add Donor")
        with st.form("donor_form"):
            name = st.text_input("Name")
```

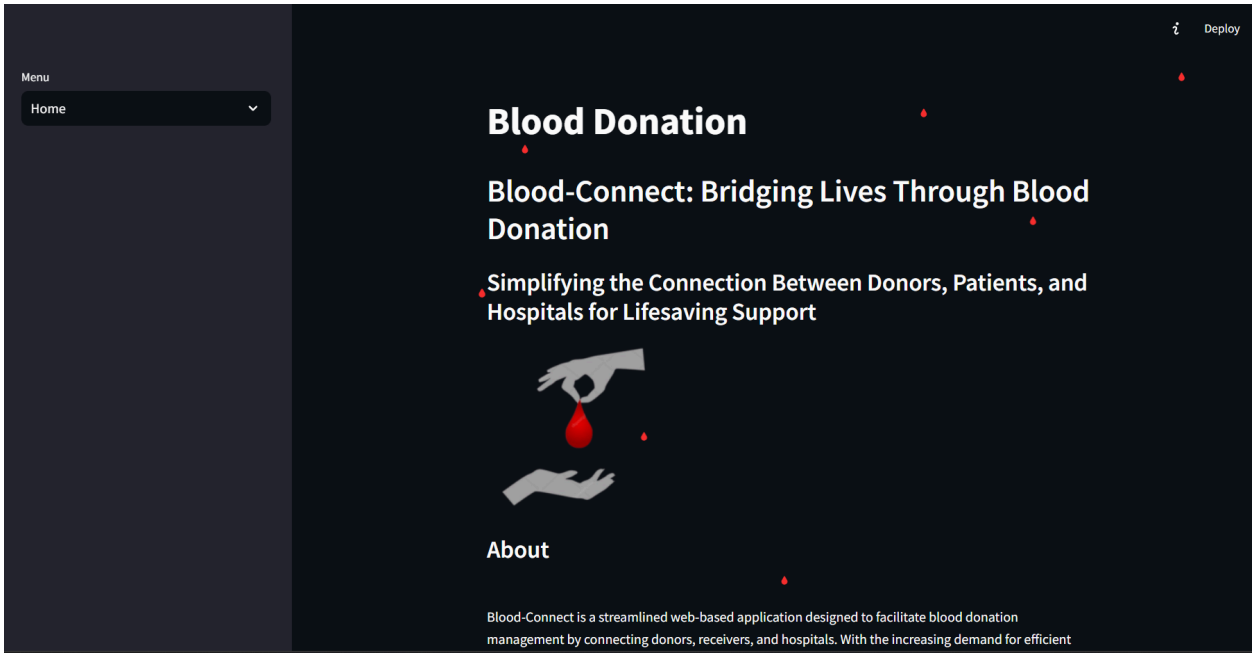
```
            phone_number = st.text_input("Phone Number")
            village_town = st.text_input("Village/Town")
            city = st.text_input("City")
            state = st.text_input("State")
            pin = st.text_input("PIN")
            blood_group = st.text_input("Blood Group")
            submitted = st.form_submit_button("Add Donor")
            if submitted:
                add_donor(name, phone_number, village_town, city, state, pin, blood_group)
                st.success("Donor added successfully!")

    elif donor_action == "Donors List":
        st.subheader("Donors List")
        donors = get_donors()
        st.write(donors)

    elif donor_action == "Delete Donor":
        st.subheader("Delete Donor")
        donor_id = st.text_input("Enter Donor ID")
        if st.button("Delete Donor"):
            delete_donor(donor_id)
            st.success("Donor deleted successfully!")

elif choice == "Receiver":
    receiver_action = st.selectbox("Choose an action", ["Add Receiver", "List of Receivers", "Delete
Receiver"])

    if receiver_action == "Add Receiver":
        st.subheader("Add Receiver")
        with st.form("receiver_form"):
            name = st.text_input("Name")
            phone_number = st.text_input("Phone Number")
            village_town = st.text_input("Village/Town")
            city = st.text_input("City")
            state = st.text_input("State")
            pin = st.text_input("PIN")
            blood_group = st.text_input("Blood Group")
            submitted = st.form_submit_button("Add Receiver")
            if submitted:
```

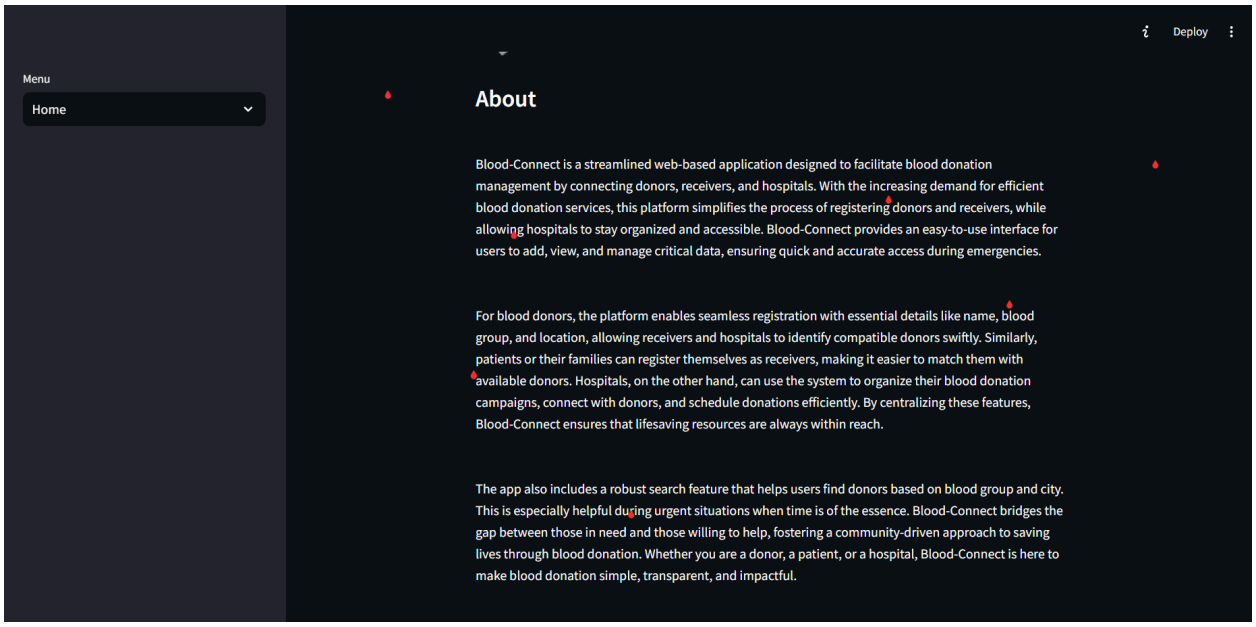
```
            add_receiver(name, phone_number, village_town, city, state, pin, blood_group)
            st.success("Receiver added successfully!")

    elif receiver_action == "List of Receivers":
        st.subheader("List of Receivers")
        receivers = get_receivers()
        st.write(receivers)

    elif receiver_action == "Delete Receiver":
        st.subheader("Delete Receiver")
        receiver_id = st.text_input("Enter Receiver ID")
        if st.button("Delete Receiver"):
            delete_receiver(receiver_id)
            st.success("Receiver deleted successfully!")

elif choice == "Hospital":
    hospital_action = st.selectbox("Choose an action", ["Add Hospital","List of hopitals", "Delete Hospital"])

    if hospital_action == "Add Hospital":
        st.subheader("Add Hospital")
        with st.form("hospital_form"):
            hospital_name = st.text_input("Hospital Name")
            phone_number = st.text_input("Phone Number")
            village_town = st.text_input("Village/Town")
            city = st.text_input("City")
            state = st.text_input("State")
            pin = st.text_input("PIN")
            submitted = st.form_submit_button("Add Hospital")
            if submitted:
                add_hospital(hospital_name, phone_number, village_town, city, state, pin)
                st.success("Hospital added successfully!")

    elif hospital_action == "List of hopitals":
        st.subheader("List of hopitals")
        hospitals = get_hosptials()
        st.write(hospitals)

    elif hospital_action == "Delete Hospital":
        st.subheader("Delete Hospital")
```

```
        hospital_id = st.text_input("Enter Hospital ID")
        if st.button("Delete Hospital"):
            delete_hospital(hospital_id)
            st.success("Hospital deleted successfully!")

elif choice == "Search":
    st.subheader("Search Donors by Blood Group and City")

    # Blood group selection
    blood_group = st.selectbox("Select Blood Type", ["A+", "B+", "AB+", "O+", "A-", "B-", "AB-", "O-"])
    city = st.text_input("Enter City")

    if st.button("Search"):
        # Fetch compatible donors using the new logic
        matching_donors = search_donors_by_blood_group_and_city(blood_group, city)

        if matching_donors:
            st.write("Matching Donors:")
            for donor in matching_donors:
                st.write(f"Name: {donor['name']}, Blood Group: {donor['blood_group']}, City: {donor['city']},
Phone Number: {donor['phone_number']}")

            # Select donor for scheduling donation
            donor_ids = [donor['id'] for donor in matching_donors]
            selected_donor_id = st.selectbox(
                "Select Donor ID for Donation",
                donor_ids,
                format_func=lambda x: next(d for d in matching_donors if d['id'] == x)['name']
            )

        else:
            st.warning("No donors found for the specified criteria.")
```

Output

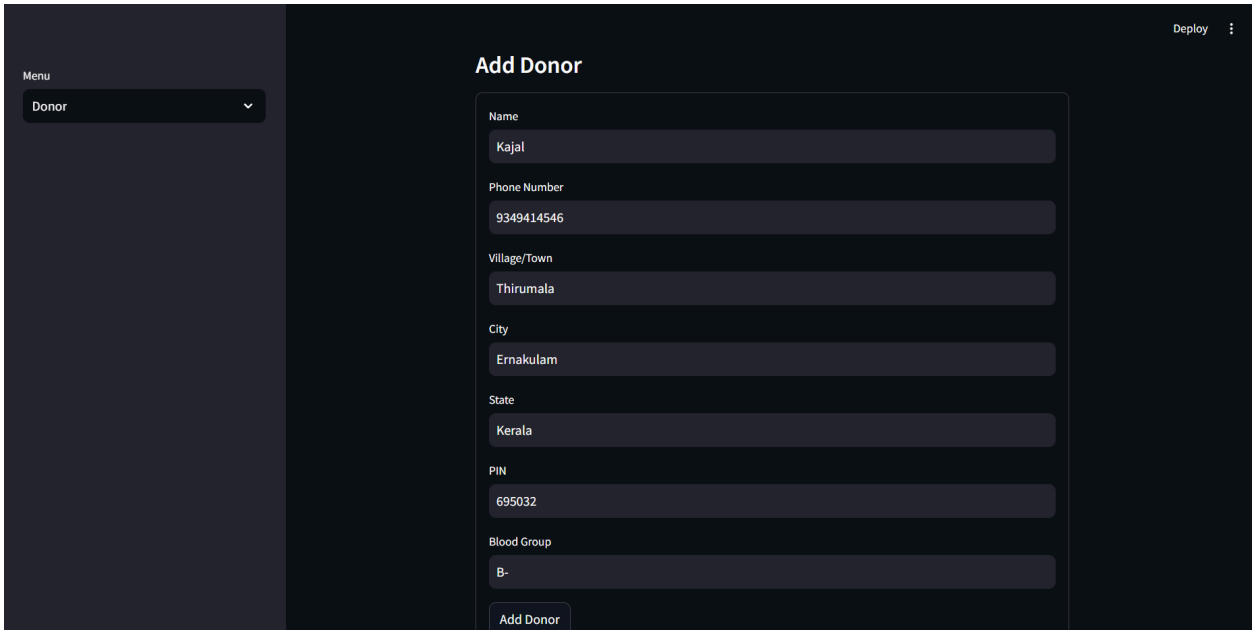
Home

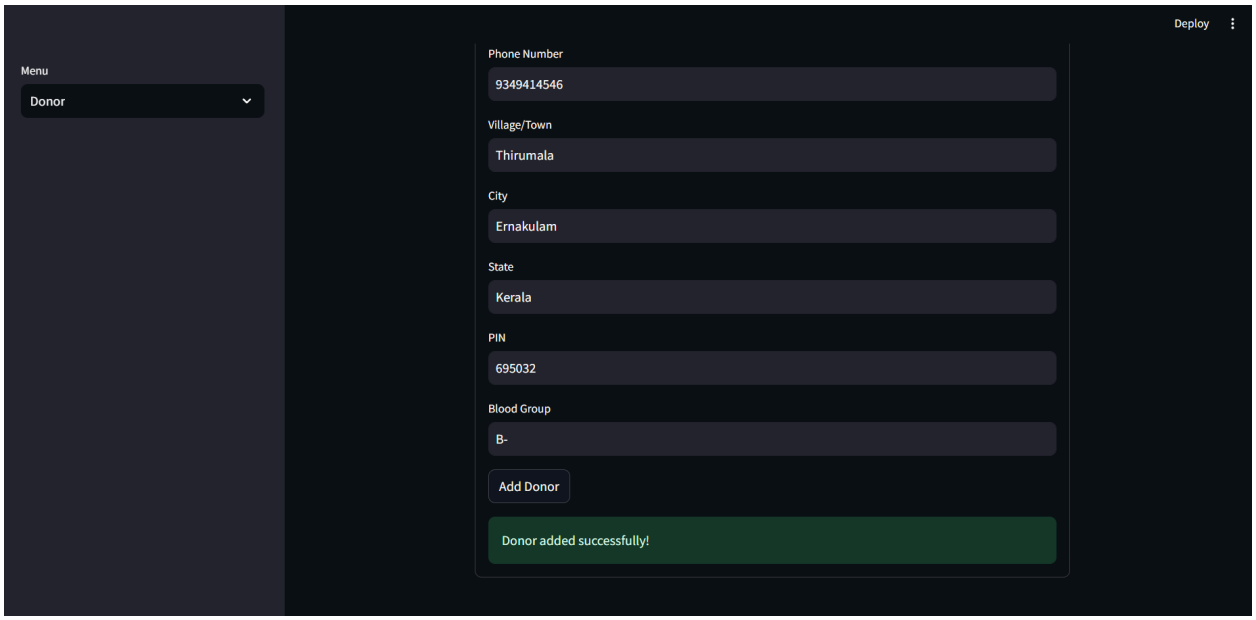

Menu
Home
Deploy
Blood Donation
Blood-Connect: Bridging Lives Through Blood Donation
Simplifying the Connection Between Donors, Patients, and Hospitals for Lifesaving Support
About
Blood-Connect is a streamlined web-based application designed to facilitate blood donation management by connecting donors, receivers, and hospitals. With the increasing demand for efficient


Menu
Home
Deploy
About
Blood-Connect is a streamlined web-based application designed to facilitate blood donation management by connecting donors, receivers, and hospitals. With the increasing demand for efficient blood donation services, this platform simplifies the process of registering donors and receivers, while allowing hospitals to stay organized and accessible. Blood-Connect provides an easy-to-use interface for users to add, view, and manage critical data, ensuring quick and accurate access during emergencies.
For blood donors, the platform enables seamless registration with essential details like name, blood group, and location, allowing receivers and hospitals to identify compatible donors swiftly. Similarly, patients or their families can register themselves as receivers, making it easier to match them with available donors. Hospitals, on the other hand, can use the system to organize their blood donation campaigns, connect with donors, and schedule donations efficiently. By centralizing these features, Blood-Connect ensures that lifesaving resources are always within reach.
The app also includes a robust search feature that helps users find donors based on blood group and city. This is especially helpful during urgent situations when time is of the essence. Blood-Connect bridges the gap between those in need and those willing to help, fostering a community-driven approach to saving lives through blood donation. Whether you are a donor, a patient, or a hospital, Blood-Connect is here to make blood donation simple, transparent, and impactful.

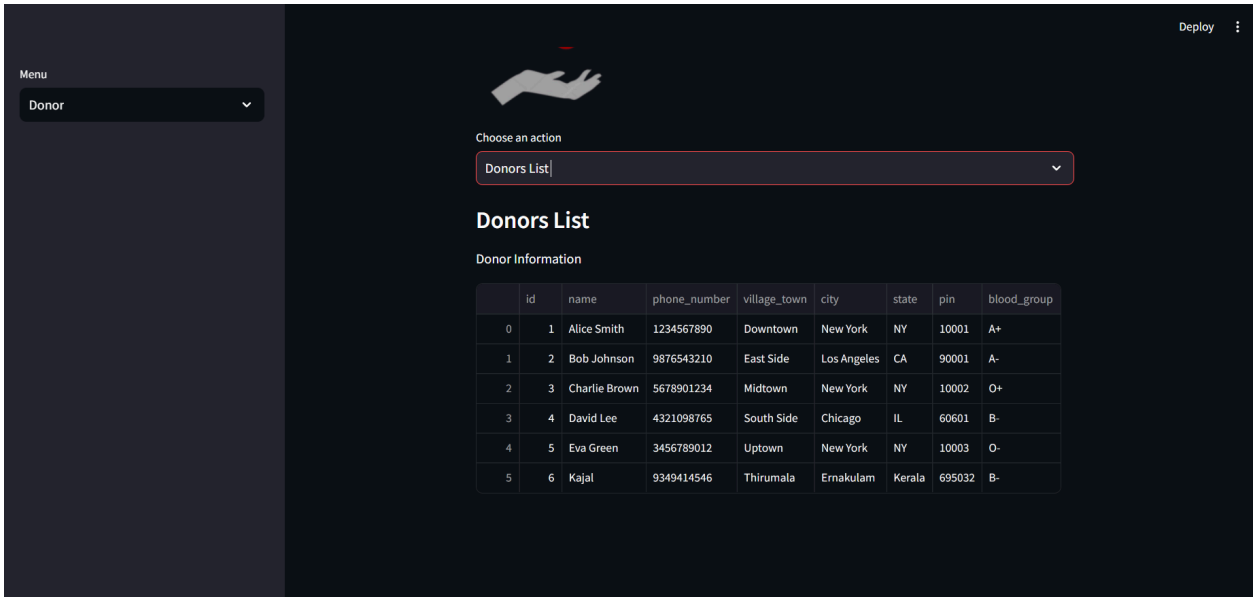
Donor

Add donor



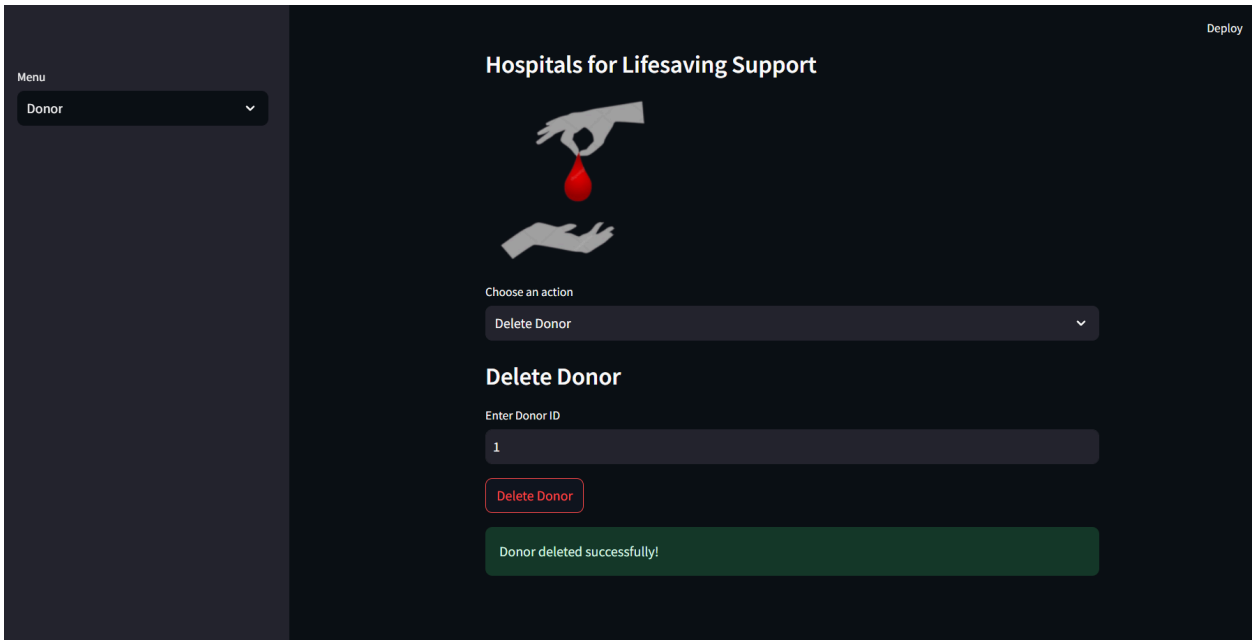


List of donor



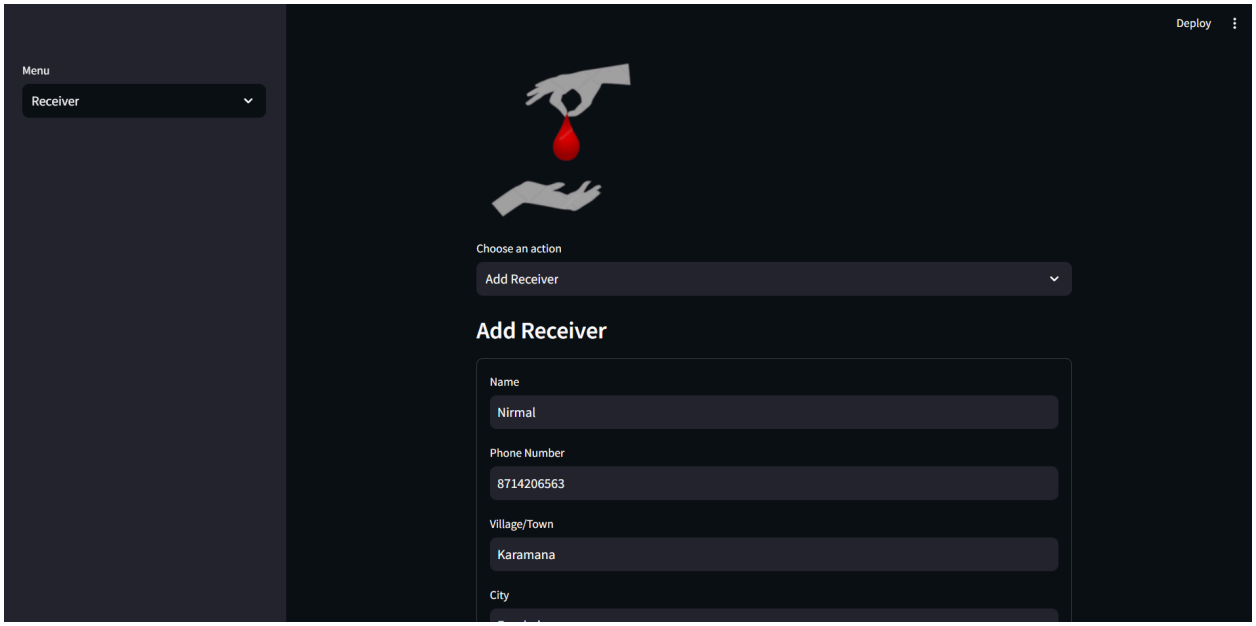
	id	name	phone_number	village_town	city	state	pin	blood_group
0	1	Alice Smith	1234567890	Downtown	New York	NY	10001	A+
1	2	Bob Johnson	9876543210	East Side	Los Angeles	CA	90001	A-
2	3	Charlie Brown	5678901234	Midtown	New York	NY	10002	O+
3	4	David Lee	4321098765	South Side	Chicago	IL	60601	B-
4	5	Eva Green	3456789012	Uptown	New York	NY	10003	O-
5	6	Kajal	9349414546	Thirumala	Ernakulam	Kerala	695032	B-

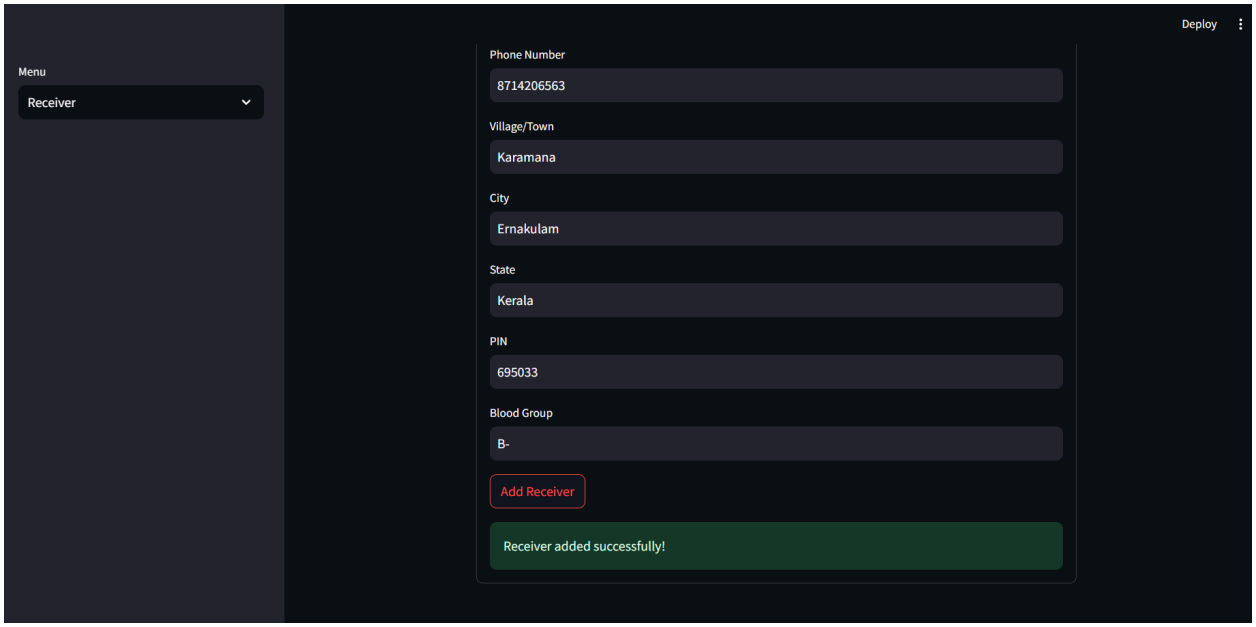
Delete donar



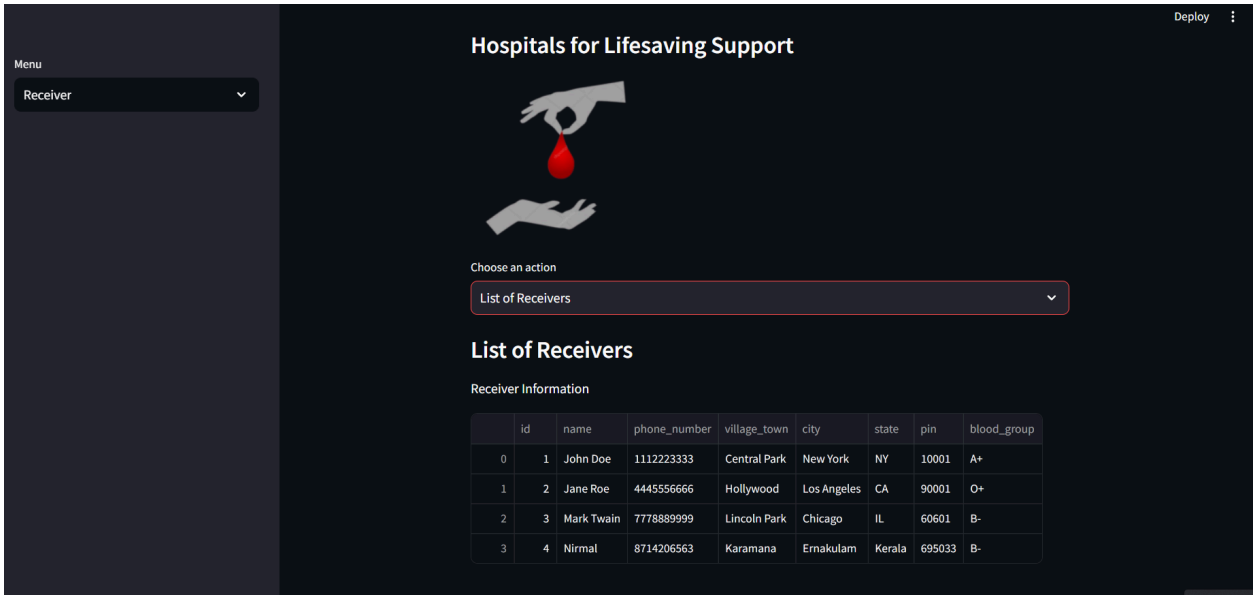
Receiver

Add receiver



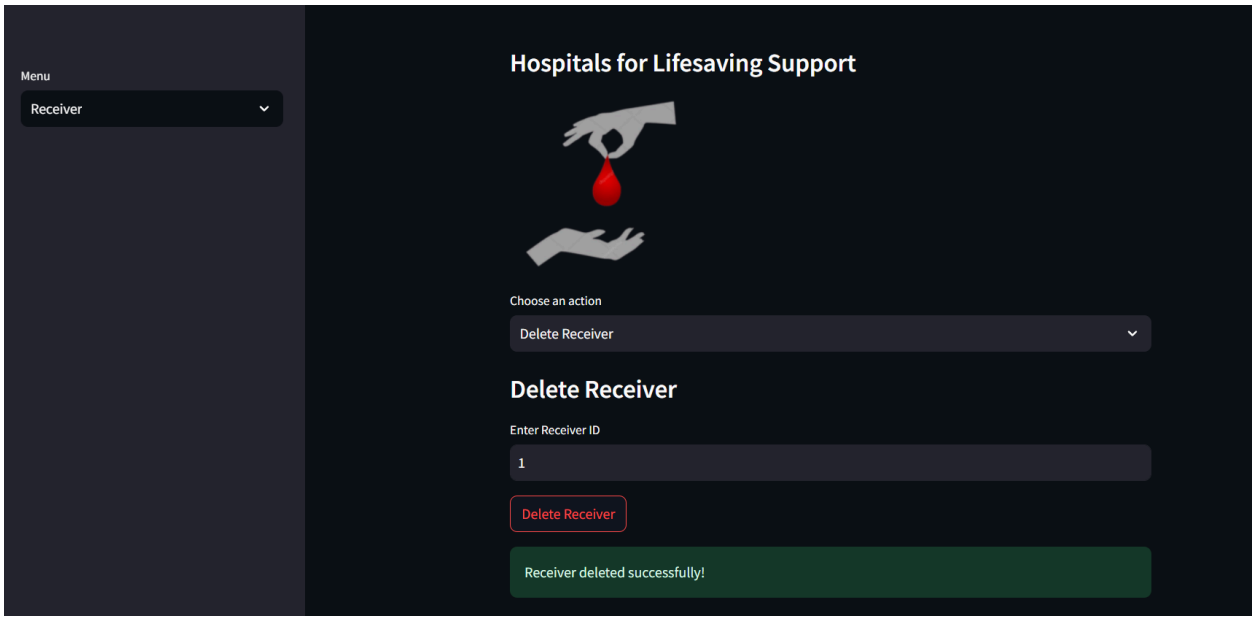


List of receiver



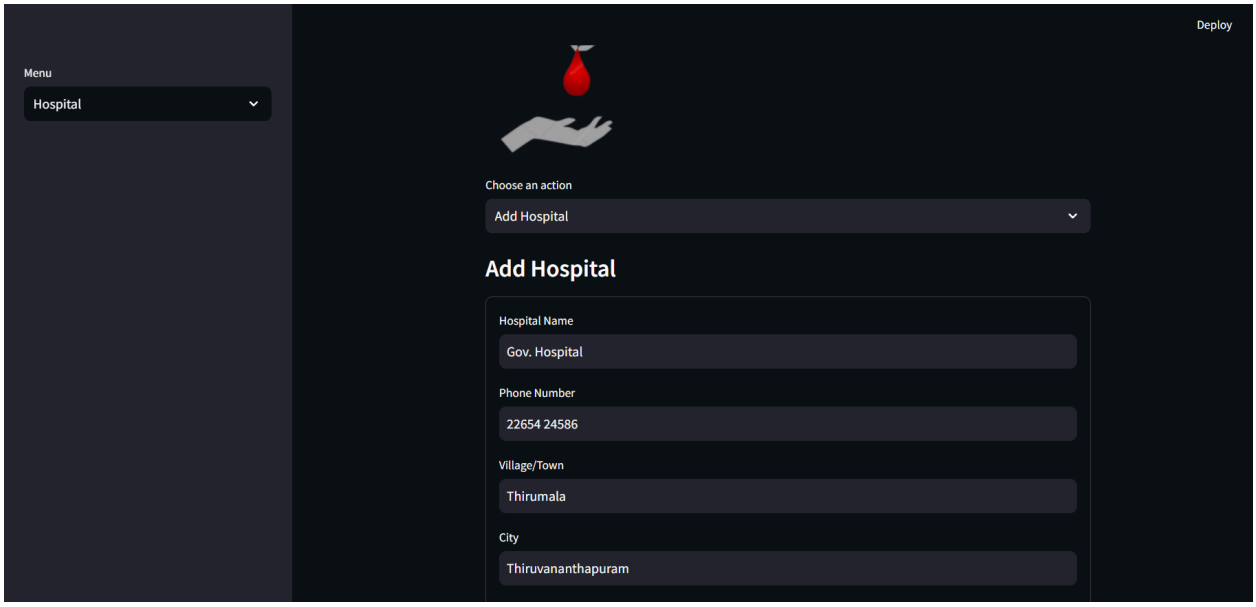
	id	name	phone_number	village_town	city	state	pin	blood_group
0	1	John Doe	1112223333	Central Park	New York	NY	10001	A+
1	2	Jane Roe	4445556666	Hollywood	Los Angeles	CA	90001	O+
2	3	Mark Twain	7778889999	Lincoln Park	Chicago	IL	60601	B-
3	4	Nirmal	8714206563	Karamana	Ernakulam	Kerala	695033	B-

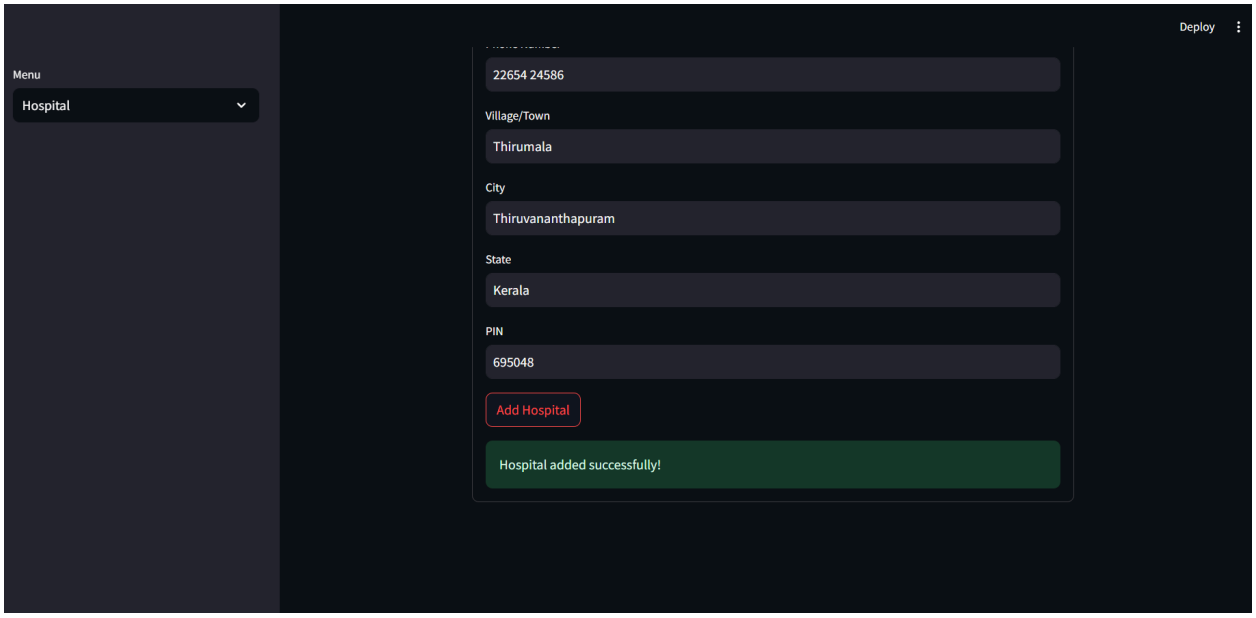
Delete receiver



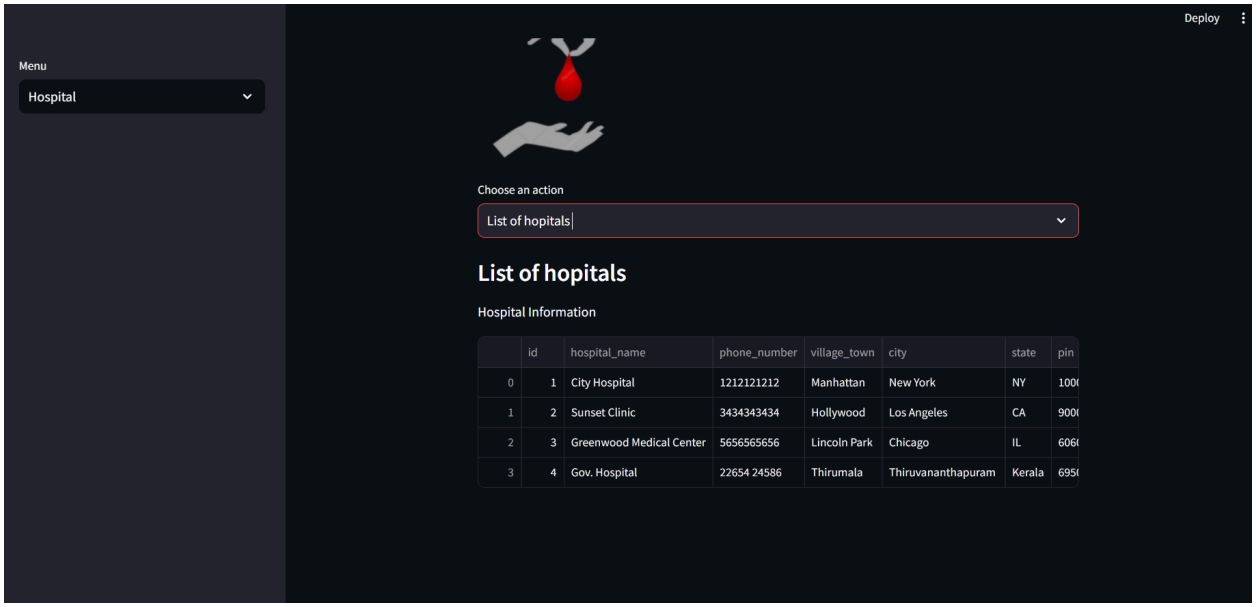
Hospital

Add hospital

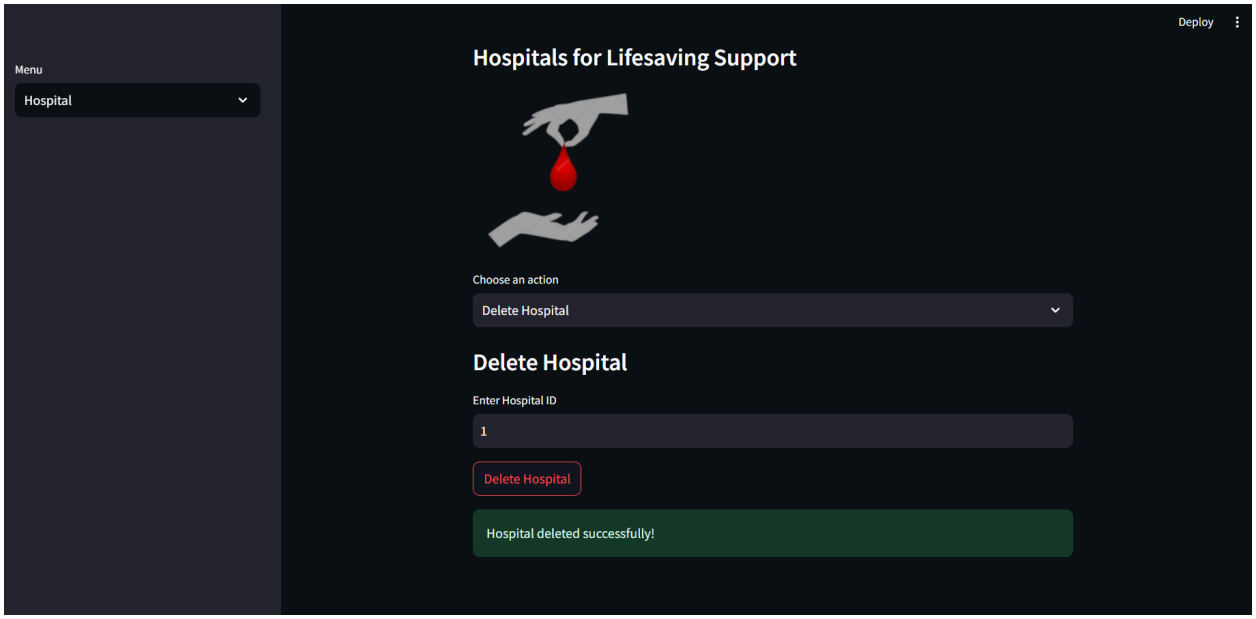




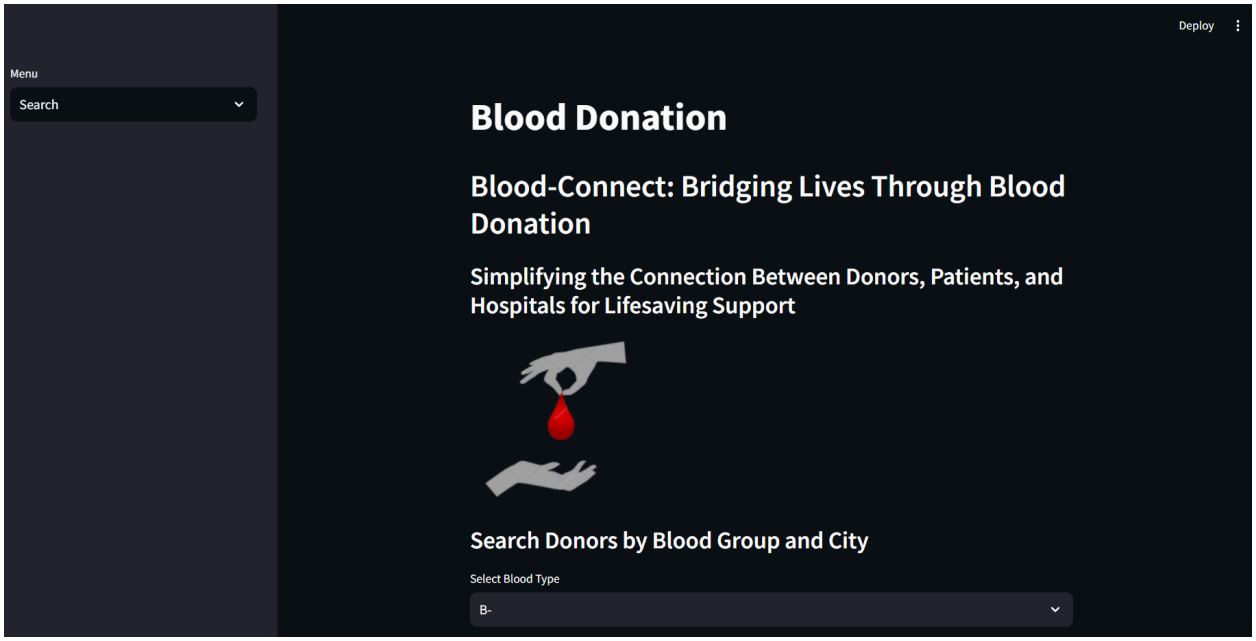
List of hospitals

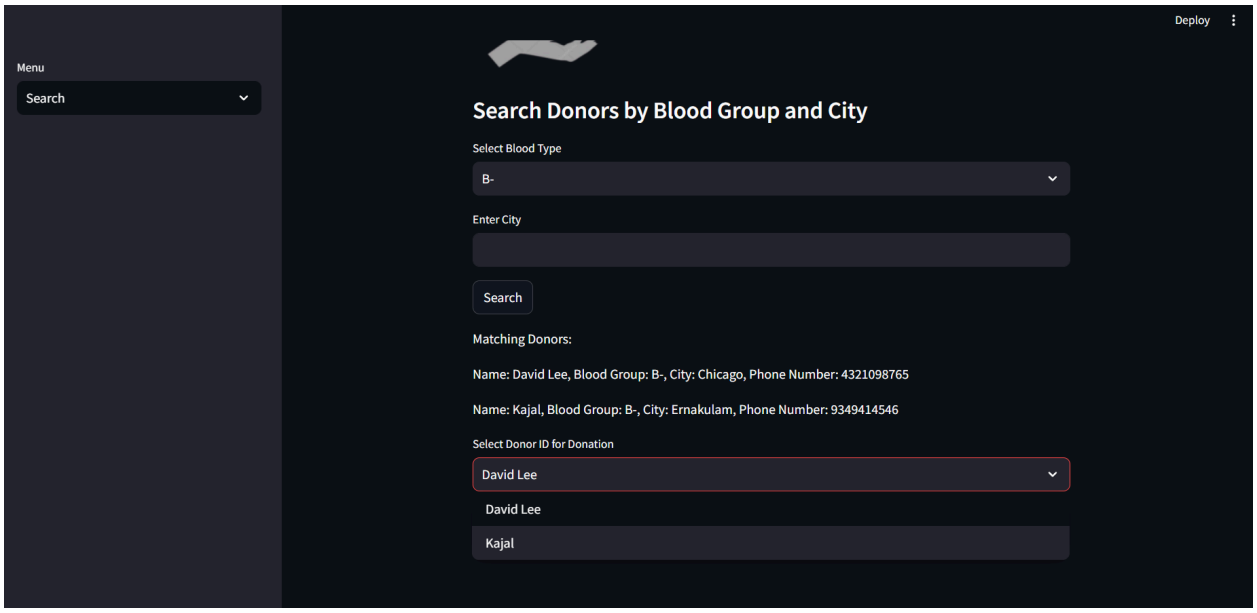


Delete hospital



Search


Deploy
Menu
Search
Blood Donation
Blood-Connect: Bridging Lives Through Blood Donation
Simplifying the Connection Between Donors, Patients, and Hospitals for Lifesaving Support
Search Donors by Blood Group and City
Select Blood Type
B-


Deploy
Menu
Search
Search Donors by Blood Group and City
Select Blood Type
B-
Enter City
Search
Matching Donors:
Name: David Lee, Blood Group: B-, City: Chicago, Phone Number: 4321098765
Name: Kajal, Blood Group: B-, City: Ernakulam, Phone Number: 9349414546
Select Donor ID for Donation
David Lee
David Lee
Kajal

Conclusion

Blood-Connect demonstrates the transformative potential of integrating technology with social good. This application revolutionizes the traditional blood donation process by simplifying connections between donors, receivers, and hospitals. Its user-friendly interface, real-time data access, and advanced features like machine learning ensure timely and accurate assistance during emergencies. Moreover, the scalable architecture of Blood-Connect enables the integration of future enhancements, such as automated scheduling and notification systems, making it a forward-thinking solution.

By fostering a sense of community and promoting regular participation in blood donation activities, Blood-Connect not only addresses the immediate logistical challenges but also encourages a culture of collective responsibility. As a robust and efficient platform, it paves the way for a new era in blood donation systems, ultimately saving countless lives and reinforcing the importance of community-driven efforts.

References

Python Libraries:

- Pandas,
- Streamlit,
- Scikit-learn,
- geopy,
- pymysql

Online Tutorials:

- Streamlit documentation,
- MySQL guides
- Knn documentations

Tools:

- Streamlit for implementing
- Xampp for SQL database
- VS code for source code

Books:

- "Database System Concepts" by Abraham Silberschatz, Henry F. Korth, and S. Sudarshan (for database design and management).
- "Getting started with Streamlit for Data Science - Create, deploy, and test your Python applications, analyses, and models with ease using Streamlit " by Tyler (for implementation)
- "Hands on machine learning with scikit-learn, keras and tensorflow" by oreilly publications (for knn)

Online Resources:

- Stack Overflow,
- GitHub repositories
- Greeks for Greeks