,

Final Project Report

Blood donation management system

PREPARED AND SUBMITTED BY

Mohammed Natour std.no(11923584)
Ahmad Siksik std.no(11923771)
Hisham Abd-Alfatah std.no(11923697)

PROJECT GUIDE

Dr. Adnan Salman

TABLE OF CONTENTS:

SUMMARY:

The Blood Bank System is a web application designed to manage blood donations, blood banks, and blood recipients. This documentation provides an overview of the system's features, setup instructions, and usage guidelines for administrators, users, and developers. It covers user registration, donor and recipient management, search functionality, appointment scheduling, notifications, and reporting. Explore this documentation to learn how to effectively use and customize the Blood Bank System.

INTRODUCTION

Welcome to the official documentation for the Blood Bank System! This documentation serves as a comprehensive guide to understand and utilize the features and functionality of the Blood Bank System. Whether you are an administrator, a user, or a developer, this documentation will provide you with the necessary information to effectively interact with and customize the system according to your requirements.

About the Blood Bank System

The Blood Bank System is a web application designed to streamline and automate the management of blood donations, blood banks, and blood recipients. It aims to simplify the process of blood donation, improve the efficiency of blood bank operations, and facilitate the timely and accurate delivery of blood to those in need.

Key Features

User Registration: Learn how to create user accounts for administrators, donors, and recipients, and manage user roles and permissions.

Donor and Recipient Management: Understand how to manage donor and recipient profiles, track their medical history, and maintain accurate records.

Blood Bank Management: Explore the features related to blood bank management, including inventory management, stock tracking, and blood type compatibility checks.

Search Functionality: Discover how to search for blood donors, recipients, and blood banks based on various criteria such as location, blood type, and availability.

Appointment Scheduling: Learn how to schedule and manage appointments for blood donations and recipient blood requests.

Notifications: Explore the notification system to keep users informed about appointment reminders, blood availability updates, and important system notifications.

Target Audience

This documentation caters to the following audience:

Administrators: Individuals responsible for managing and configuring the Blood Bank System, setting up user accounts, and maintaining system functionality.
Donors and Recipients: Users interested in donating blood or in need of blood transfusions, who will interact with the system to register, schedule appointments, and find suitable matches.
Developers: Individuals interested in customizing or extending the functionality of the Blood Bank System, integrating with external systems, or developing additional features.
Getting Started
To get started with the Blood Bank System, refer to the installation guide for instructions on setting up the application environment. Once the system is installed and running, follow the respective user guides for administrators, donors, and recipients to understand how to effectively use the system.

We hope this documentation provides you with the necessary guidance to leverage the Blood Bank System to its fullest potential. If you have any questions or need further assistance, please refer to the contact information provided at the end of this document.

Let's begin the journey of efficiently managing blood donations and saving lives with the Blood Bank System!

FRAMEWORKS

Django

Django is a high-level Python web framework that follows the model-view-controller (MVC) architectural pattern. It provides a robust set of tools and features for building web applications quickly and efficiently. Django offers built-in support for handling routing, database management, user authentication, and more.

The Bloodbank project utilizes Django as the backend framework to handle the server-side logic, database integration, and REST API development. With Django, we can take advantage of its powerful features to implement the blood bank functionality, manage user authentication, and provide a seamless API for frontend communication.

React

React is a JavaScript library for building user interfaces. It follows a component-based approach, where UI components are built and composed to create complex user interfaces. React provides a declarative syntax, efficient rendering, and a virtual DOM (Document Object Model) for efficient updates to the UI.

In the Bloodbank project, React is employed as the frontend framework to develop a dynamic and interactive user interface. With React, we can build reusable UI components, manage state effectively, and create a responsive and engaging experience for the blood bank application users.

DATABASE

PostgreSQL

PostgreSQL is a powerful open-source relational database management system (RDBMS) known for its reliability, performance, and extensive support for advanced database functionalities. It offers a wide range of features, including support for complex queries, ACID (Atomicity, Consistency, Isolation, Durability) transactions, and data integrity.

The Bloodbank project utilizes PostgreSQL as the database management system to store and retrieve data. PostgreSQL provides a solid foundation for managing the application's data, ensuring data consistency, and enabling efficient querying and manipulation of data. With PostgreSQL, we can leverage its advanced features to handle the blood bank application's data storage needs effectively.

By combining the Django backend framework, React frontend framework, and PostgreSQL database, the Bloodbank project achieves a powerful and scalable web application. Django handles the backend logic, database integration, and REST API development, while React handles the frontend user interface and interactivity. PostgreSQL serves as the reliable and robust database backend, storing and retrieving data efficiently.

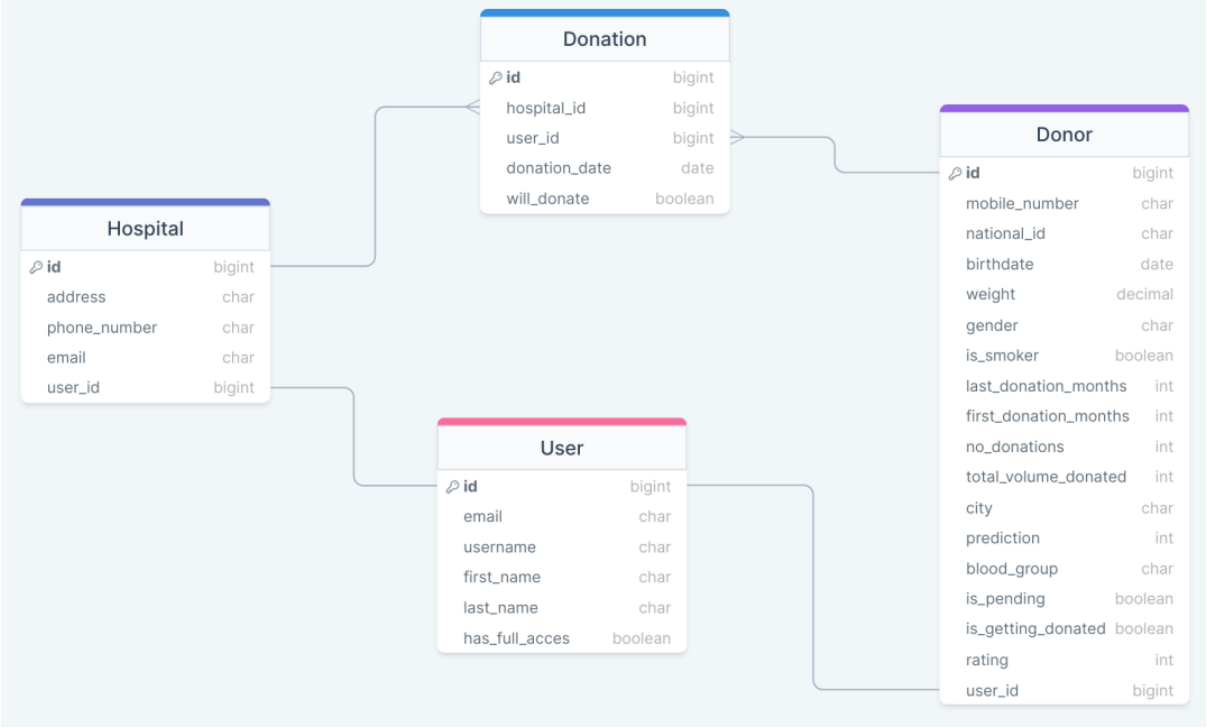
This technology stack offers a solid foundation for building a feature-rich and high-performing blood bank application, allowing us to leverage the strengths of each framework and the database to create a seamless user experience.

DATABASE DESIGN

The Bloodbank project uses PostgreSQL as the database for storing application data. The database design is based on the following Django models:

Entity-Relationship (ER) Diagram

Here's an ER diagram representing the database structure of the Bloodbank project:



Model Description

CustomUser: Represents a user in the application, with fields such as username, email, first_name, last_name, and access-related flags.
Donor: Represents a blood donor, associated with a CustomUser. Includes fields like mobile_number, national_id, birthdate, weight, gender, and donation-related information.
Hospital: Represents a hospital, associated with a CustomUser. Includes fields like address, phone_number, and email.
Donation: Represents a blood donation event, involving a Hospital and a Donor. Includes fields like donation_date and will_donate.
The ER diagram visually illustrates the relationships between these entities, their attributes, and cardinalities. It provides a clear overview of the database schema and serves as a reference for understanding the data structure and connections within the Bloodbank application.

Please note that the diagram is a representation of the database design and may not include all the intricate details of field constraints, indexes, or additional relationships. For a comprehensive understanding of the database structure, it is recommended to review the models and their associated migrations in the project.

DATABASE CONFIGURATION

The Blood Bank project uses a relational database to store and manage its data. Django provides support for various database backends, including PostgreSQL, MySQL, SQLite, and Oracle. In this section, we will cover the configuration of the database backend for the project.

Database Backend Selection

To configure the database backend, open the settings.py file located in the bloodbank directory. Look for the DATABASES setting, which defines the database connection settings. By default, the project is configured to use the SQLite database backend. If you prefer to use a different database backend, you can modify the settings accordingly. For example, to use PostgreSQL, you would change the following settings:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'bloodbank',
```

```
        'USER': 'mohammed',
        'PASSWORD': 'mohammed',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

Migrations

Django provides a built-in migration system that manages changes to the database schema over time. To apply the initial migrations and create the necessary database tables, run the following command in the terminal:

(venv) $ python manage.py migrate

This will synchronize the database with the current state of the project's models.

Creating Superuser

The superuser is an administrative user with full access to the Django admin interface. To create a superuser, use the following command:

(venv) $ python manage.py createsuperuser

Follow the prompts to provide a username and password for the superuser.

Database Operations

With the database configured, you can perform various database operations using Django's ORM (Object-Relational Mapping) and query API. You can define models in the models.py file, representing database tables, and use the provided API to interact with the data.

For example, to retrieve all donors from the database, you can use the following code snippet:

```
from bank.models import Donor

def get_all_donors():
    donors = Donor.objects.all()
    return donors
```

Refer to Django's official documentation for more details on working with models and performing database operations.

You have successfully configured the database backend for the Blood Bank project. Now you can use the power of Django's ORM to interact with the database and manage the project's data effectively.

BACKEND DEVELOPMENT

The backend of the Blood Bank project is responsible for handling the server-side logic, database interactions, and API endpoints. It is developed using Django, a popular Python web framework. In this section, we will cover the main aspects of backend development.

Project Structure
The backend codebase is organized as follows:

bloodbank/: The root directory of the Django project.
bank/: The main Django app for the Blood Bank project.
migrations/: Contains database migration files.
models.py: Defines the database models for the project.
views.py: Contains the view functions for handling API requests.
urls.py: Defines the API endpoints and their corresponding view functions.
bloodbank/: The project's settings and configuration directory.
settings.py: Contains the project settings, including database configuration.
manage.py: The Django project management script.

API Endpoints
The Blood Bank backend exposes several API endpoints to interact with the system. Here is an overview of the available endpoints:

Donor Endpoints:

/donor/: GET (List all donors)
/donor/<int:donor_id>/: GET (Retrieve a specific donor)
/donor/<int:donor_id>/put/: PUT (Update a specific donor)
/donor/<int:donor_id>/patch/: PATCH (Partial update of a specific donor)
/donor/signup/: POST (Create a new donor account)
/donor/login/: POST (Authenticate and login a donor)
/donor/calculate_age/: POST (Calculate the age of a donor)

/donor/prediction/: POST (Make a prediction using the machine learning model)

Hospital Endpoints:

/hospital/login/: POST (Authenticate and login a hospital)
/hospital/signup/: POST (Create a new hospital account)
/hospital/: GET (List all hospitals)
/hospital/<int:hospital_id>/: GET (Retrieve a specific hospital)

Donation Endpoints:

/donations/: GET (List all donations)
/donations/<int:donation_id>/: GET (Retrieve a specific donation)
/donations/create/: POST (Create a new donation)
/donations/<int:donation_id>/update/: PUT (Update a specific donation)
/donations/<int:donation_id>/delete/: DELETE (Delete a specific donation)

These endpoints provide the necessary functionality to interact with donors, hospitals, and donations in the Blood Bank system.

Business Logic and Data Validation
The backend implements the business logic of the Blood Bank system, including data validation, processing user requests, and performing database operations. It enforces the necessary constraints and rules to ensure the integrity and consistency of the system's data.

Error Handling and Response Formats
The backend handles errors gracefully and provides meaningful responses to clients. In case of validation errors or other exceptions, appropriate error messages and status codes are returned. The responses follow standard HTTP response formats and conventions.

Congratulations! You now have an understanding of the backend development aspects of the Blood Bank project.

FRONTEND DEVELOPMENT :

The frontend development phase of your graduation project involves the creation of a website for blood donation purposes. This website will enable donors to create accounts and hospitals to schedule appointments. The goal is to develop a user-friendly platform that streamlines the blood donation process and enhances user engagement.

Technologies and Frameworks

To accomplish the frontend development, we will employ the following technologies and frameworks:

- React.js: We will utilize React.js, a JavaScript library known for its component-based architecture, to construct the user interface. React.js provides an efficient way to build reusable UI components and manage their states effectively.

- Next.js: Next.js, a framework built on top of React.js, will be employed to leverage its server-side rendering capabilities and performance optimizations. It simplifies the setup process and offers features such as automatic code splitting and routing, which will greatly benefit the project.

- Redux: Redux, a state management library, will be integrated to handle the global state of the application. It provides a predictable and centralized approach to state management, which will prove valuable for managing user authentication, donor profiles, and appointment scheduling.

- React Router: React Router, a routing library for React applications, will enable us to manage client-side routing. It will facilitate the definition of different routes for login, account creation, donor profiles, and appointment lists.

Project Structure

To ensure a structured and maintainable codebase, we will adhere to a recommended project structure for the frontend development phase:

- components/ (Contains reusable UI components)
- pages/ (Defines different pages/routes of the application)
- store/ (Includes Redux store configuration and actions)
- styles/ (Manages stylesheets and CSS files)

This structure promotes code modularity and reusability, enhancing the overall maintainability of the project.

User Interface Design

A critical aspect of frontend development is the creation of an intuitive and visually appealing user interface. We will prioritize the design of a user-friendly interface with clear navigation, well-organized layouts, and consistent styling.

To achieve responsiveness and streamline the styling process, we will leverage modern CSS frameworks such as Bootstrap or Tailwind CSS. Custom CSS will also be used to accommodate any unique design requirements specific to the project.

Conclusion

During the frontend development phase of your graduation project, we will employ React.js, Next.js, Redux, and React Router to create an efficient and user-friendly blood donation website. By adhering to the recommended project structure and focusing on user interface design principles, we aim to deliver a well-organized codebase and a visually appealing interface.

MACHINE LEARNING

The Blood Bank project incorporates machine learning techniques to make predictions and enhance its functionality. Specifically, it utilizes a machine learning model to predict the likelihood of future blood donations from donors. In this section, we will explore the machine learning aspect of the project.

Machine Learning Model

The machine learning model used in the project is trained on historical donor data to predict the probability of future blood donations. The model is trained using a supervised learning algorithm, and the features used for prediction have factors such as months since first donation, months since first donation, number of donations and total volume donated.

Model Training

The model training process involves preparing the training dataset, selecting appropriate features, and choosing the machine learning algorithm. This process is typically performed offline using a separate script or Jupyter notebook.

Saving the Trained Model

Once the model is trained, it can be serialized and saved as a file for later use. In the Blood Bank project, the trained model is saved as a model.pkl file.

Integration with the Backend

To integrate the machine learning model with the backend, a Python script named predict.py is included in the project. This script contains the necessary code to load the trained model and make predictions based on incoming data.

Prediction Endpoint

The Blood Bank backend exposes a prediction endpoint (/donor/prediction/) that allows clients to make predictions using the trained machine learning model. When a POST request is sent to this endpoint with the required data, the backend uses the predict.py script to process the data and return the predicted outcome.

Example usage:

POST /donor/prediction/

```
{
  "months_since_first_donation": 50,
  "months_since_last_donation": 2,
  "number_of_donations": 8,
  “total_volume_donated”:
}
```

The response from the prediction endpoint contains the predicted probability or any other relevant information based on the machine learning model's output.

Model Updates and Retraining

As the Blood Bank project evolves and new data becomes available, it may be necessary to update and retrain the machine learning model periodically. This process involves collecting new data, retraining the model, and replacing the existing model.pkl file with the updated version.

By incorporating machine learning, the Blood Bank project enhances its capabilities to predict blood donation probabilities, allowing for better management and planning of blood resources.

You have learned about the machine learning component of the Blood Bank project. The integration of machine learning techniques adds valuable predictive capabilities to the system, enabling more efficient blood resource management and decision-making.

CONCLUSION:

In conclusion, the Blood Bank project is an innovative and user-friendly web application designed to streamline blood donation management and bridge the gap between donors and hospitals. Through its intuitive interface, secure authentication system, and robust backend implementation, the project provides a comprehensive solution for efficient blood bank operations.

Throughout this documentation, we have covered the installation and setup process for both the frontend and backend components of the project. We discussed the project structure, database configuration, and API endpoints, providing developers with a solid foundation for understanding and contributing to the project.

One of the key highlights of the Blood Bank project is the integration of machine learning capabilities. By leveraging advanced algorithms, the project incorporates a prediction model that estimates the likelihood of successful blood donations based on donor information. This predictive analysis not only helps blood banks optimize their operations but also empowers hospitals with valuable insights to better manage their resources and provide timely assistance to patients in need.

The Blood Bank project demonstrates the power of Django as a robust web framework for developing scalable applications. Its modular architecture and extensive set of built-in features make it an ideal choice for handling complex data structures, implementing secure authentication, and creating efficient API endpoints.

By utilizing Django's ORM (Object-Relational Mapping), developers can easily interact with the database, perform data validations, and ensure data integrity. The project also incorporates best practices for error handling, providing informative responses and appropriate status codes to clients.

We believe that the Blood Bank project has the potential to make a significant impact in the blood donation ecosystem. Its user-centric design, seamless integration of machine learning, and solid backend infrastructure create a reliable platform that facilitates the process of blood donation and contributes to saving lives.

We hope that this documentation has provided you with a comprehensive understanding of the Blood Bank project's development and functionality. Whether you are a developer looking to contribute to the project or a user seeking to utilize its features, we encourage you to explore and engage with the application.

Thank you for choosing the Blood Bank project, and we appreciate your involvement in making a positive difference in the lives of those in need of blood transfusions.