

Abhigyan Madhukalya | Computer Science | November 10, 2023

Computer Science Project

Blood bank management

## **Overview**

The Blood Bank Application is designed to facilitate blood donation activities, allowing users to donate blood and view a list of blood donors based on different criteria. The application is implemented in Python and uses a MySQL database to store donor information.

## **file structure**

* app.py: Main application script responsible for creating the database, tables, and handling user interactions.
* models.py: Defines functions for creating the database and tables.
* bloodbank\_functions.py: Contains functions for blood donation actions and viewing donor lists.
* validate.py: Provides validation functions for user input.

## **Database schema**

The application uses MySQL database with the following schema:

*# Create donors table*

        cursor.execute(

            """

            CREATE TABLE IF NOT EXISTS donors (

        id INT AUTO\_INCREMENT PRIMARY KEY,

                name VARCHAR(255) NOT NULL,

                blood\_type VARCHAR(3) NOT NULL,

                contact\_number VARCHAR(15),

                donor\_id INT UNIQUE

            )

            """

        )

* `id`(INT): Primary key for the donor record.
* `donor\_id`(INT): Unique identifier for each donor.
* `name`(VARCHAR): Name of donor.
* `blood\_type`(VARCHAR): Blood type of the donor.
* `contact\_number`(VARCHAR): Contact number of the donor.

## **Code documentation**

1. **app.py**

* Main Functionality:
  + Initialises the database and tables.
  + Handles user interactions using the ‘inquirer’ library.
* Usage

Run the command `python app.py` in shell.

1. **models.py**

* create\_database function:
  + Creates the database if it doesn't exist.

def *create\_database*():

    try:

        conn = mysql.connector.connect(*host*=DB\_HOST, *user*=DB\_USER, *password*=DB\_PASSWORD)

        cursor = conn.cursor()

*# Create the database if it doesn't exist*

        cursor.execute(*f*"CREATE DATABASE IF NOT EXISTS {DB\_NAME}")

        conn.commit()  *# Commit the changes*

        cursor.close()  *# Close the cursor*

        conn.close()  *# Close the connection*

    except mysql.connector.Error as e:

*print*(*f*"Something went wrong during database creation: {*str*(e)}")

* create\_tables function:
  + Creates the 'donors' table if it doesn't exist.

def *create\_tables*():

    try:

        conn = mysql.connector.connect(

*host*=DB\_HOST, *user*=DB\_USER, *password*=DB\_PASSWORD, *database*=DB\_NAME

        )

        cursor = conn.cursor()

*# Create donors table*

        cursor.execute(

            """

            CREATE TABLE IF NOT EXISTS donors (

        id INT AUTO\_INCREMENT PRIMARY KEY,

                name VARCHAR(255) NOT NULL,

                blood\_type VARCHAR(3) NOT NULL,

                contact\_number VARCHAR(15),

                donor\_id INT UNIQUE

            )

            """

        )

        conn.commit()  *# Commit the changes*

        cursor.close()  *# Close the cursor*

        conn.close()  *# Close the connection*

    except mysql.connector.Error as e:

*print*(*f*"Something went wrong during DB initialization: {*str*(e)}")

1. **bloodbank\_functions.py**

* perform\_action function:
  + Executes user-selected actions such as donating blood or viewing donor lists.
* donate\_blood function:
  + Adds a new donor record to the database.
* view\_entire\_list function:
  + Displays the entire list of donors.
* view\_by\_name function:
  + Displays a list of donors based on the provided name.
* view\_by\_blood\_type function:
  + Displays a list of donors based on the provided blood type.

1. **validate.py**

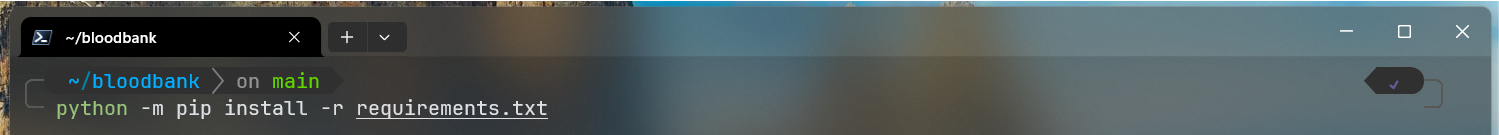
* validate\_name function:
  + Validates the format of the provided name.
* validate\_contact\_number function:
  + Validates the format of the provided contact number.
* validate\_blood\_type function:
  + Validates the format of the provided blood type.

## Dependencies

1. **mysql-connector-python:** Used for connecting to the MySQL database.
2. **inquirer:** Enables interactive command-line user interface.
3. **rich:** Provides rich formatting for terminal output.

## setup

1. Install dependencies.



1. Setup environmental variables (mariadb\_username, mariadb\_password)
2. Run the application.

