**Blood Bank Inventory and Donor Management Applicationn**

### INTRODUCTION

**1.1.** **Abstract**

The Blood Bank Management System is a web-based application developed using Python’s Flask framework. It aims to streamline the process of blood donation and request management by automating tasks such as donor registration, blood packet tracking, user authentication, and notification handling. The system ensures efficient storage, donation, and availability tracking of blood inventory, enhancing the responsiveness of the blood bank to meet urgent medical needs.

**1.2. Motivation**

The primary motive behind developing the Blood Bank Management System is to streamline and automate the critical processes involved in blood donation, inventory management, and request handling. In many hospitals and clinics, these tasks are still carried out manually, leading to delays, data inconsistencies, and difficulty in accessing real-time information during emergencies. This project aims to eliminate such inefficiencies by providing a user-friendly web platform where donors can be registered, blood inventory can be updated, and urgent requests can be managed swiftly. By digitizing these processes, the system enhances the reliability and responsiveness of blood banks, ultimately helping save lives through timely availability of blood units.

1. **PROJECT DESCRIPTION AND GOALS**
   1. **Survey on Existing System**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sn** | **Paper Name** | **Remarks** | **Year** | **Reference** |
| 1. | A systematic review and research perspective on recommender systems | This research paper presents a systematic review of recent advancements in recommender systems, focusing on applications like books, movies, and products. Despite the precision of current systems, scalability, cold-start, and sparsity challenges persist. The study analyzes applications, conducts algorithmic assessments, and establishes a taxonomy for effective recommender systems. Evaluation criteria include datasets, simulation platforms, and performance metrics, providing a concise overview of the field's current state, highlighting gaps and challenges for future development. | 2022 | Roy, D., Dutta, M. A systematic review and research perspective on recommender systems. *J Big Data* **9**, 59 (2022). https://doi.org/10.1186/s40537-022-00592-5 |
| 2. | Prioritizing tasks in software development: A systematic literature review | This work focuses on task prioritization in software development, aiming to identify effective tools and techniques. Conducting a systematic literature review following the PRISMA statement, the study reveals that most existing approaches concentrate on bug prioritization. Notably, recent works highlight the significance of "pull request prioritization" and "issue prioritization," reflecting the growing influence of version control and issue management software systems. The study also notes common metrics for evaluating prioritization models, such as f-score, precision, recall, and accuracy. This research provides valuable insights for IT practitioners, offering a comprehensive overview of the current state of task prioritization in the Software Engineering domain. | 2024 | Bugayenko Y, Bakare A, Cheverda A, Farina M, Kruglov A, Plaksin Y, et al. (2024) Prioritizing tasks in software development: A systematic literature review. PLoS ONE 18(4): e0283838. https://doi.org/10.1371/journal.pone.0283838 |
| 3. | Prioritizing Use Cases: A Systematic Literature Review | This research highlights the importance of use-case-based prioritization in software development. A systematic literature review of 40 approaches over the past two decades reveals a focus on user-centric requirements in areas like IoT and mobile development. Notably, only 32.5% considered scenario-based prioritization, and the majority were semiformally developed (53.8%). The findings underscore the need for new approaches addressing gaps in strategic goal inclusion and criteria like risks and quality-related requirements, providing insights for practitioners and researchers aiming to enhance prioritization practices. | 2023 | Odeh, Y.; Al-Saiyd, N. Prioritizing Use Cases: A Systematic Literature Review. *Computers* **2023**, *12*, 136. https://doi.org/10.3390/computers12070136 |
| 4. | Web Scraping as a Data Collection Strategy: The Perils and Pitfalls | This article explores the integration of artificial intelligence, specifically web scraping, into discourse analysis in the social work domain. Focusing on a study of blogging discourse on Black women's mental health during the dual pandemics of COVID-19 and anti-Black racism, the research leverages Python coding to automatically extract information from Medium.com. The study highlights obstacles, resolutions, and key recommendations for future web scraping studies, emphasizing the effectiveness and efficiency of this method with careful planning, preparedness for challenges, and resource considerations. Barriers such as expertise and technology resources are addressed, along with considerations for virtual work environments and managing hardware and software demands. | 2024 | DeVance Taliaferro, Jocelyn DeVance and Hedadji, Fatima and Duling, Emma, Web Scraping as a Data Collection Strategy: The Perils and Pitfalls. Available at SSRN: https://ssrn.com/abstract=4479267 or http://dx.doi.org/10.2139/ssrn.4479267 |
| 5. | WordNet Semantic Relations Based Enhancement of KNN Model for Implicit Aspect Identification in Sentiment Analysis | This paper addresses the Implicit Aspect Identification task in sentiment analysis, a crucial element for real-world applications in e-commerce and manufacturing. The proposed approach enhances K-Nearest Neighbors (KNN) by incorporating WordNet semantic relations for improved distance computation. Empirical evaluations on electronic products and restaurant review datasets demonstrate the effectiveness of the approach, considering factors such as KNN distance, the number of nearest neighbors (K), and behavior towards Overfitting and Underfitting. The results indicate that the proposed method enhances KNN performance, particularly in Implicit Aspect Identification tasks, offering valuable insights for sentiment analysis applications. | 2023 | Benarafa, H., Benkhalifa, M. & Akhloufi, M. WordNet Semantic Relations Based Enhancement of KNN Model for Implicit Aspect Identification in Sentiment Analysis. *Int J Comput Intell Syst* **16**, 3 (2023). https://doi.org/10.1007/s44196-022-00164-8 |
| 6. | Sentiment Analysis “Using SVM, KNN and SVM with PCA” | This paper explores the vast potential of sentiment analysis in natural language processing, emphasizing its ability to decipher human sentiments in various contexts. Focusing on restaurant reviews, the study compares different sentiment analysis techniques—Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Support Vector Machine with Principal Component Analysis (PCA). The goal is to identify the most effective technique for accurately classifying reviews as positive or negative. Automation of sentiment analysis proves valuable for restaurant owners, offering insights into customer preferences and areas for improvement. The study achieves a notable 96% accuracy using the SVM model, showcasing its efficacy compared to other classification models. | 2023 | Verma, P., Bhardwaj, T., Bhatia, A., Mursleen, M. (2023). Sentiment Analysis “Using SVM, KNN and SVM with PCA”. In: Bhardwaj, T., Upadhyay, H., Sharma, T.K., Fernandes, S.L. (eds) Artificial Intelligence in Cyber Security: Theories and Applications. Intelligent Systems Reference Library, vol 240. Springer, Cham. https://doi.org/10.1007/978-3-031-28581-3\_5 |
| 7. | YouTube Sentimental Analysis Using a Combined Approach of KNN and K-means Clustering Algorithm | This paper addresses the necessity of sentiment analysis in comprehending user opinions on YouTube, a widely used video-sharing platform. With the surge in comments on popular channels, dealing with this vast and unstructured data requires effective applications or methods. The study employs sentiment analysis, combining K-Nearest Neighbor (KNN) and K-means clustering approaches to categorize YouTube comments. The proposed technique is compared with SVM classifier and Naive Bayes for accuracy, showcasing its effectiveness in analyzing sentiments on a larger platform like YouTube. | 2025 | Adhikari, S., Kaushik, R., Obaid, A.J., Jeyalaksshmi, S., Balaganesh, D., Hanoon, F.H. (2023). YouTube Sentimental Analysis Using a Combined Approach of KNN and K-means Clustering Algorithm. In: Peng, SL., Jhanjhi, N.Z., Pal, S., Amsaad, F. (eds) Proceedings of 3rd International Conference on Mathematical Modeling and Computational Science. ICMMCS 2023. Advances in Intelligent Systems and Computing, vol 1450. Springer, Singapore. https://doi.org/10.1007/978-981-99-3611-3\_4 |
| 8. | Sentiment Analysis of Review Datasets Using Naive Bayes and K-NN Classifier | This project addresses the surge in sentimental content on the web, focusing on movie and hotel reviews in social media. The sentiment-focused web crawling framework utilizes statistical methods to capture subjective style and sentence polarity. The study employs two supervised machine learning algorithms, K-Nearest Neighbour (K-NN) and Naive Bayes, for sentiment analysis. In movie reviews, Naive Bayes outperforms K-NN, while both algorithms show comparable accuracies for hotel reviews. The project emphasizes the importance of timely discovery of sentimental web content for applications like contextual advertisements, recommendation systems, and market trend analysis in the Web 2.0 era. | 2016 | [Dey, L., Chakraborty, S., Biswas, A., Bose, B., & Tiwari, S. (2016). Sentiment Analysis of Review Datasets Using Naïve Bayes‘ and K-NN Classifier. International Journal of Information Engineering and Electronic Business, 8(4), 54–62. MECS Publisher. ISSN: 2074-9031. DOI: 10.5815/ijieeb.2016.04.07.](http://dx.doi.org/10.5815/ijieeb.2016.04.07) |
| 9. | Sentiment Analysis in the Era of Large Language Models: A Reality Check | The paper explores the application of large language models (LLMs), exemplified by ChatGPT, in sentiment analysis (SA). While LLMs show promising results in simpler tasks, their performance lags in more complex sentiment analysis tasks requiring a deeper understanding or structured sentiment information. The study encompasses 13 tasks on 26 datasets, comparing LLMs with small language models (SLMs) trained on domain-specific datasets. Notably, LLMs exhibit significant advantages in few-shot learning scenarios, indicating their potential when annotation resources are limited. The paper also highlights the limitations of current evaluation practices and introduces a new benchmark, SENTIEVAL, to assess LLMs' sentiment analysis capabilities comprehensively. | 2023 | Zhang, W., & Deng, Y. (2023). Sentiment Analysis in the Era of Large Language Models: A Reality Check. Retrieved from https://synthical.com/article/85237ecb-ae59-47ec-9c7c-c26866cf9cfa. arXiv preprint arXiv:2305.15005. |
| 10. | Modelling Sentiment Analysis: LLMs and augmentation techniques | This paper explores various approaches for binary sentiment classification on a limited training dataset, utilizing large language models (LLMs) like BERT, RoBERTa, and XLNet, known for delivering state-of-the-art results in sentiment analysis. Additionally, the paper introduces diverse data augmentation techniques to address the challenges posed by the small training dataset. | 2023 | Guillem Senabre Prades. "Modelling Sentiment Analysis: LLMs and Data Augmentation Techniques." arXiv preprint, 2023. |

* 1. **Research Gap**

Despite the growing demand for efficient healthcare management systems, many blood banks still rely on outdated or manual methods for managing donor records, tracking blood inventory, and handling urgent blood requests. These traditional systems lack real-time data access, transparency, and integration with notification mechanisms, often resulting in delays during emergencies. Moreover, there is limited adoption of secure, web-based platforms that ensure both data privacy and operational efficiency in smaller or rural healthcare facilities. The research gap, therefore, lies in the absence of a centralized, automated, and user-friendly solution that can be easily adopted by blood banks to streamline operations and enhance responsiveness. This project addresses that gap by developing a modern, secure, and scalable system using open-source technologies to improve the management and accessibility of blood-related data.

* 1. **Problem Statement**

In many parts of the world, blood banks and hospitals continue to rely on outdated, manual systems to manage critical processes such as blood donation records, inventory tracking, and recipient requests. These systems are often paper-based or limited to isolated spreadsheets, which are prone to human error, data duplication, and loss. As a result, during medical emergencies, it becomes difficult to quickly determine blood availability or locate suitable donors. The absence of a real-time, centralized system can cause dangerous delays in fulfilling blood requests, sometimes with life-threatening consequences. The lack of efficient communication between blood banks and potential donors or requesters further amplifies the challenges in managing time-sensitive scenarios.

Moreover, most existing blood bank systems, if any, are either too costly for small clinics or do not offer the level of customization and simplicity required by medium- to low-resource health centers. In addition, many of these systems lack robust user authentication, proper donor logging, or notification features for managing urgent blood demands. There is also a noticeable gap in systems that allow both the general public and authorized medical staff to interact with the platform in a secure and role-appropriate manner. Public users often have no way to request blood through a digital platform, and reception or admin staff are left with inefficient methods to respond, update inventory, or register donors. These inefficiencies not only affect emergency responsiveness but also hamper the systematic growth and reliability of blood bank operations.

This project addresses these critical problems by proposing a secure, web-based Blood Bank Management System built using Flask and MySQL. The system provides a centralized platform for donor registration, blood inventory management, request handling, and notification tracking. Public users can request blood online, and employees can log in to manage donations, inventory updates, and track pending requests—all while ensuring data security and accuracy. By integrating these functionalities into a single, streamlined application, the proposed system aims to improve the speed, accuracy, and transparency of blood bank operations, particularly for small to medium-sized healthcare facilities that lack advanced IT infrastructure.

1. **TECHNICAL SPECIFICATION**
   1. **Requirements**

**3.1.1.** Functional

**Data Gathering:**

Broad Coverage: The system should also be capable of extending its data collection capabilities to other emerging platforms and social media to ensure comprehensive market feedback is captured.

Real-time Data Streaming: Implement real-time data gathering capabilities to allow immediate analysis of fresh data, which is critical for time-sensitive feedback.

**Data Preprocessing:**

Language Detection: Incorporate a language detection step to handle multi-language content appropriately, ensuring that the preprocessing tools used are language-specific.

Error Handling: Develop robust error handling mechanisms to manage incomplete or corrupt data entries without disrupting the preprocessing workflow.

**Comment Categorization:**

Scalable Clustering: Ensure the clustering algorithm can scale with increased data volume without sacrificing processing speed or accuracy.

Dynamic Model Updating: Implement mechanisms for the periodic retraining of the clustering model to adapt to new trends and changes in user interaction patterns.

**Priority Assignment:**

User Impact Analysis: Integrate an analysis layer to assess the potential impact of each comment based on the commenter's influence and historical engagement metrics.

Contextual Awareness: Develop the system’s ability to consider the context of the discussion when assigning priorities, recognizing urgent issues even in seemingly neutral comments.

**Output Generation:**

Visualization Tools: Incorporate visualization tools in the output to present the prioritized comments and insights in an intuitive and easy-to-understand manner for quick decision-making.

Feedback Loop: Establish a feedback loop mechanism where the output results can be evaluated and refined based on user or moderator input to continually enhance the accuracy of the prioritization process.

**3.1.2.** Non-Functional

**Performance:**

Adaptive Performance Optimization: Implement adaptive algorithms to optimise resource allocation based on current load and data complexity, ensuring consistent performance under changing situations.

Parallel Processing: Use parallel processing techniques to manage and analyse data simultaneously, dramatically lowering the time required to provide outputs.

Reliability:

Failover Mechanisms: Include automatic failover capabilities to switch to a backup system in the event of a failure, assuring uninterrupted operation.

Regular System Audits: Schedule regular audits and updates to ensure that all components work properly and that any possible problems are addressed promptly.

**Scalability:**

Cloud Integration: Enable dynamic resource allocation depending on the system's present requirements by integrating with cloud services to promote scalability.

Load Balancing: Employ load balancing strategies to divide network traffic and data processing equally among servers.

**Security:**

Encryption Protocols: To prevent unwanted access to sensitive data, use robust encryption algorithms for both data in transit and data at rest.

Frequent Security Updates: To guard against fresh vulnerabilities, keep up a routine of updating security software and protocols.

**Usability:**

Interactive Tutorials: To help new users navigate and operate the system efficiently, include interactive tutorials and help sections on the dashboard.

Customizable User Interfaces: Increase comfort and personalisation by giving users the ability to alter the interface to suit their preferences and usability requirements.

**Maintainability:**

Automated Testing: Implement automated testing frameworks to regularly check and ensure that all parts of the system are functioning as intended.

Version Control Integration: Use version control systems to manage changes and updates to the software, ensuring that maintenance and upgrades are systematically tracked and implemented.

**Compatibility:**

Multi-platform Support: Ensure the system is compatible across various operating systems and devices, including mobile platforms, to enhance accessibility.

Standards Compliance: Adhere to international web standards and protocols to ensure compatibility and interoperability with a wide range of web technologies.

**Accuracy:**

Continuous Learning: Integrate continuous learning mechanisms that allow the system to learn from past decisions and refine its algorithms over time, improving accuracy.

Quality Assurance Metrics: Establish and monitor key quality assurance metrics that track the accuracy of data categorization and prioritization, ensuring the system meets high standards.

* 1. **Feasibility Study**

### ****1. Technical Feasibility****

The project is technically feasible due to the use of well-established and widely supported technologies. Python, with the Flask web framework, offers simplicity and flexibility for rapid development. MySQL provides a robust relational database for data storage and retrieval. The system’s requirements — a basic server environment, a web browser, and internet connectivity — make it easy to deploy across most healthcare environments. Additionally, the use of open-source libraries such as passlib for password hashing and Flask-MySQLdb for database interaction ensures secure and efficient system operations. The technical team involved possesses adequate experience in these technologies, ensuring that implementation and maintenance are manageable.

### ****2. Operational Feasibility****

The system is highly operable in real-world blood bank environments, particularly in small- to medium-scale hospitals or donation centers. Users such as receptionists and admin staff require minimal training to navigate the intuitive web interface. The application’s role-based design ensures that public users, donors, and employees access only the features relevant to them, improving security and usability. The ability to handle donor registration, request submission, inventory management, and notifications in a single platform significantly reduces the operational burden and improves response time during emergencies. This system thus offers practical improvements over existing manual or semi-digital approaches.

### ****3. Economic Feasibility****

From an economic standpoint, the project is cost-effective. Since all core components (Flask, MySQL, Bootstrap, HTML/CSS, and related Python libraries) are open-source, there are no licensing costs involved. The system can be hosted on a low-cost server or even run locally within an internal hospital network. Additionally, long-term operational costs are minimal, requiring only periodic updates or maintenance. The efficiency gained in managing blood stock, donor logs, and request handling can translate into savings in manpower and faster medical response, making the return on investment (ROI) highly favorable.

### ****4. Legal Feasibility****

The system complies with general software development and data protection practices. Passwords are encrypted using secure hashing techniques to ensure user credentials are not stored in plain text. Although the current system does not implement advanced compliance with health data regulations (such as HIPAA or GDPR), the modular design allows for easy integration of such measures in the future. Ensuring restricted access to sensitive data and maintaining proper audit logs can help the system conform to data protection standards required in medical environments.

### ****5. Schedule Feasibility****

The project is designed to be developed within a short to moderate timeframe, depending on the size of the development team. A working prototype with core functionality — including registration, login, donor entry, blood stock updates, and notification handling — can be developed and deployed within 4 to 6 weeks. Additional features like SMS/email integration, analytics, or mobile responsiveness can be added in subsequent development cycles. Given the clarity of requirements and availability of tools, the project is well within achievable time limits.

* 1. **System Specification**

**3.3.1.** Hardware Specification

**AWS EC2 Instance:** Using an AWS EC2 instance to host the backend is a smart decision that leverages Amazon's cloud infrastructure to assure scalability and stability. EC2 instances can be dynamically modified to match the system's computational demands, giving administrators greater flexibility in managing computational resources.

**High Capacity GPU (e.g., NVIDIA GeForce RTX 3090):** A high-performance GPU, such as the NVIDIA GeForce RTX 3090, is required for processing big datasets and completing complicated calculations quickly. GPUs are especially useful for machine learning and data analysis activities, where parallel computing can drastically cut data processing and model training time.

**3.3.2.** Software Specification

**Python Environment with Anaconda Navigator:** Using Anaconda Navigator to manage the Python environment is a practical option because it simplifies the installation and management of various Python packages and their dependencies. Anaconda includes a comprehensive set of tools and libraries required for data science and machine learning projects, such as MatPlotLib for data visualisation and NumPy for numerical computations, among others.

**Clustering Calculations:** The software must perform clustering computations, which are critical to the system's capacity to categorise and prioritise user comments. These calculations demand strong computing assistance, necessitating the employment of powerful hardware and efficient software libraries.

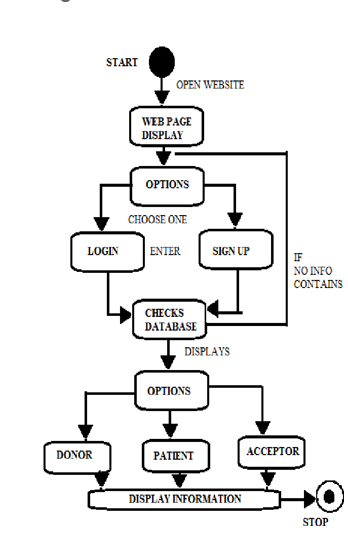
**Flask for Backend API:** Flask is a Python web framework that is both lightweight and powerful, making it ideal for developing APIs. It is chosen to function as a microservice backend, processing API requests between the frontend and the server. Flask's simplicity and versatility make it ideal for projects that demand a clean and efficient approach to deploy web services without the expense of larger frameworks such as Django.

**Integration and Implementation :**

Integration: By combining high-performance hardware with advanced software settings, the system is better able to handle intense tasks such as real-time data processing and machine learning model deployments.

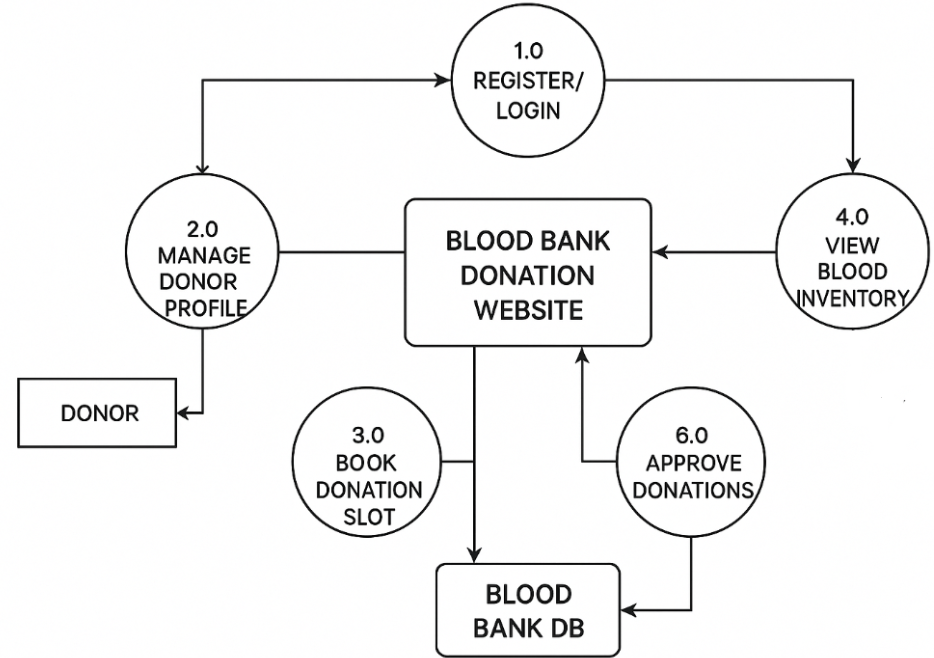
Implementation: These standards must be carefully planned and configured to guarantee that all components work together seamlessly. This includes creating EC2 instances, configuring GPUs for peak performance, establishing a Python environment with Anaconda, and deploying Flask to handle API calls**.**

1. **DESIGN APPROACH AND DETAILS**
   1. **System Architecture**

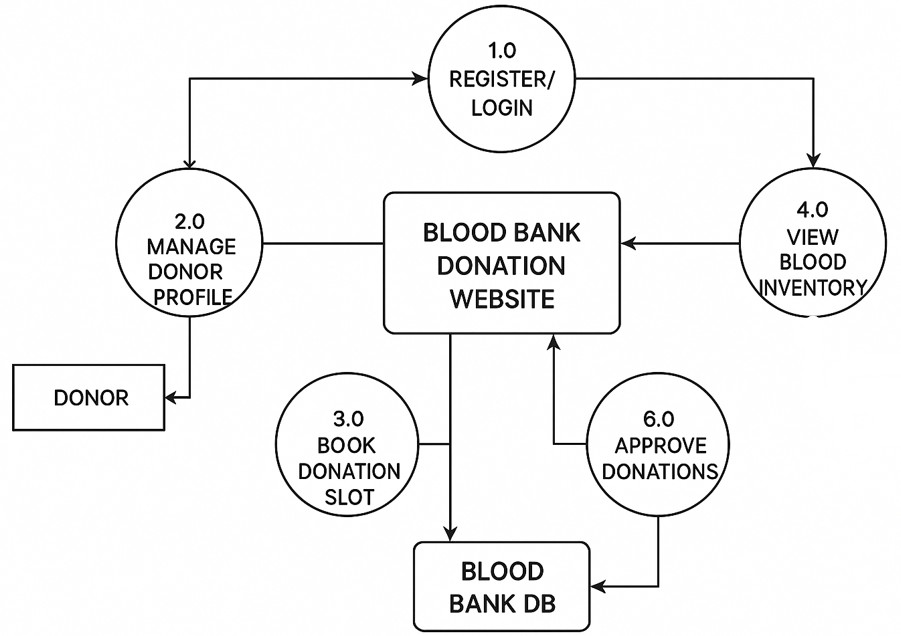
****

* 1. **Design**

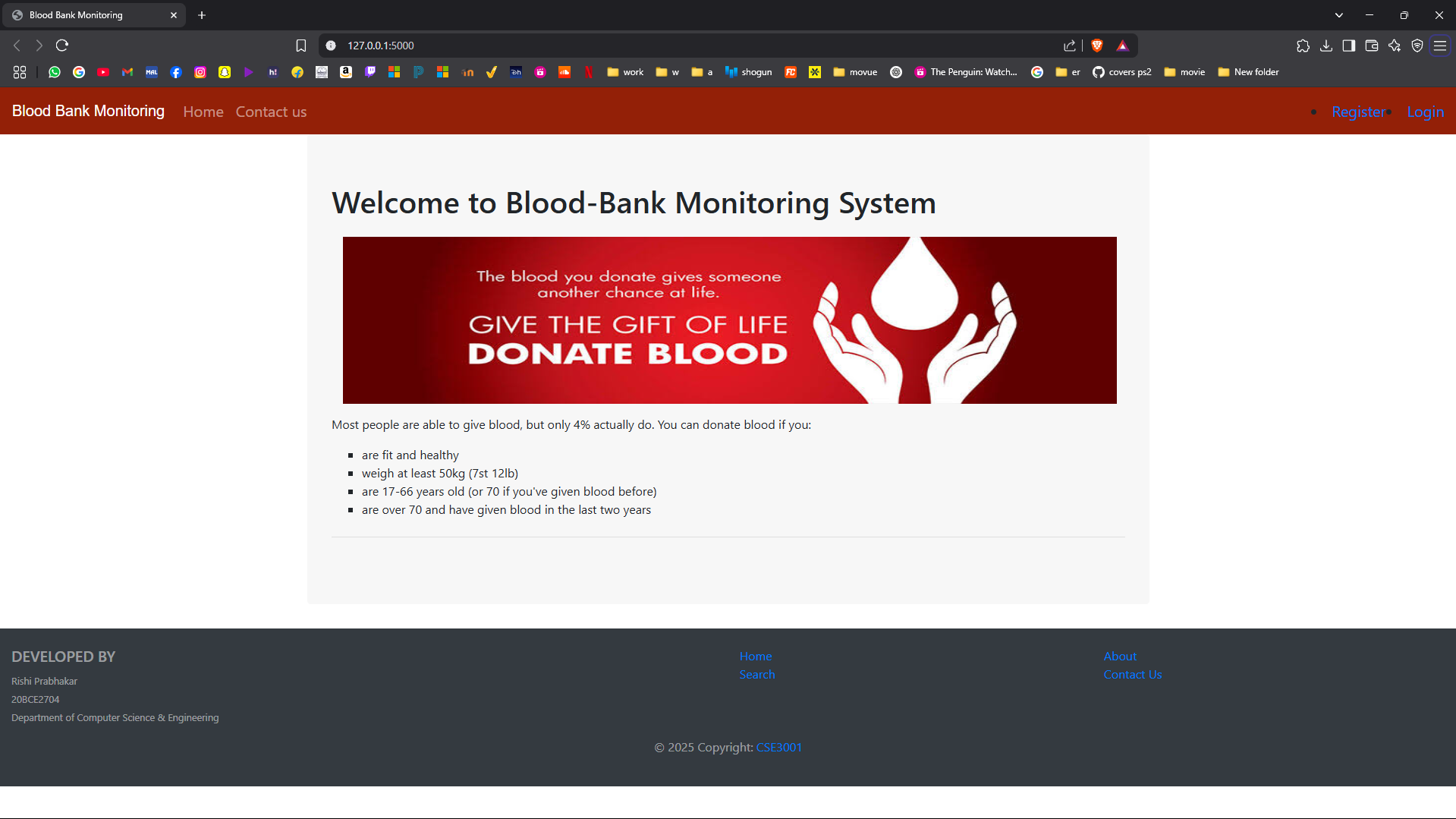
**4.2.1**. **Data Flow Diagram**

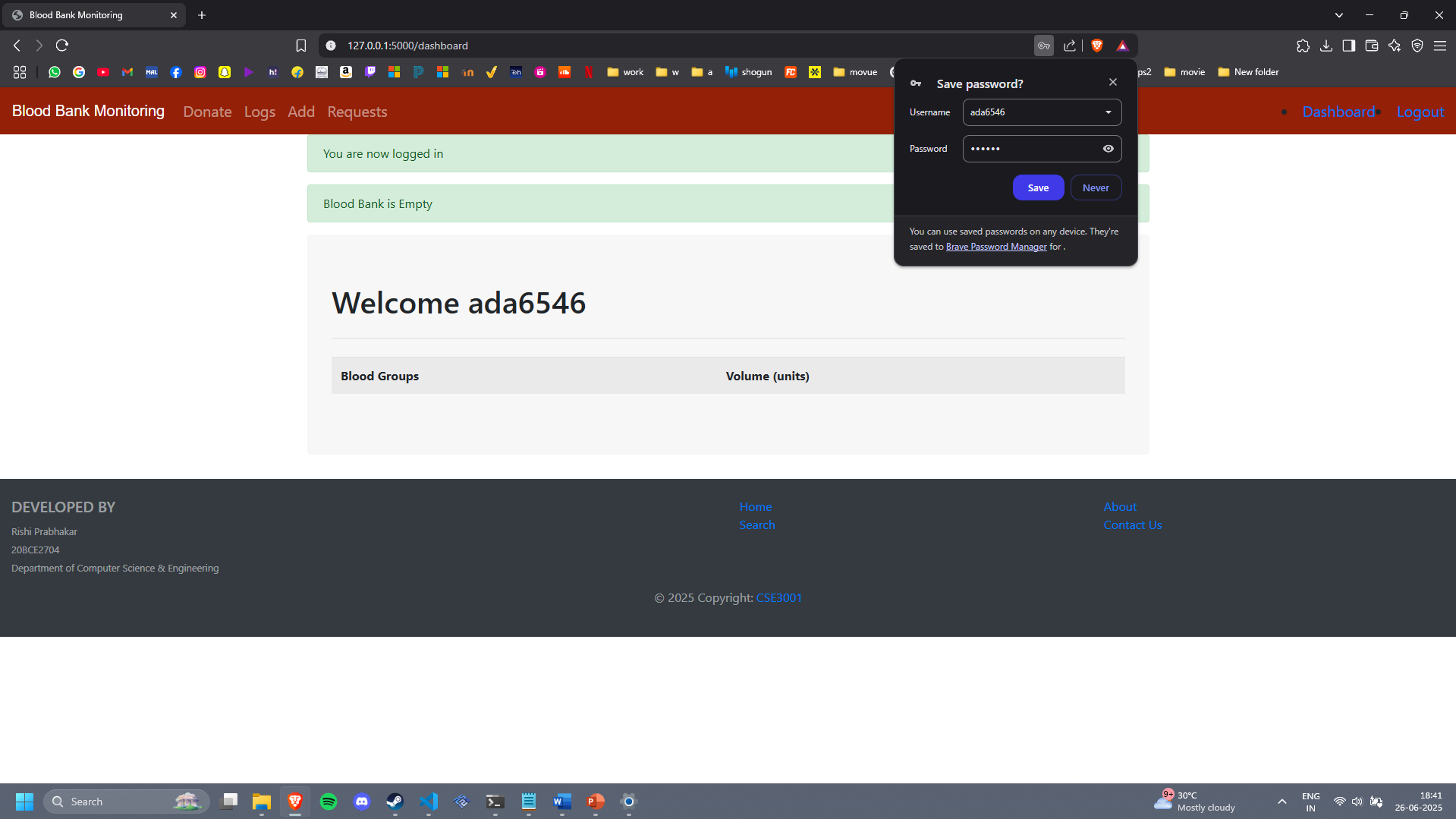
****

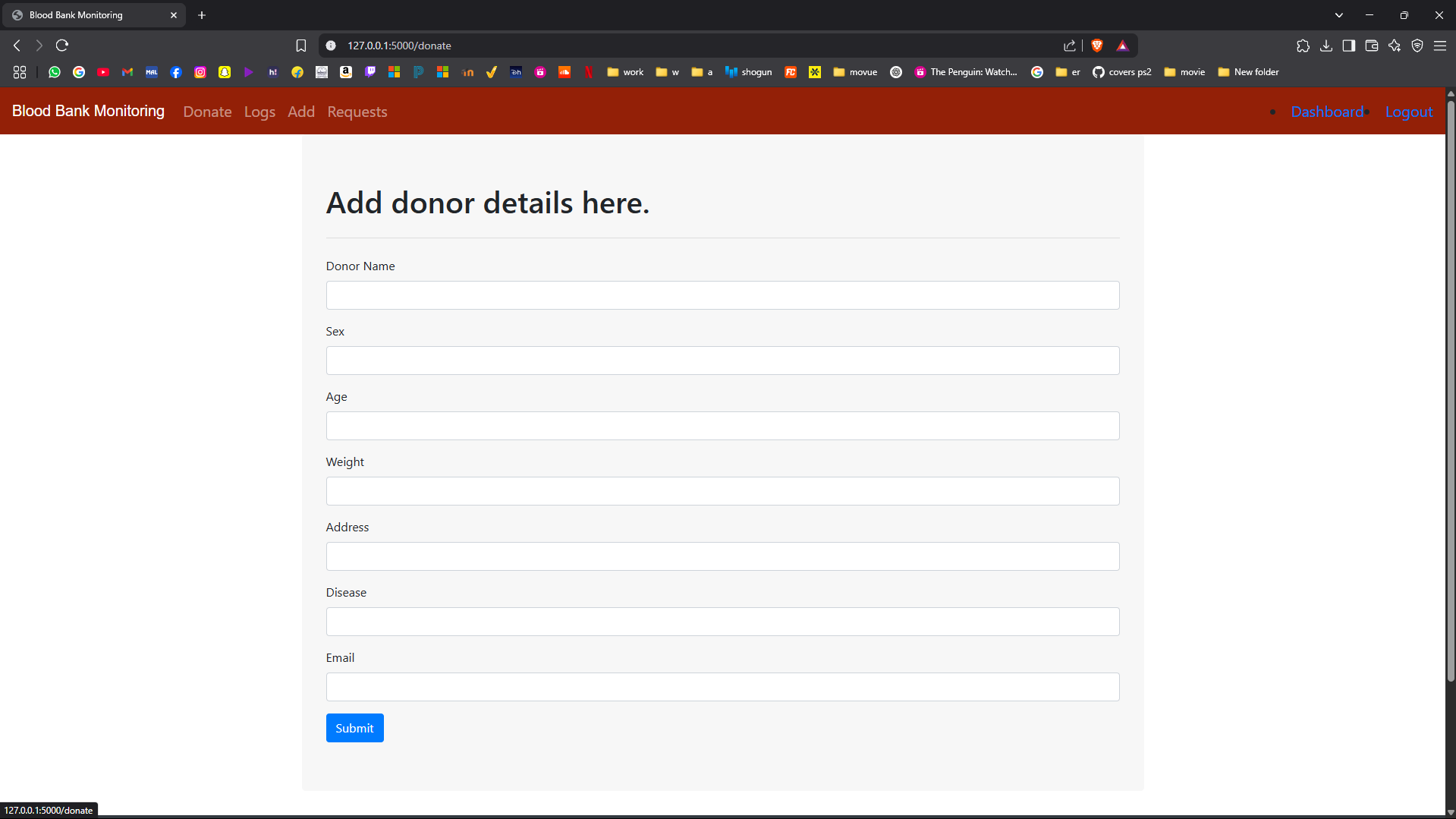
4.2. Dataflow diagram illustration

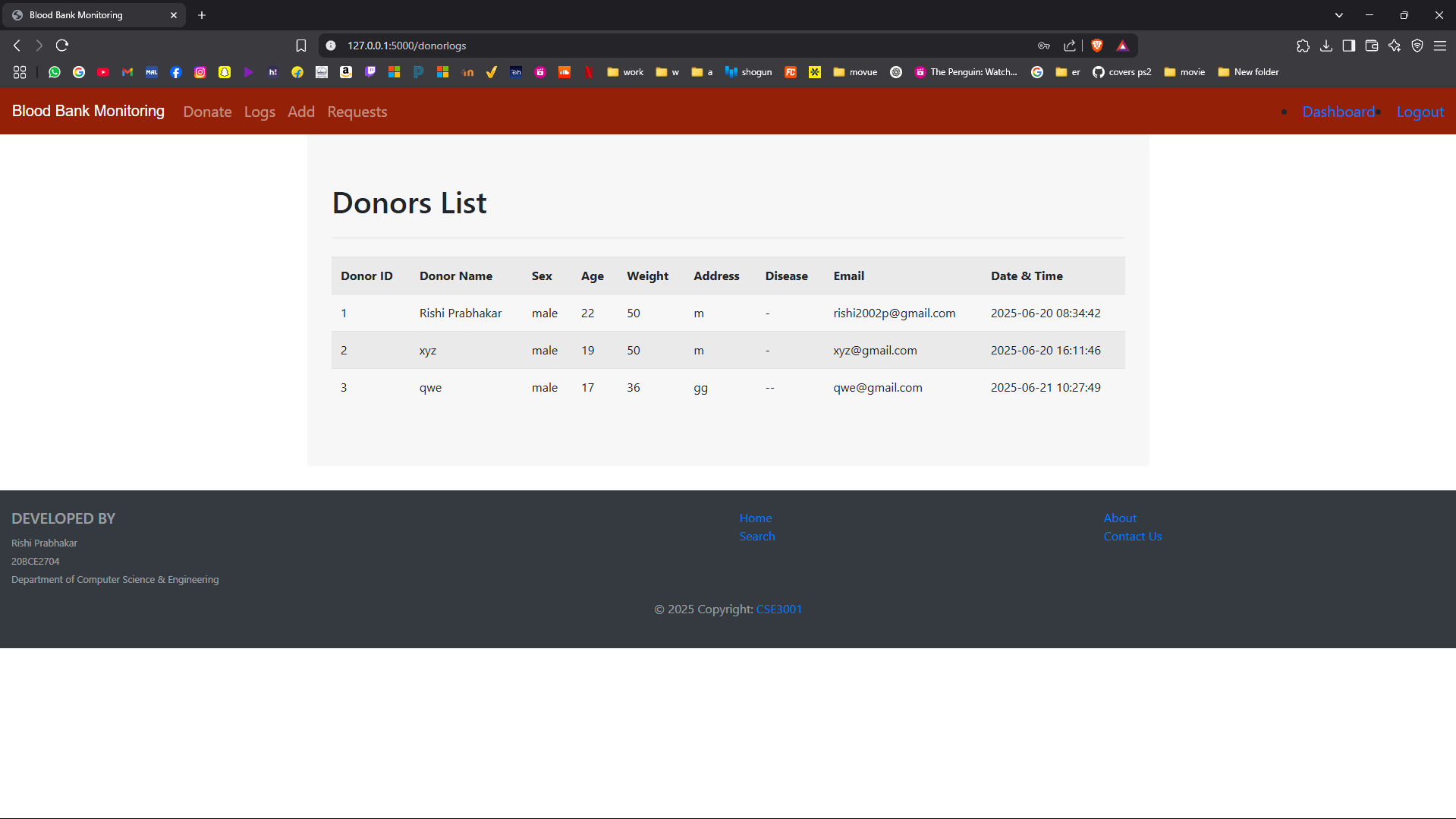


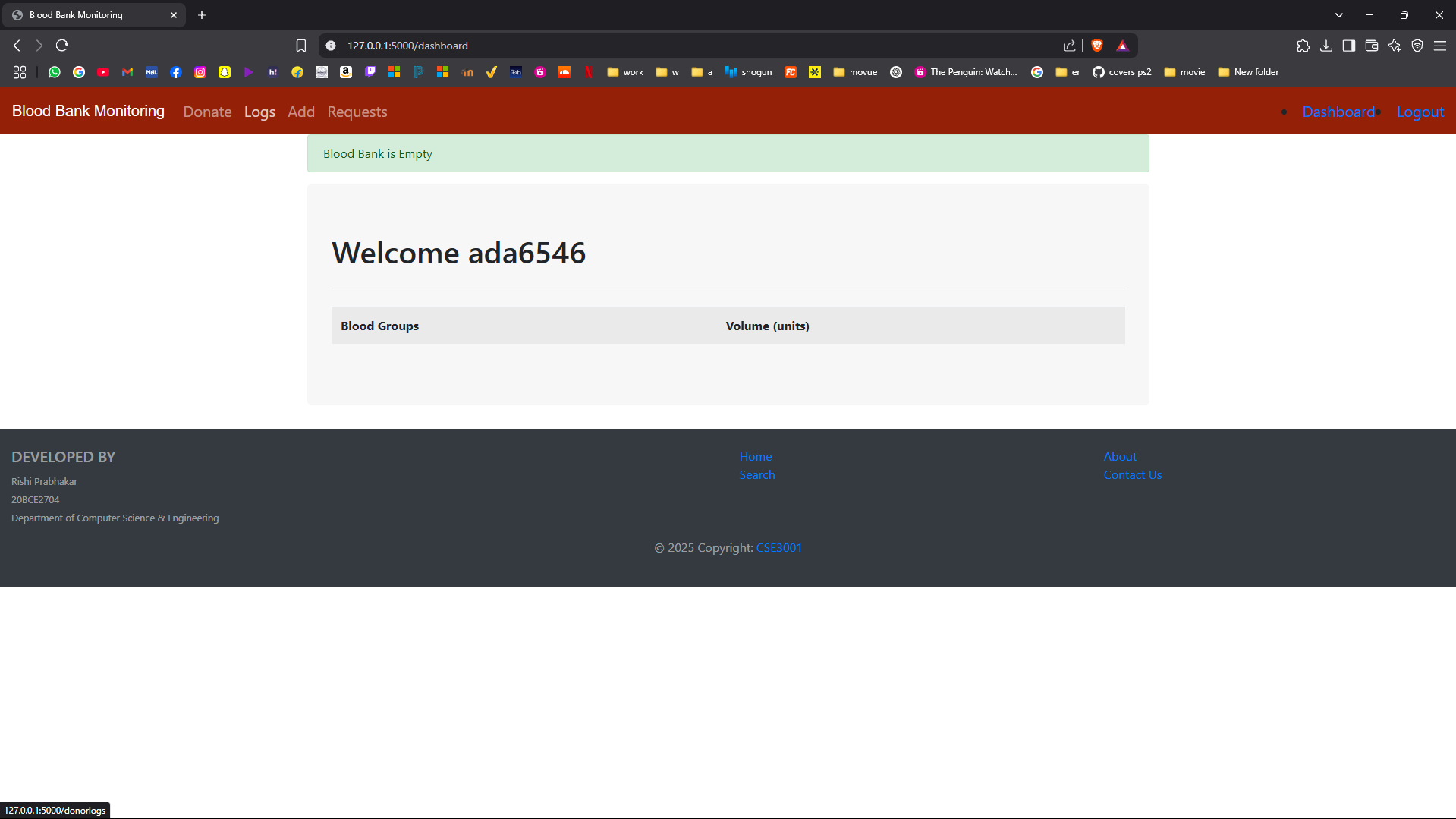
1. **PROJECT DEMONSTRATION**

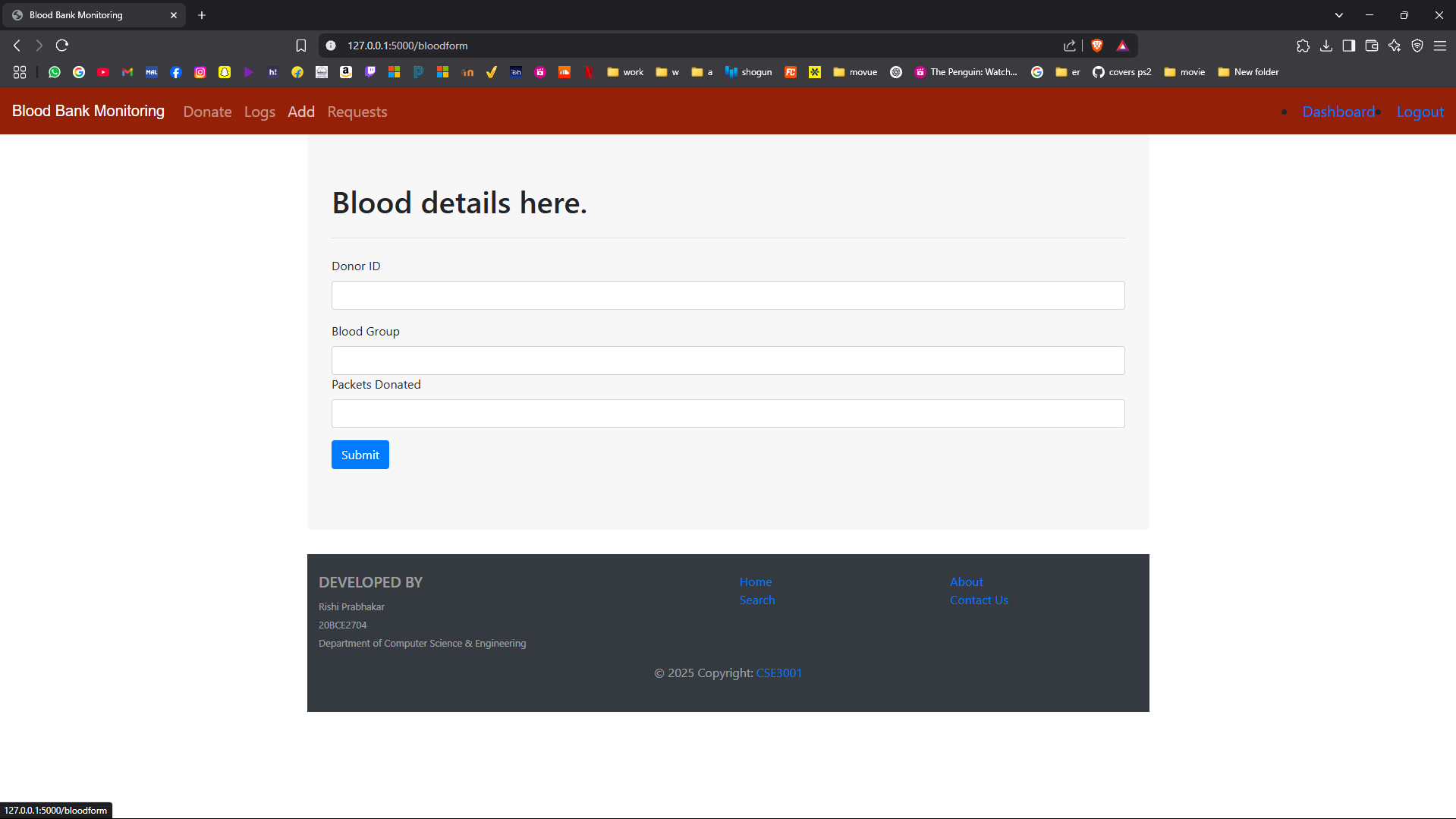












1. **COST ANALYSIS / RESULT & DISCUSSION**
   1. **Cost Analysis**

A cost analysis of the Blood Bank Donation Website project involves three primary components: **development**, **infrastructure**, and **maintenance**. The **initial development cost** is estimated to be between ₹2,45,000 and ₹5,10,000. This includes UI/UX design, frontend and backend development, database setup, admin panel creation, API integrations (such as SMS or email notifications), and thorough testing and project management. Once the system is built, **monthly infrastructure costs** such as cloud hosting, domain registration, SSL certificates, database services, and alert systems range from ₹5,000 to ₹12,000, depending on the scale and traffic. Additionally, **ongoing maintenance and support** to handle bug fixes, performance monitoring, and feature updates may cost an extra ₹10,000 to ₹20,000 per month. Altogether, the project requires a solid upfront investment with recurring costs for smooth operation, security, and user satisfaction.

1. **SUMMARY**

The **Blood Bank Donation Website** is an online platform designed to streamline blood donation and request processes between **donors**, **hospitals**, and **administrators**. It allows **donors** to register, update health profiles, book donation slots, and receive alerts. **Hospitals or admins** can request specific blood types, view inventory levels, and manage donation approvals. The system maintains a centralized **database** to track donor information, blood stock, and donation history. Key features include secure login, appointment booking, inventory management, and automated communication (email/SMS). The platform improves accessibility, speeds up matching of donors and recipients, and enhances the overall efficiency of blood bank operations.

1. **REFERENCES**

Web Links:

[1] Cloudflare. "Under Attack." [Online]. Available: https://www.cloudflare.com/under-attack. Accessed: Dec. 2014.

[2] The Daily Beast. "Hackers' 10 Most Famous Attacks: Worms and DDoS Takedowns." [Online]. Available: http://www.thedailybeast.com/articles/2010/12/11/hackers-10-most-famous-attacks-wormsandddos-takedowns.html. Accessed: Dec. 2014.

Journals:

[3] G. Zacharia, "Trust Management through Reputation Mechanisms," in Workshop in Deception, Fraud and Trust in Agent Societies, Third International Conference on Autonomous Agents (Agents’99), ACM, 1999. International Journal of Security and Its Applications, vol. 9, no. 9, 2015, pp. 210. [Online]. Available: [Source Link].

[4] L. Eschenauer, V. D. Gligor, and J. Bara, "On Trust Establishment in Mobile Ad Hoc Networks," in Security Protocols, Springer, 2004, pp. 47-66.

[5] N.Ch. S.N. Iyengar, Gopinath Ganapathy, P.C. Mogan Kumar, and Ajith Abraham, "A multilevel thrust filtration defending mechanism against DDoS attacks in cloud computing environment," International Journal of Grid and Utility Computing, vol. 5, no. 4, 2014, pp. 236-248.

Book:

[6] A.P. Malvino and D.P. Leach, "Digital Principles and Applications," Tata McGraw Hill, 2014, Special Edition – 2009.

1. **APPENDIX A – SAMPLE CODE**

**App.py**

|  |
| --- |
| **from flask import Flask, render\_template, flash, redirect, request, url\_for, session, logging**  **from flask\_mysqldb import MySQL**  **from wtforms import Form, StringField, TextAreaField, PasswordField, validators, SelectField**  **from passlib.hash import sha256\_crypt**  **import random**  **from functools import wraps**  **app = Flask(\_\_name\_\_)**  **app.secret\_key='some secret key'**    **app.config['MYSQL\_HOST']='localhost'**  **app.config['MYSQL\_USER']='root'**  **app.config['MYSQL\_PASSWORD']='123456'**  **app.config['MYSQL\_DB']='Bloodbank'**  **app.config['MYSQL\_CURSORCLASS']='DictCursor'**  **mysql =  MySQL(app)**  **@app.route('/')**  **def index():**  **return render\_template('home.html')**  **@app.route('/contact', methods=['GET','POST'])**  **def contact():**  **if request.method == 'POST':**  **bgroup = request.form["bgroup"]**  **bpackets = request.form["bpackets"]**  **fname = request.form["fname"]**  **adress = request.form["adress"]**    **cur = mysql.connection.cursor()**    **cur.execute("INSERT INTO CONTACT(B\_GROUP,C\_PACKETS,F\_NAME,ADRESS) VALUES(%s, %s, %s, %s)",(bgroup, bpackets, fname, adress))**  **cur.execute("INSERT INTO NOTIFICATIONS(NB\_GROUP,N\_PACKETS,NF\_NAME,NADRESS) VALUES(%s, %s, %s, %s)",(bgroup, bpackets, fname, adress))**    **mysql.connection.commit()**  **cur.close()**  **flash('Your request is successfully sent to the Blood Bank','success')**  **return redirect(url\_for('index'))**  **return render\_template('contact.html')**  **class RegisterForm(Form):**  **name = StringField('Name', [validators.DataRequired(),validators.Length(min=1,max=25)])**  **email = StringField('Email',[validators.DataRequired(),validators.Length(min=10,max=50)])**  **password = PasswordField('Password', [**  **validators.DataRequired(),**  **validators.EqualTo('confirm',message='Password do not match')**  **])**  **confirm = PasswordField('Confirm Password')**  **@app.route('/register', methods=['GET','POST'])**  **def register():**  **form = RegisterForm(request.form)**  **if request.method  == 'POST' and form.validate():**  **name = form.name.data**  **email = form.email.data**  **password = sha256\_crypt.hash(str(form.password.data))**  **e\_id = name+str(random.randint(1111,9999))**    **cur = mysql.connection.cursor()**  **cur.execute("INSERT INTO RECEPTION(E\_ID,NAME,EMAIL,PASSWORD) VALUES(%s, %s, %s, %s)",(e\_id, name, email, password))**    **mysql.connection.commit()**    **cur.close()**  **flashing\_message = "Success! You can log in with Employee ID " + str(e\_id)**  **flash( flashing\_message,"success")**  **return redirect(url\_for('login'))**  **return render\_template('register.html',form = form)**  **@app.route('/login', methods=['GET', 'POST'])**  **def login():**  **if request.method == 'POST':**  **e\_id = request.form["e\_id"]**  **password\_candidate = request.form["password"]**  **cur = mysql.connection.cursor()**  **result = cur.execute("SELECT \* FROM RECEPTION WHERE E\_ID = %s", [e\_id])**  **if result > 0:**  **data = cur.fetchone()**  **password = data['PASSWORD']**  **if sha256\_crypt.verify(password\_candidate, password):**  **# Passed**  **session['logged\_in'] = True**  **session['e\_id'] = e\_id**  **flash('You are now logged in', 'success')**  **return redirect(url\_for('dashboard'))**  **else:**  **error = 'Invalid login'**  **return render\_template('login.html', error=error)**  **cur.close()**  **else:**  **error = 'Employee ID not found'**  **return render\_template('login.html', error=error)**  **return render\_template('login.html')**  **def is\_logged\_in(f):**  **@wraps(f)**  **def wrap(\*args, \*\*kwargs):**  **if 'logged\_in' in session:**  **return f(\*args, \*\*kwargs)**  **else:**  **flash('Unauthorized, Please login!', 'danger')**  **return redirect(url\_for('login'))**  **return wrap**  **@app.route('/logout')**  **@is\_logged\_in**  **def logout():**  **session.clear()**  **flash('You are now logged out', 'success')**  **return redirect(url\_for('index'))**  **@app.route('/dashboard')**  **@is\_logged\_in**  **def dashboard():**  **cur = mysql.connection.cursor()**  **cur.execute("SELECT \* FROM BLOODBANK")**  **details = cur.fetchall()**  **result = len(details)**  **if result>0:**  **return render\_template('dashboard.html',details=details)**  **else:**  **msg = ' Blood Bank is Empty '**  **return render\_template('dashboard.html',msg=msg)**  **cur.close()**  **@app.route('/donate', methods=['GET', 'POST'])**  **@is\_logged\_in**  **def donate():**  **if request.method  == 'POST':**  **dname = request.form["dname"]**  **sex = request.form["sex"]**  **age = request.form["age"]**  **weight = request.form["weight"]**  **address = request.form["address"]**  **disease =  request.form["disease"]**  **demail = request.form["demail"]**  **cur = mysql.connection.cursor()**  **cur.execute("INSERT INTO DONOR(DNAME,SEX,AGE,WEIGHT,ADDRESS,DISEASE,DEMAIL) VALUES(%s, %s, %s, %s, %s, %s, %s)",(dname , sex, age, weight, address, disease, demail))**  **mysql.connection.commit()**  **cur.close()**  **flash('Success! Donor details Added.','success')**  **return redirect(url\_for('donorlogs'))**  **return render\_template('donate.html')**  **@app.route('/donorlogs')**  **@is\_logged\_in**  **def donorlogs():**  **cur = mysql.connection.cursor()**  **result = cur.execute("SELECT \* FROM DONOR")**  **logs = cur.fetchall()**  **if result>0:**  **return render\_template('donorlogs.html',logs=logs)**  **else:**  **msg = ' No logs found '**  **return render\_template('donorlogs.html',msg=msg)**  **cur.close()**  **@app.route('/bloodform',methods=['GET','POST'])**  **@is\_logged\_in**  **def bloodform():**  **if request.method  == 'POST':**  **d\_id = request.form["d\_id"]**  **blood\_group = request.form["blood\_group"]**  **packets = request.form["packets"]**  **cur = mysql.connection.cursor()**  **cur.execute("INSERT INTO BLOOD(D\_ID,B\_GROUP,PACKETS) VALUES(%s, %s, %s)",(d\_id , blood\_group, packets))**  **cur.execute("SELECT \* FROM BLOODBANK")**  **records = cur.fetchall()**  **cur.execute("UPDATE BLOODBANK SET TOTAL\_PACKETS = TOTAL\_PACKETS+%s WHERE B\_GROUP = %s",(packets,blood\_group))**  **mysql.connection.commit()**  **cur.close()**  **flash('Success! Donor Blood details Added.','success')**  **return redirect(url\_for('dashboard'))**  **return render\_template('bloodform.html')**  **@app.route('/notifications')**  **@is\_logged\_in**  **def notifications():**  **cur = mysql.connection.cursor()**  **result = cur.execute("SELECT \* FROM CONTACT")**  **requests = cur.fetchall()**  **if result>0:**  **return render\_template('notification.html',requests=requests)**  **else:**  **msg = ' No requests found '**  **return render\_template('notification.html',msg=msg)**  **cur.close()**  **@app.route('/notifications/accept')**  **@is\_logged\_in**  **def accept():**  **flash('Request Accepted','success')**  **return redirect(url\_for('notifications'))**  **@app.route('/notifications/decline')**  **@is\_logged\_in**  **def decline():**  **msg = 'Request Declined'**  **flash(msg,'danger')**  **return redirect(url\_for('notifications'))**  **if \_\_name\_\_ == '\_\_main\_\_':**  **app.run(debug=True)** |

**Bloodbank\_session.sql**

|  |
| --- |
| CREATE TABLE RECEPTION(  E\_ID VARCHAR(54) PRIMARY KEY,  NAME VARCHAR(100),  EMAIL VARCHAR(100),  PASSWORD VARCHAR(100),  REGISTER\_DATE TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  );  CREATE TABLE DONOR(  D\_ID INT(3) NOT NULL AUTO\_INCREMENT,  DNAME VARCHAR(50),  SEX VARCHAR(10),  AGE INT(3),  WEIGHT INT(3),  ADDRESS VARCHAR(150),  DISEASE VARCHAR(50),  DEMAIL VARCHAR(100),  Bloodgroud VARCHAR(4),  DONOR\_DATE TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  CONSTRAINT PK\_2 PRIMARY KEY(D\_ID)  );  CREATE TABLE BLOODBANK(  B\_GROUP VARCHAR(4),  TOTAL\_PACKETS INT(4),  CONSTRAINT PK\_3 PRIMARY KEY(B\_GROUP)  );  CREATE TABLE BLOOD(  B\_CODE INT(4) NOT NULL AUTO\_INCREMENT,  D\_ID INT(3),  B\_GROUP VARCHAR(4),  PACKETS INT(2),  CONSTRAINT PK\_4 PRIMARY KEY(B\_CODE),  CONSTRAINT FK\_1 FOREIGN KEY(D\_ID) REFERENCES DONOR(D\_ID) ON DELETE CASCADE ON UPDATE CASCADE,  CONSTRAINT FK\_2 FOREIGN KEY(B\_GROUP) REFERENCES BLOODBANK(B\_GROUP) ON DELETE CASCADE ON UPDATE CASCADE  );  CREATE TABLE CONTACT(  CONTACT\_ID INT(3) NOT NULL AUTO\_INCREMENT,  B\_GROUP VARCHAR(4),  C\_PACKETS INT(2),  F\_NAME VARCHAR(50),  ADRESS VARCHAR(250),  CONSTRAINT PK\_5 PRIMARY KEY(CONTACT\_ID),  CONSTRAINT FK\_3 FOREIGN KEY(B\_GROUP) REFERENCES BLOODBANK(B\_GROUP) ON DELETE CASCADE ON UPDATE CASCADE  )ENGINE=InnoDB AUTO\_INCREMENT=100 DEFAULT CHARSET=latin1;  CREATE TABLE NOTIFICATIONS(  N\_ID INT(3) NOT NULL AUTO\_INCREMENT,  NB\_GROUP VARCHAR(4),  N\_PACKETS INT(2),  NF\_NAME VARCHAR(50),  NADRESS VARCHAR(250),  CONSTRAINT PK\_6 PRIMARY KEY(N\_ID)  )ENGINE=InnoDB AUTO\_INCREMENT=100 DEFAULT CHARSET=latin1;  CREATE TABLE CONTACT (      ID INT AUTO\_INCREMENT PRIMARY KEY,      B\_GROUP VARCHAR(5),      C\_PACKETS INT,      F\_NAME VARCHAR(50),      ADRESS TEXT  );  DELIMITER //  CREATE TRIGGER agecheck  BEFORE INSERT ON DONOR  FOR EACH ROW  BEGIN      IF NEW.age < 21 THEN          SIGNAL SQLSTATE '45000'          SET MESSAGE\_TEXT = 'Donor must be at least 21 years old';      END IF;  END;  //  DELIMITER ; |