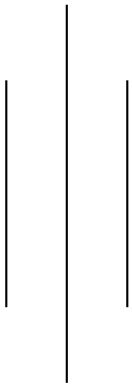
TRIBHUWAN UNIVERSITY
INSTITUTE OF ENGINEERING (IOE)

PULCHOWK CAMPUS

A Report on



Bloodbank Management System

Submitted By:

Aditi Kharel 077BEI008
Asmita Sigdel 077BEI013
Aviyanshu Adhikari 077BEI014

Submitted To:

Department of
Electronics and Computer
Engineering

Abstract

This document describes the project Blood Bank Management System developed as a course project for Database Management System. This document contains details about the tools used and the codes written to implement the management system. We have used Django web framework and PostgreSQL for databases.

Contents

1 Introduction

Database Management System is a software system that manages databases. It is used to store, retrieve, and run queries on data. A DBMS serves as an interface between an end-user and a database, allowing users to create, read, update, and delete data in the database. It manages the data, and the database schema, allowing for data to be manipulated or extracted by users and other programs. It offers numerous advantages, such as data integrity, security, concurrent access, and data abstraction.

- **Data:** DBMS stores data (text, numbers, images, videos) in a structured manner.
- **Database:** A database of a structured collection of data. It consists of tables, each of which holds data about a specific entity.
- **Table:** A table is a fundamental component of a database. Each row in a table represents a single record or entry, and each column represents an attribute of that record.
- **Schema:** A schema defines the structure of the database, including the tables, their columns, data types, and relationships between tables. It acts as a blueprint for the database.

These features are demonstrated through our project **VitalCare**- a blood donation management system.

2 Objectives

- To understand the core concepts of database management and apply them.
- To build a useful system that addresses the problems present with current blood management systems.
- To optimize the data management process.
- To ensure security and privacy of the website users

3 Problem Statement

There are a lot of blood donation management systems in Nepal. They provide services like provision of blood bank information, forms for requesting blood and donating blood, and information of blood donation programs happening around. However, they fail to collect the medical information of blood donors through the platform itself. The inquiry

of medical history is done later which can cost time and sometimes even the health of the recipient. We believe that this should be a crucial information that should be taken into consideration beforehand. We aspire to include all these important features in a single system.

4 Overview of Project

VitalCare is a blood donation management system that keeps track of blood donors, receivers, and blood banks. It mainly aims to provide a better system for blood donation campaigns and programs. Created with the aim of simplifying the navigation of numerous blood bank systems and providing insights into available management solutions, this project is designed to enhance various facets of blood banking. It can streamline tasks related to donor and patient management,and uphold the precision and security of data. Blood Bank Management System is an indispensable tool in the healthcare industry, promoting blood safety, efficient resource utilization, and the overall well-being of patients and donors alike. Its significance lies in its ability to bridge the gap between donors and recipients, making life-saving blood transfusions accessible and secure.

5 System Design

5.1 Requirements

5.1.1 PostgreSQL

Often referred to as "Postgres", it stands as one of the most renowned open-source relational database management systems. It is a trusted choice as a backend database of dynamic websites and web applications, predominantly in the realm of web development. Due to its wide use in the industry and the features it provides, we chose PostgreSQL over other SQL tools.

- It can efficiently handle large datasets and concurrent users, making it suitable for our application.
- It is fully ACID (Atomicity, Consistency, Isolation, Durability) compliant, ensuring data consistency and integrity, which is crucial for applications involving sensitive data like our project.
- It includes robust security features, including role-based access control, SLL support, and data encryption. These features are essential for protecting sensitive medical data.

It is necessary to analyse the required tables for the system and they are:

	table_name name
1	events
2	person
3	blood_bank
4	blood
5	bank_post
6	surgeries
7	immunizations
8	diseases
9	relationship
10	reception

Figure 1: Tables required for the project

5.1.2 Django

Django, a high-level Python web framework, simplifies web development by providing a well-structured, reusable, and maintainable codebase. It excels in handling both the backend and frontend aspects of web application development.

- Its ORM (Object Relational Mapping) abstracts the database layer by allowing us to define data models as Python classes.
- It includes a range of built-in features and modules for authentication, URL routing, and form handling.
- It includes built-in protection against common web vulnerabilities. This commitment to security is crucial when handling sensitive medical data.
- It provides a comprehensive authentication system that simplifies user registration, login, and password management.
- It includes a powerful template engine that separates HTML from Python code, enhancing maintainability.

5.2 Data Flow

Donor and Receiver data, blood banks data, profile of users, and their medical information are organized within the PostgreSQL database. The Django backend facilitates

data manipulation and retrieval through Python-based models, views, and controllers, ensuring data consistency and integrity. User interactions and data updates are carefully managed to maintain the integrity of the database. The connection of data between PostgreSQL and Django is done through migrations. The commands are:

```
py manage.py makemigrations
py manage.py migrate
```

5.3 Development Environment

To ensure a controlled and consistent development environment, we've utilized Python's 'venv' module to create virtual environments. This approach isolates our project's dependencies, promoting a well-structured development workflow.

- Creating a Virtual Environment: We've established a virtual environment within our project directory using the venv module.
- Activating the Virtual Environment: We've activated the virtual environment based on the operating system.
- Installing Project Dependencies: We've used pip to install the required project dependencies from the requirements.txt file while the virtual environment was active.

We have used pip freeze for our requirements.txt so as to avoid errors while collaborating which could be caused due to different dependencies' versions.

```
pip freeze> requirements.txt
```

The required pac:

```
asgiref==3.7.2
crispy-bootstrap4==2022.1
Django==4.2.4
django-bootstrap==0.2.4
django-crispy-bootstrap==0.1.1.1
django-crispy-forms==2.0
django-environ==0.10.0
packaging==23.1
Pillow==10.0.0
psycopg2==2.9.7
sqlparse==0.4.4
tzdata==2023.3
```

5.4 Database Design (ER Diagram)

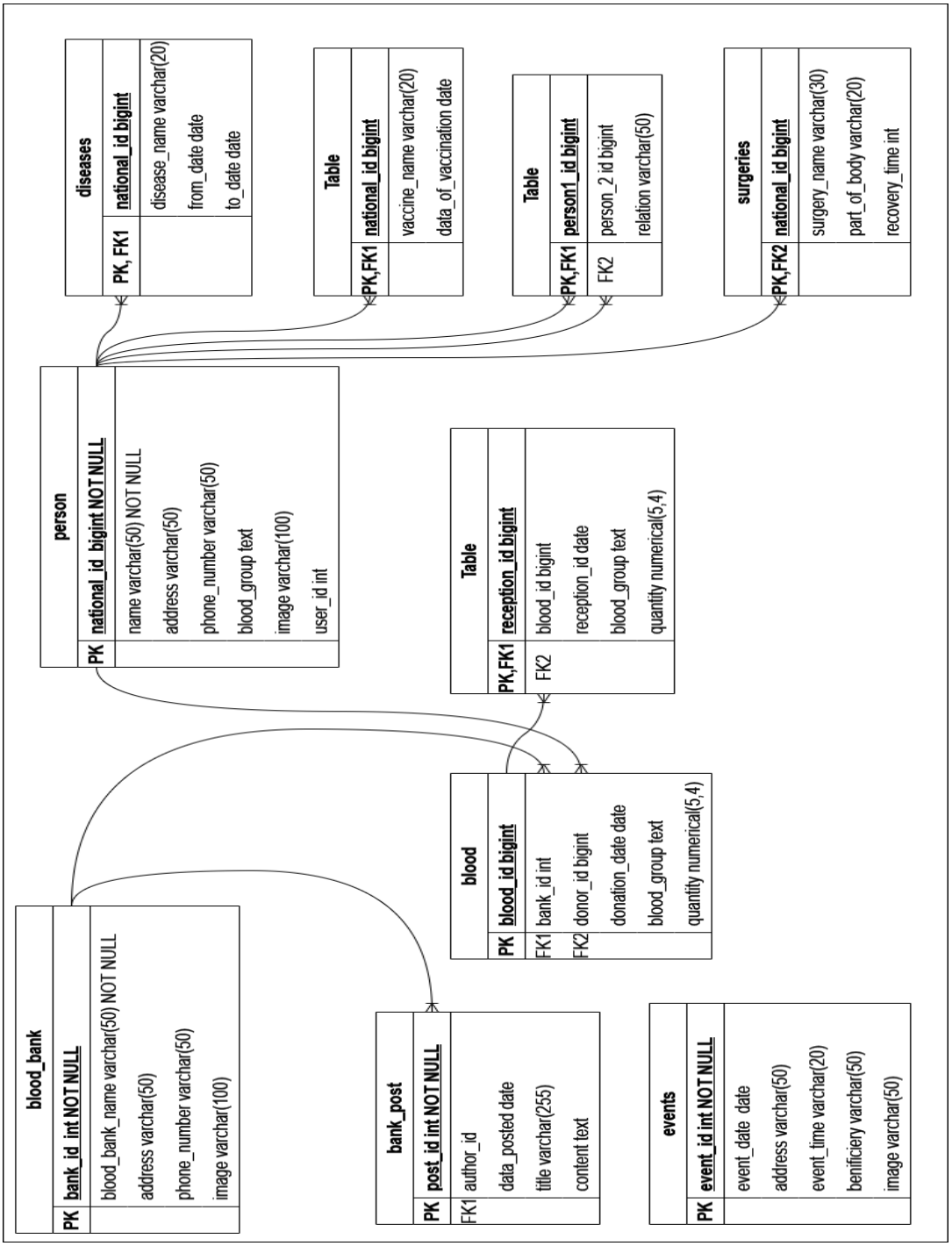


Figure 2: ER Diagram

5.5 System Architecture

In our Blood Bank Management System, we have adopted a modern and efficient architectural design that leverages the capabilities of Django as a full-stack web framework and PostgreSQL as the backend database management system. This architecture is engineered to ensure the seamless flow of data and user interactions while maintaining security and performance.

- User interface managed by Django's built-in templating engine.
- Django's backend for managing application logic and data processing, including views, models, and URL routing.
- PostgreSQL for housing all our application's data.

5.6 Special elements

1. **Django MVT**:Django's Model-View-Template (MVT) architecture is a robust framework for building web applications with clear separation of concerns. In this pattern, the Model handles data representation and business logic, simplifying database interactions through the Object-Relational Mapping (ORM). The View manages user interactions, processing requests, and serving responses, encapsulating application logic. The Template focuses on rendering the user interface, using dynamic placeholders to display data from the Model. In our project, figure 3 shows examples of models, views and templates.

```python
class Events(models.Model):
    event_id = models.AutoField(primary_key=True)
    event_date = models.CharField(max_length=50, blank=True, null=True)
    address = models.CharField(max_length=50, blank=True, null=True)
    event_time = models.CharField(max_length=50, blank=True, null=True)
    beneficiary = models.CharField(max_length=50,blank=True, null=True)
    image=models.ImageField(upload_to='')
    class Meta:
        db_table = 'events'
```

(a) A model in our project

```python
from django.shortcuts import render
from .models import *
from users.models import *
from django.db.models import Q

#handle routes

def home(request):
    context ={
        'posts': BankPost.objects.all()
    }
    return render(request, 'bank/home.html',context)

```

(b) View for processing request

bank > templates > bank > about.html > ...

```python
{% extends "bank/base.html" %}
{% block content %}
    <h1>Welcome!</h1>
    <p>
        In our commitment to making a positive impact on
        saving lives through the effective management of
        the critical blood resource, we proudly introduce
        this website. It provides access to vital information
        about blood donors and recipients, as well as a
        comprehensive directory of the city's blood banks.
    </p>

```

(c) Template for About page(HTML)

Figure 3: Django MVT

2. **Bootstrap Cards**: The main feature of our application is the information dispension through events posted, and providing blood bank information to the users. These are displayed as cards as seen in figure 4.

Figure 4: Card created using bootstrap

Bootstrap cards are an integral component of the Bootstrap framework, serving as compact containers that hold various types of content, such as text, images, buttons, or even forms. With their responsive design, these cards effortlessly adapt to different screen sizes and devices, maintaining an appealing presentation across the board.In our project, we've used buttons and images. The button color has been specified by

```
class= "btn btn-danger"
```

3. **Cripsy forms**: With Crispy Forms, we can define our forms using Django's Form classes as usual, but we gain the ability to easily control the form's layout, styling, and presentation using template-based configuration. One of the standout features of Crispy Forms is its template tags, which enable you to effortlessly control form rendering through Django templates. This is why we've chosen Crispy Forms for seamless integration with Django.

```
{% extends "bank/base.html" %}
{% load crispy_forms_tags %}
```

Figure 5: Template tag feature of Crispy Forms

6 User Interface/User Experience

The figure 6 shows the user interface of our website. This is the bloodbank page which lists a number of blood banks one can get in contact with for emergency. Although most websites only list them we've also provided the website link for ease of access. This significantly improves the UI/UX aspect of our project.

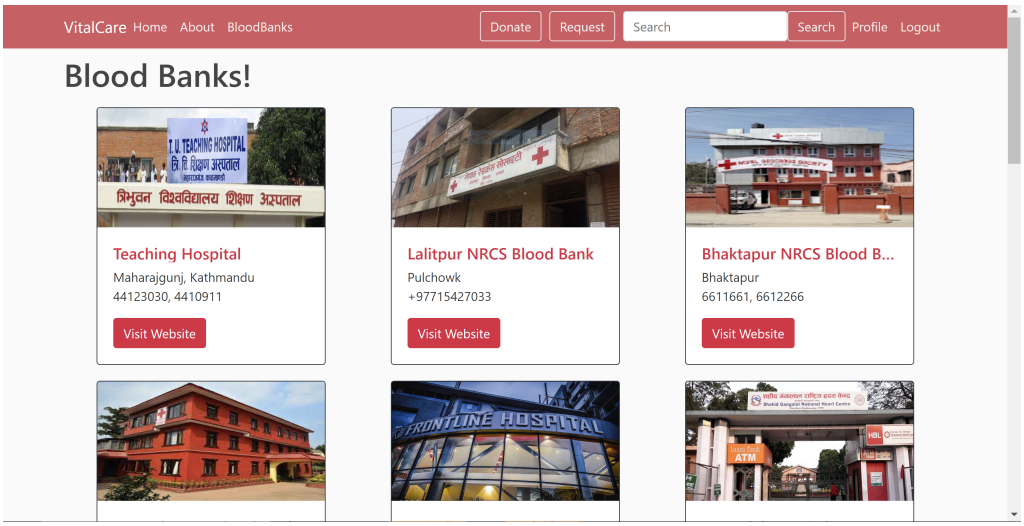


Figure 6: UI/UX of VitalCare

7 Key Features

1. Donor Management
   - Registration of blood donors with personal and medical information.
   - Blood type compatibility screening for donors.
   - Donor eligibility checks.
2. Recipient Management
   - Registration of patients and their medical history.
   - Blood type matching and compatibility checks.
   - Prioritization of emergency requests.
3. Donor Database
   - Maintain a centralized donor database with contact information.

- Streamline communication and engagement with donors.
- Facilitate personalized donor interactions and targeted campaigns.

4. Blog Posts on Blood Donation
    - Publish informative blog posts on blood donation and health tips.
    - Raise awareness and educate the community about blood donation.
    - Dispel myths, share success stories, and encourage regular donations.
5. Search feature
    - Search the entire website
    - Filter the events based on your preferences
    - Search for informative blogs.

8 Limitations

We managed to include the crucial features in the project, but there were still some limitations that need to be addressed.

1. Although the medical data has been collected which made it easier for selection of valid donors, we couldn't develop a feature to automatically filter them due to the time limitation.
2. A lot of the tables in the database couldn't be normalised more for optimization.
3. Role based login and permission handling is currently unavailable. We made the website with the presumption that we are acting as the Blood Bank/ Hospital admin.

9 Future Enhancements

If we get to work on this project in the future as well, there are some areas that could be improved.

1. Query based filtering could be implemented in the website as well.
2. Elimination duplicate data could be done as much as possible to optimize the database.
3. The user experience of the website can be improved.

4. Role based login could be enabled.

5. Hosting of the website could be done.

10 Result

The result was a fully responsive website seamlessly integrated with the database. The concepts of data migration, insertion of data into database through code, query, as well as crispy forms was studied. Management of the database was done and security was ensured by using `.env` file with security key so credentials could be safe.The implementation of authentication, security and data relations management of DBMS helped us understand the idea behind them.

11 Conclusion

The blood donation management system provides us with a seamless experience of managing blood, donors and receivers. Moreover, while creating the website and database, our team learned the core concepts of database management, working with schemas, Django MVT, and virtual environments. Although all the features that we initially planned could not be completed due to the limited time, we did learn to store and retrieve information from the database and integrate it with the frontend. There is still room for future improvements and we believe that this project could actually come in use if all features can be developed. Thus, Vitalcare was developed considering the use cases of database management

References

[1] Django documentation. https://docs.djangoproject.com/en/4.2/.

[2] er diagram - tool. https://app.diagrams.net/.

[3] freecodecamp - website. https://freecodecamp.org/.

[4] javatpoint django tutorial. https://www.javatpoint.com/django-tutorial.

[5] Postgresql - documentation. https://www.postgresql.org/docs/.

[6] Reference website. https://nrcs.org/.

[7] w3schools - tutorials. https://www.w3schools.com/.