IBM NALAIYA THIRAN

PROJECT REPORT

K.L.N COLLEGE OF ENGINEERING, POTTAPALAYAM

(An Autonomous institution, affiliated to Anna university, Chennai)

Problem statement : Plasma Donor Application

Team ID : PNT2022TMID11597

Team Leader : R. Ranjith Kumar

Team members : 1. K. Praveen Kumar

2. O.S. Surya Prakash

3. R. Rakesh Prasanna

4. P. Yogeshwaran

Faculty mentor : T.T. Mathangi

Evaluator : A. Selvaraj

Industry mentor : Prof Navya

INDEX

1. Introduction:

1.1 Project Overview:

The main goal of our project is to design a user-friendly web application that is like a scientific vehicle from which we can help reduce mortality or help those affected by COVID-19 by donating plasma from patients who have recovered without approved antiretroviral therapy planning for deadly COVID-19 infection, plasma therapy is an experimental approach to treat those COVID-positive patients and help them recover faster. Therapy, which is considered reliable and safe. If a particular person has fully recovered from COVID-19, they are eligible to donate their plasma. As we all know, the traditional methods of finding plasma, one must find out for oneself by looking at hospital records and contacting donors have been recovered, sometimes may not be available at home and move to other places. In this type of scenario, the health of those who are sick becomes disastrous. Therefore, it is not considered a rapid process to find plasma. The main purpose of the proposed system, the donor who wants to donate plasma can simply register through the web application and can donate the plasma to the blood bank, the blood bank can apply for the donor and once the donor has accepted the request, the blood bank can add the units they need and the hospital can also send the request to the blood bank that urgently needs the plasma for the patient and can take the plasma from the blood
bank.

1.2 Purpose:

The Plasma Donation Application would help Donors, as well as patients in need of plasma. It would allow you to search for Plasma Donors within your city and having a specific Blood Group. People who have fully recovered from COVID-19 have antibodies in their plasma that can attack the virus. This convalescent plasma is being evaluated as a treatment for patients with serious or immediately life-threatening COVID-19 infections, or those judged by a healthcare provider to be at high risk of progression to severe or life-threatening disease. This application can be considered as a contribution of its developers towards the medical unit of the country as well as towards humanity.

2. Literature Survey:

2.1 Existing System:

When a patient needs plasma, he/she must contact a compatible donor on their circle, but it is difficult to find a suitable donor in a group for a particular time of period. Currently people in need of plasma post pleas on social media to attract potential donors, but pleas on social media take longer to reach a wider audience. As a result, recipients are unable to find the donors within the required time.

2.2 References:

1. R. C. Gojko Adzic, "Serverless computing: Economic and architectural impact," ***ESEC/FSE*****, 2017:** In this paper, the author has carried out analysis based on the opportunities presented by serverless computing. They emphasise that serverless services are more affordable approach for many network services and it is more user friendly as serverless approach will relieve the customers from the intricacies of deployment. These services will help to improve the new business opportunities.

2. P. C. P. C. a. V. I. M. Yan, "Building a chatbot with serverless computing," ***IBM watson research center,*****2016:** Author conducted a survey of existing serverless platform in this paper from source projects, industry, academia, use cases, and key characteristics and has described the challenges and the open problems associated with it. Authors work presented a handson experience of serverless technologies using different services from different cloud provides such as Amazon, Google, IBM, Microsoft Azure.

3. S. E. a. B. J. J. Short, ""Cloud Event Programming Paradigms: Applications and Analysis,"," ***9th IEEE International Conference on Cloud Computing (CLOUD),*** **pp. pp. 400-406, 2017:** In this paper three demonstrators for IBM Bluemix OpenWhisk was presented. They exhibit even-based programming triggered by weather forecast data, speech utterances and Apple WatchOS2 application data. And also demonstrated a chatbot using IBM Bluemix OpenWhisk that calls on the IBM Watson services which include dates, weather, alarm services, news and music tutor.

4. Z. Al-Ali, ““Making Serverless Computing More Serverless,”,” ***IEEE 11th International Conference on Cloud Computing (CLOUD),*** **pp. pp. 456-459, 2018., 2018:** In this paper serverlessOS was designed. It comprises of components such as 1. desegregation model that leverages desegregation for abstraction but it will enable resources to move fluidly between servers for the performance. 2. The second key component is cloud orchestration layer which helps to manage fine-grained resource placement and allocation throughout the application lifetime with the help of global and local decision making 3. And the third component is an isolation capability which enforces data and resource isolation.

5. A. S. a. S. Jindal, ““EMARS: Efficient Management and Allocation of Resources in Serverless,”,” ***IEEE11th International Conference on Cloud Computing (CLOUD),*** **pp. pp. 827-830, 2018:** In this paper an efficient resource management system for serverless computing framework was proposed which aims to enhance resource with a focus on memory allocation among the containers and the design which was added on top of an open-source serverless platform, openLambda and it is based on allocation workloads and serverless functions memory needs events are triggered.

6. Hamlin, M. R. A., & Mayan, J. A. (2016, 16-17 Dec. 2016). ***Blood donation and life saver-blood donation app.*** **Paper presented at the 2016 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT):** On journal, it was described that the proposed blood bank system was connecting between blood bank and personal donor by sending a message to regular/permanent donor who has been registered before.

7. Sayali Dhond, Pradnya Randhavan, Bhagyashali Munde, Rajnandini Patil, and Vikas Patil, “Android Based Health Application in Cloud Computing For Blood Bank”, International Engineering Research Journal (IERJ) Volume 1 Issue 9 pp. 868-870, 2015: On this journal users can search donor by the nearest location from them by using GPS (Global Positioning System). After the information sent, the closest donor will get an alert for blood donor needs. Blood bank android-based application on cloud computing has been done by the previous study.

8. P. Priya, V. Saranya, S. Shabana and Kavitha Subramani, "The optimization of Blood Donor Information and Management System by Technopedia," International Journal of Innovative Research in Science, Engineering and Technology, Volume 3, Special Issue 1, 2014: Blood donor information and optimization management system also has been done by Priya et al.

9. Sultan Turhan, "An Android Application for Volunteer Blood Donors", Computer Science & Information Technology- CSCP, pp. 23–30, 2015: The smartphone application is being developed to allow searching for voluntary donor nearby, followed by communication between donor especially on the emergency situations.

10. Catassi, C. A., Petersen, E. L. "The Blood Inventory Control System Helping Blood Bank Management Through Computerized Inventory Control", Transfusion, Vol. 7, No. 60, 196: In this article, Catassi and Petersen described computerized blood bank inventory. The purpose is to control the distribution of blood bank and hospital. It is possible to monitor daily blood status.

11. Mittal, N., & Snotra, K. (2017, 26-27 Oct. 2017). *Blood bank information system using Android application.* Paper presented at the 2017 Recent Developments in Control, Automation & Power Engineering (RDCAPE): Mittal and Snotra on their research explain the availability of blood supply during emergency situations is highly important for patients in need. Blood donor centre exist to fulfil this need. But whether personal donor and medical facility, there is no available media to connect them directly. That is why personal donor, and the medical facility should be connected.

12. Ali, R. S., Hafez, T. F., Ali, A. B., & Abd-Alsabour, N. (2017, 22-24 March 2017). *Blood bag: A web application to manage all blood donation and transfusion*

processes. **Paper presented at the 2017 International Conference on Wireless Communications, Signal Processing and Networking (Wisp NET):** In the other case Ali et al propose a blood bag system. It is a web-based system which connect with the central database to control all data from the blood bank and blood donation campaign. Basically, this system identifies donors, tests, and stores blood bags, and deliver them to patients. Blood bag system supports donor and blood bank to help patients in needs of blood donation by centralized control system which can arrange all transfusion process. Every process recorded in the database. With huge data and information, Blood Bank Information System will be very useful that can be managed as decision making system.

2.3 Problem Statement Definition:

During the COVID-19 crisis, the requirement of plasma became a high priority, and the donor count has become low. Saving the donor information and helping the needy by notifying the current donor list, would be a helping hand. Regarding the problem faced, an application is to be built which would take the donor details, store them, and inform them upon a request.

Who does the problem affect?

People who are affected by COVID and need a Plasma Donor.

What is the issue?

When a patient needs plasma, he/she must contact a compatible donor on their circle, family, and friends but it is difficult to find suitable donor within a limited group of people in a given time.

What is the impact of the issue?

During the COVID 19 crisis, the requirement of plasma became high and the donor count being low. It is very difficult to find the respective blood group donors when someone is in need.

What would happen if we didn't solve the problem?

The gap between the Donor and Recipient would widen. People who are eager to donate plasma cannot find the right recipient. Currently, people in need of Plasma post Pleas on social media to attract potential donors. But Plea's on social media take longer to reach a wider audience. As a result, recipients are unable to find donors within the required time.

What would happen when it is fixed?

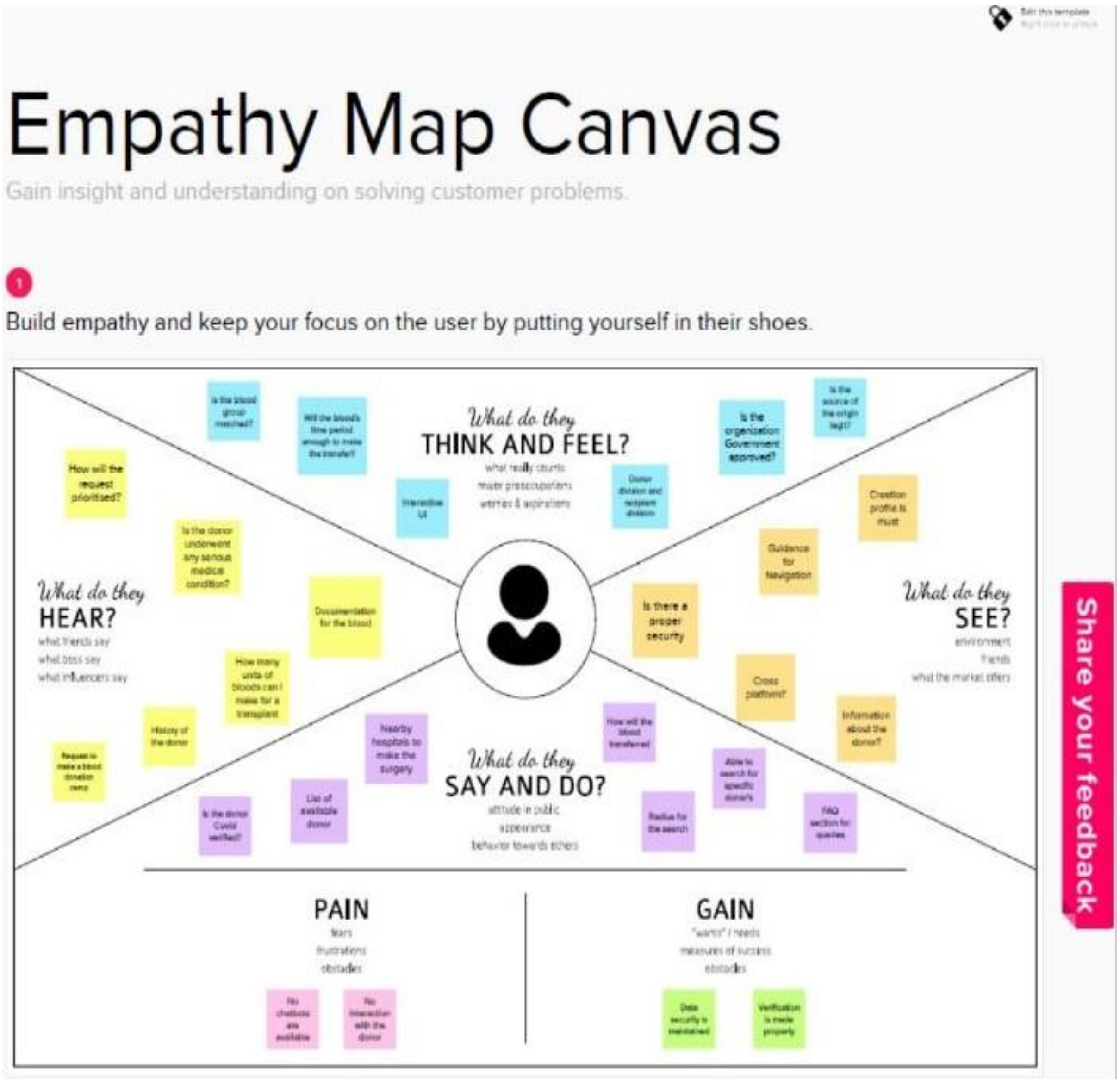
The application makes it feasible for the COVID-19 patients to get a plasma donor easily and makes it possible to find a plasma donor without much difficulty.

Why is it important that we fix the problem?

In severe cases if the recipient is unable to find a donor, then his/her condition could worsen and may potentially result in death.

3. Ideation and Proposed Solution:

3.1 Empathy Map Canvas:



Edit this template
Empathy Map Canvas
Gain insight and understanding on solving customer problems.
1
Build empathy and keep your focus on the user by putting yourself in their shoes.
What do they THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
Is the blood group matched?
Will the blood's time period enough to make the transfer?
Interactive UI
Donor division and recipient division
Is the organization Government approved?
Is the source of the origin legit?
What do they HEAR?
what friends say
what boss say
what influencers say
How will the request prioritised?
Is the donor underwent any serious medical condition?
Documentation for the blood
How many units of bloods can I make for a transplant
History of the donor
Request to make a blood donation camp
What do they SEE?
environment
friends
what the market offers
Creation profile is must
Guidance for Navigation
Is there a proper security
Cross platform?
Information about the donor?
What do they SAY AND DO?
attitude in public
appearance
behavior towards others
Nearby hospitals to make the surgery
List of available donor
Is the donor Covid verified?
How will the blood transferred
Able to search for specific donors
Radius for the search
FAQ section for queries
PAIN
fears
frustrations
obstacles
No chatbots are available
No interaction with the donor
GAIN
"wants" / needs
measures of success
obstacles
Data security is maintained
Verification is made properly
Share your feedback

3.2 Ideation and Brainstorming:

3.3 Proposed Solution:

S.No.	**Parameter**	**Description**
1.	Problem Statement (Problem to be solved)	During COVID 19 crisis the requirement for plasma increased drastically as there were no vaccination found to treat the infected patients, with plasma therapy the recovery rates where high but the donor count was very low and, in such situations, it was very important to get the information about the plasma donors. Saving the donor information and notifying about the current donors would be a helping hand as it can save time and help the users to track down the necessary information about the donor.

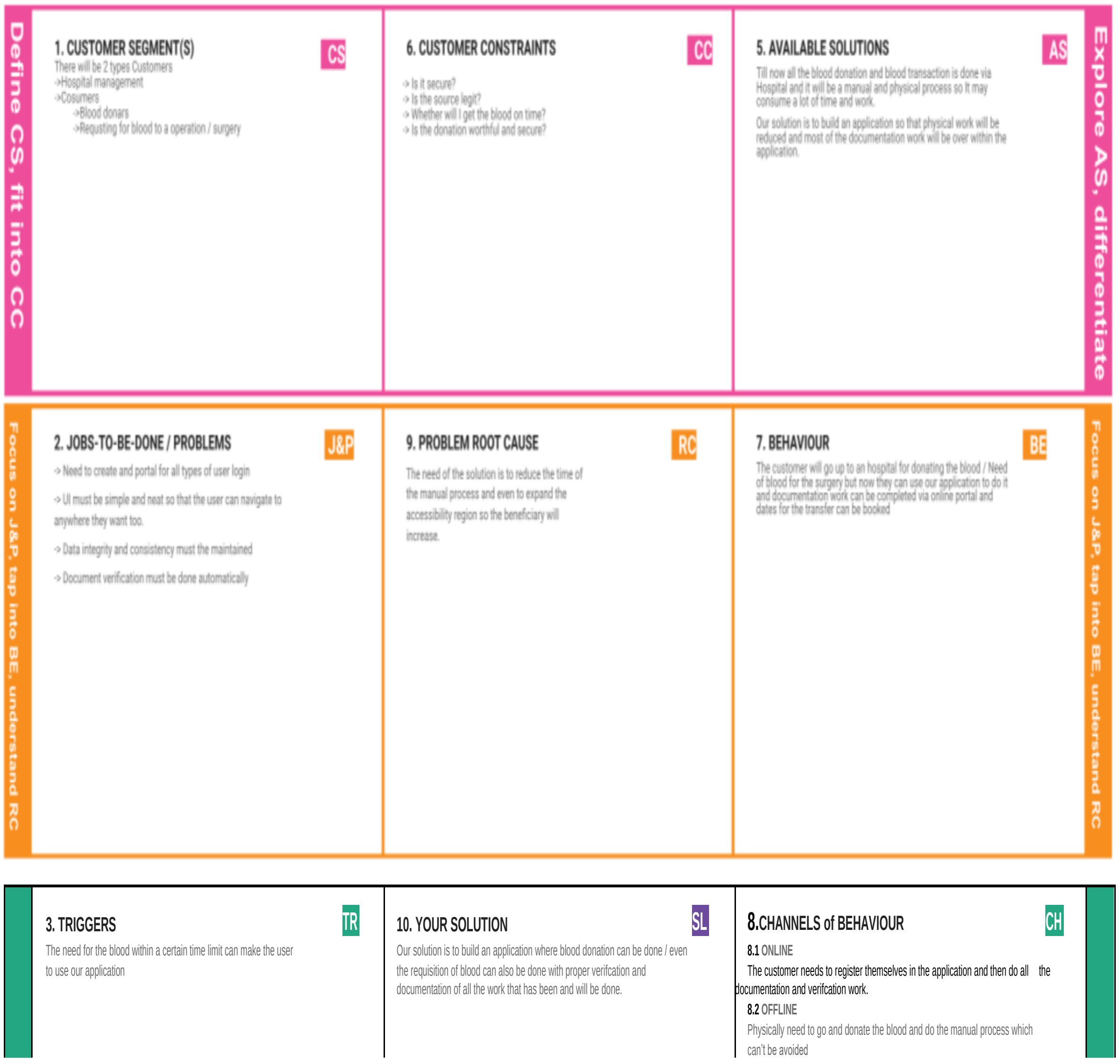
2.	Idea / Solution description	To address the above problem, we are proposing in devising a Plasma Donor Management Application so that users can get proper tracking of Donors in a responsible way and can get the timely delivery of the required medical assistance with blood plasma which can potentially save lives.
3.	Novelty / Uniqueness	Our visions to put together a holistic application for the Plasma Donation cycles. We will implement a blood donation slot appointment for the donation phase, Locations of nearby blood camps. We have also planned to put together the perks and points feature where people are awarded coupons and gift cards for donation blood which they can redeem later. We have also planned to implement a push notification to donor whenever their blood is being used for the medical purposes so that the donors can feel included throughout the blood donation cycle.

4.	Social Impact / Customer Satisfaction	The social impact of our project is immense. By projecting the blood donors as heroes and giving them perks and contests we would be able to attract more and more people to donate blood. Moreover, the idea of slotting blood donation appointments will streamline blood donation process. People who urgently need blood can always browse our application for nearby blood banks for their blood needs.
5.	Business Model (Revenue Model)	The Revenue model for our application is Sales driven. The Order of Blood from our application will be charged some amount of fees. We can get commission payments from listed blood camps. Moreover, we can also conduct Donation Campaigns.
6.	Scalability of the Solution	The scalability of our solution depends upon the initial success and revenue generated from our application. With more revenue we can also facilitate delivery of blood to the hospitals and put forward a subscription-based model with the Hospitals themselves.

		With more and more customers we can also monetize the customer blood requirement data for analytics purposes.

3.4 Problem Solution fit:

Project Title: Plasma Donor Application **Project Design Phase-I - Solution Fit Template** **Team ID:** PNT2022TMID11597


Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

There will be 2 types Customers
->Hospital management
->Cosumers
->Blood donars
->Requsting for blood to a operation / surgery

6. CUSTOMER CONSTRAINTS CC

-> Is it secure?
-> Is the source legit?
-> Whether will I get the blood on time?
-> Is the donation worthful and secure?

5. AVAILABLE SOLUTIONS AS

Till now all the blood donation and blood transaction is done via Hospital and it will be a manual and physical process so It may consume a lot of time and work.

Our solution is to build an application so that physical work will be reduced and most of the documentation work will be over within the application.

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

-> Need to create and portal for all types of user login

-> UI must be simple and neat so that the user can navigate to anywhere they want too.

-> Data integrity and consistency must the maintained

-> Document verification must be done automatically

9. PROBLEM ROOT CAUSE RC

The need of the solution is to reduce the time of the manual process and even to expand the accessibility region so the beneficiary will increase.

7. BEHAVIOUR BE

The customer will go up to an hospital for donating the blood / Need of blood for the surgery but now they can use our application to do it and documentation work can be completed via online portal and dates for the transfer can be booked

Focus on J&P, tap into BE, understand RC

3. TRIGGERS TR

The need for the blood within a certain time limit can make the user to use our application

10. YOUR SOLUTION SL

Our solution is to build an application where blood donation can be done / even the requisition of blood can also be done with proper verifcation and documentation of all the work that has been and will be done.

8.CHANNELS of BEHAVIOUR CH

8.1 ONLINE

The customer needs to register themselves in the application and then do all the documentation and verifcation work.

8.2 OFFLINE

Physically need to go and donate the blood and do the manual process which can't be avoided

4. Requirement Analysis:

4.1 Functional Requirement:

Following are the functional requirements of the proposed solution.

FR No.	FunctionalRequirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Application Registration through Gmail Registration through LinkedIn Certain details must be submitted such as e-mail address, password, and password confirmation.
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP The login screen is used to verify the identity of the user. The account can be accessed using the user's registered email address and password.
FR-3	User login	Login using Registered email Id Operator has registered then the software operator should be able to login to the web application. The login information will be stored on the database for future use.
FR-4	Searching/reporting requirements	Users can use the search bar to look up information about camps and other topics.
FR-5	User Plasma Request	Should be able to request plasma in an emergency, software operators need to define plasma group,location,require data, contact. The plasma request will be sent to the plasma bank and then Inventory to check the availability. Users can request to donate plasma by filling out the request form on the page. Once the request is submitted, they will notified through email.

FR-6	Plasma stock	Receiving the plasma request from the clinic the plasma stock in the plasma bank Inventory will be searched to match the requested plasma request. Thus, match plasma units will be sent to the clinic.
FR-7	Statistical data	The availability of plasma is given in the page as stats, which will be helpful for the users.
FR-8	Certification	After the donor donates plasma, we will give them a digital certificate of appreciation and authentication.
FR-9	Distribution status/ View donation camps	If the distribution seems to be delayed then the clinic manager must be able to call the distribution person to get the update revise on the distribution. View the list of donation camps happening nearby.
FR-10	User Verification	User credentials are verified.
FR-11	Virtual Assistants(Chatbot)	A virtual assistant is a software agent that can carry out tasks or provide services on behalf of a person in response to commands or inquiries. When users enter their inquiries, the system will respond with pertinent information about plasma and details of plasma donation.
FR-12	Donor & Recipient Confirmation	Donor & Recipient are allocated to a certain time.

4.2 Non-Functional Requirements:

Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	The cost of the plasma units are standardized. The user interface of the plasma donor system must Be well-designed and welcoming.
NFR-2	Security	The application prevents the donors and recipients' data from being hijacked or misused. Data storage is required to have high security systems, just like it is by many other applications. Databases can keep all the donor information that is viewed by applications. It must be secured with email Id and password.

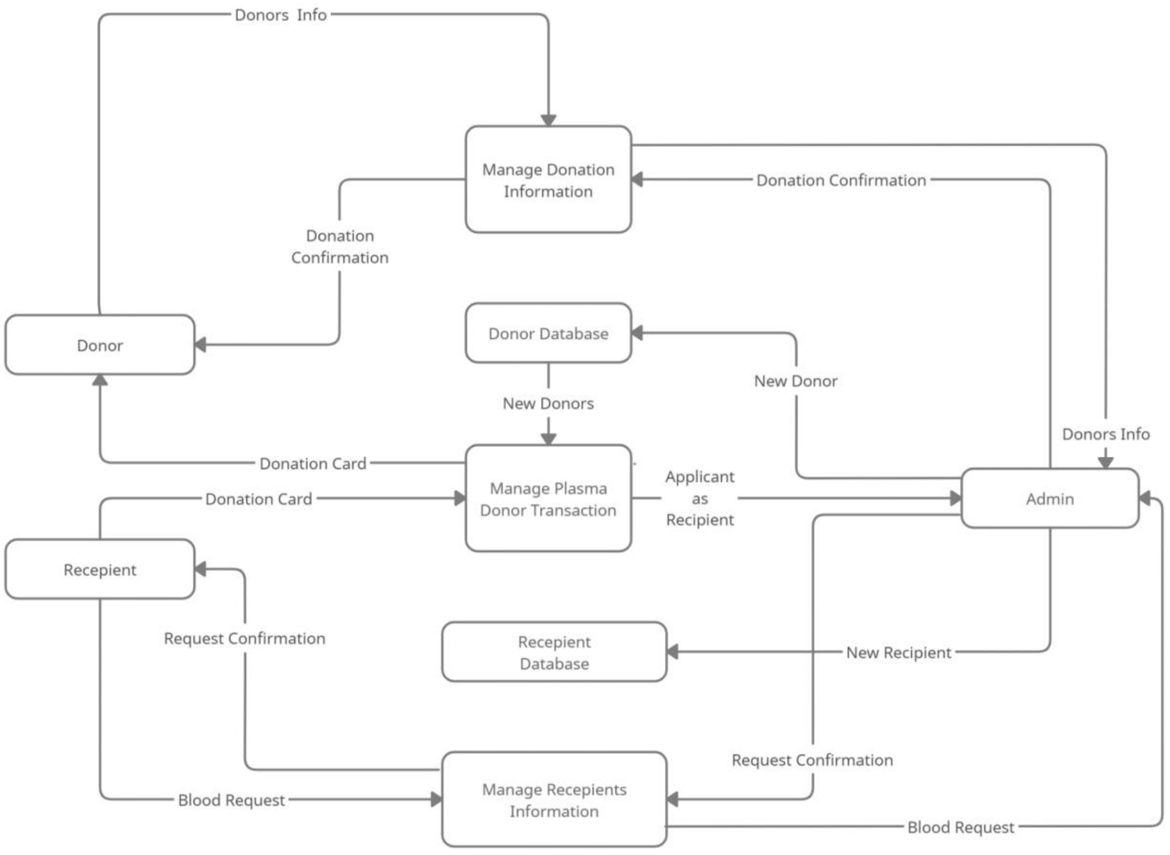
NFR-3	Reliability	The system has the ability to work all the times without failures apart from network failure. A donor can have the faith on the system. The authorities will keep the privacy of all donors in a proper manner. The application works under specific need of plasma in required time.
NFR-4	Performance	The system is interactive and the delays involved are less. When connecting to the server the delay is based on the distance of the 2 systems and the configuration between them so there is high probability that there will be or not a successful connection in less than 20 seconds for the sake of good communication. The Plasma donor System must perform well in different scenarios. The system is interactive and delays involved are

		less.
		The application tries to provide quick responses to the recipients.
NFR-5	Availability	The application runs properly and meets the user requirements. The system including the online and offline components should be available 24/7. The system should be available all times, meaning the user can access it is using application. In case if a hardware failure or database corruption, a replacement page will show. Also, in case of a hardware failure or database corruption, backups of the database should be retrieved from the application data folder and saved by the administrator.
NFR-6	Scalability	The application should can handle growing numbers of users and load without

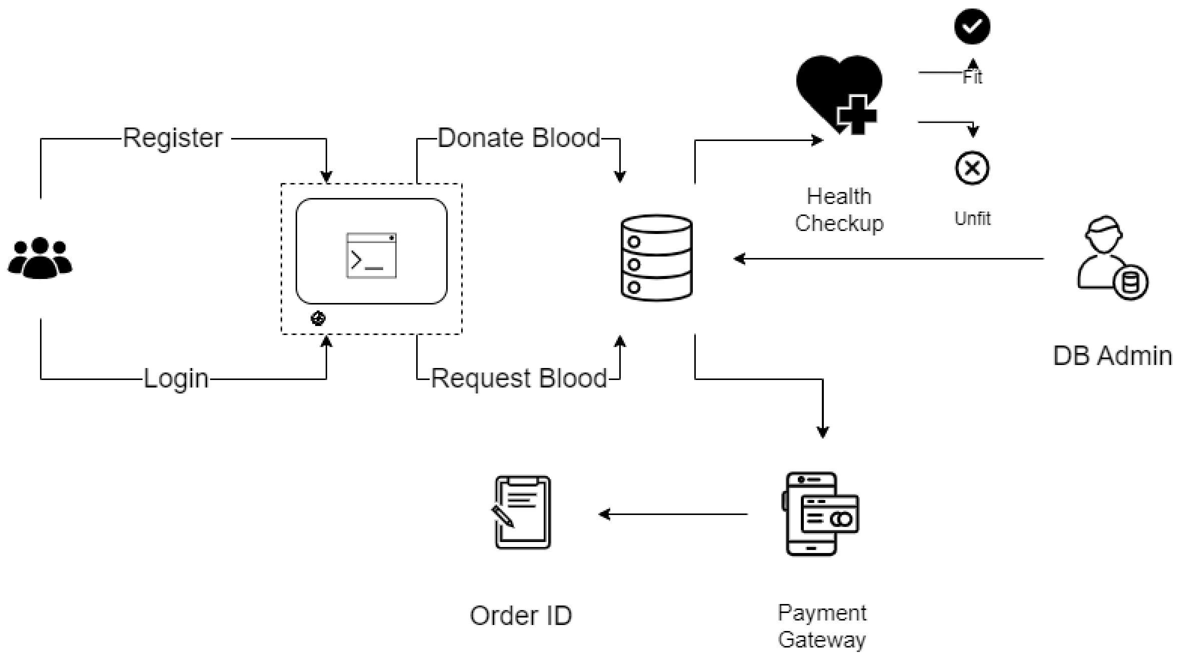
		compromising on performance and causing disruptions to user experience. The system offers the proper resources for issue solutions and is designed to protect sensitive information during all phases of operation In the application to handle an increase in workload without Performance degradation, or its ability to quickly enlarge. The solution must allow the hardware end of the deployed software services and components to be scaled horizontally as well as vertically.

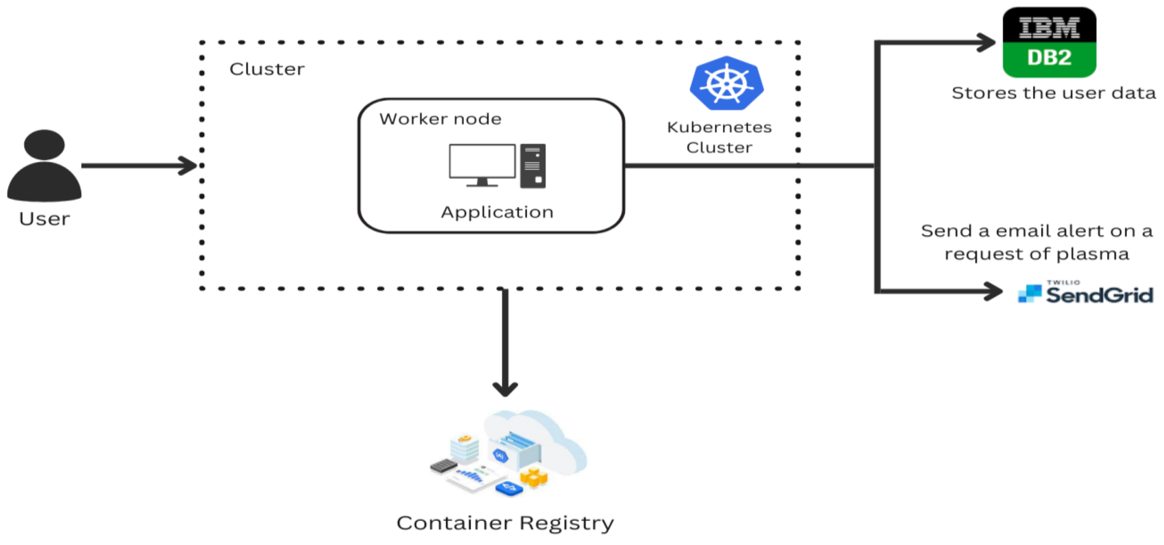
5. Project Design:

5.1 Data Flow Diagrams:

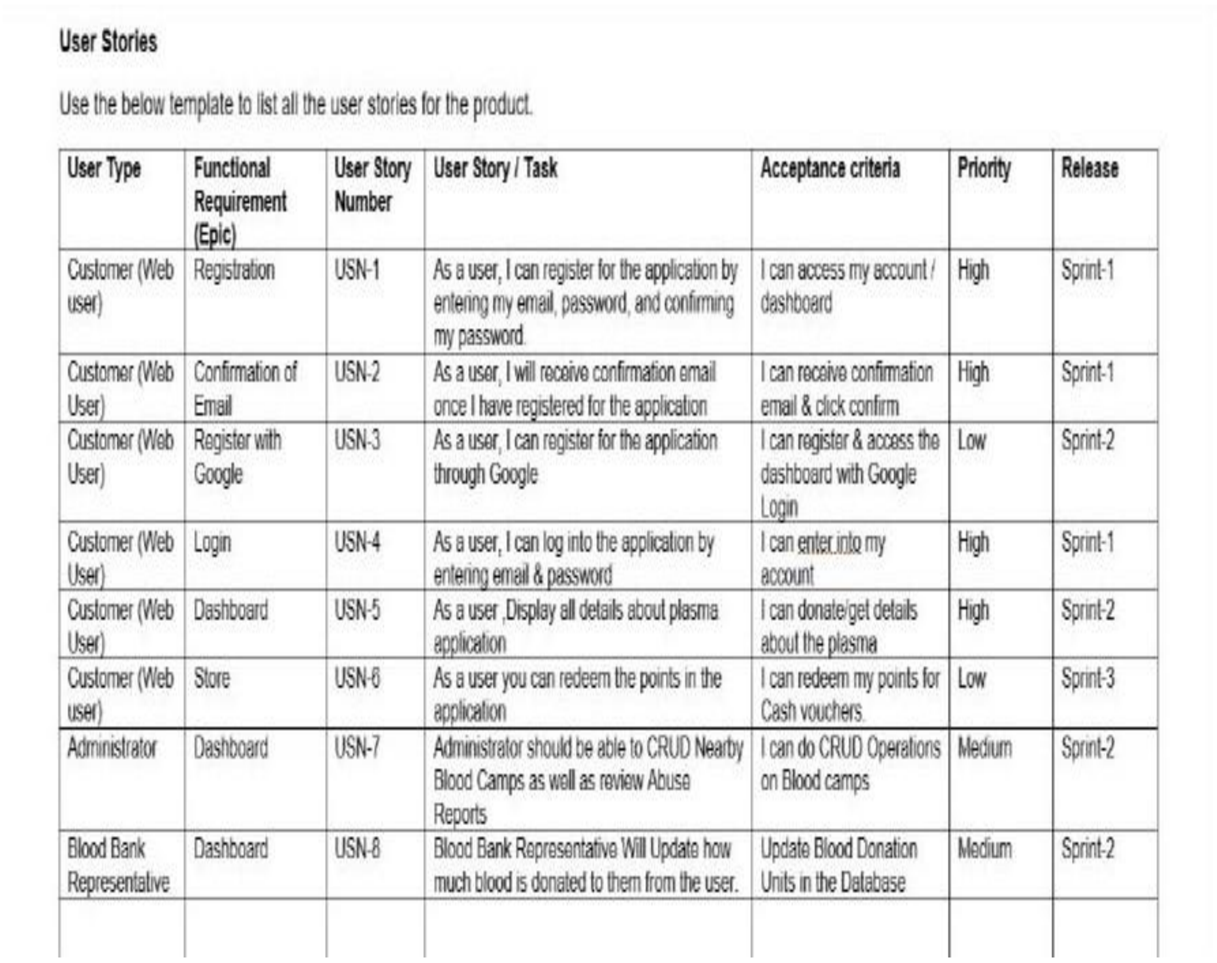


5.2 Solution and Technical Architecture:





5.3 User Stories:



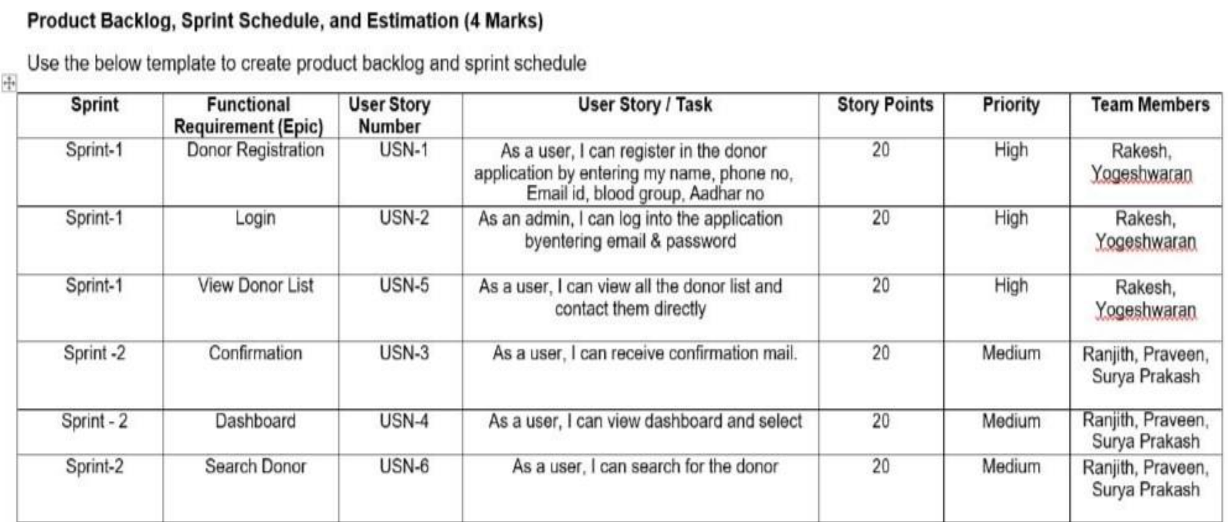
User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Web user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
Customer (Web User)	Confirmation of Email	USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
Customer (Web User)	Register with Google	USN-3	As a user, I can register for the application through Google	I can register & access the dashboard with Google Login	Low	Sprint-2
Customer (Web User)	Login	USN-4	As a user, I can log into the application by entering email & password	I can enter into my account	High	Sprint-1
Customer (Web User)	Dashboard	USN-5	As a user ,Display all details about plasma application	I can donate/get details about the plasma	High	Sprint-2
Customer (Web user)	Store	USN-6	As a user you can redeem the points in the application	I can redeem my points for Cash vouchers	Low	Sprint-3
Administrator	Dashboard	USN-7	Administrator should be able to CRUD Nearby Blood Camps as well as review Abuse Reports	I can do CRUD Operations on Blood camps	Medium	Sprint-2
Blood Bank Representative	Dashboard	USN-8	Blood Bank Representative Will Update how much blood is donated to them from the user.	Update Blood Donation Units in the Database	Medium	Sprint-2

6. Project Planning and Scheduling:

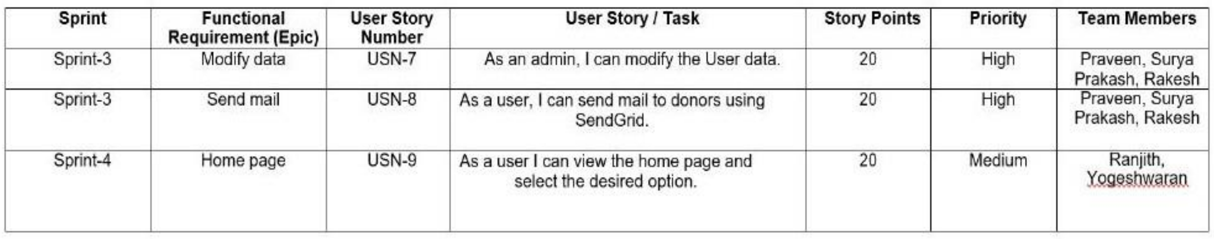
6.1 Sprint Planning and Estimation:



Product Backlog, Sprint Schedule, and Estimation (4 Marks)

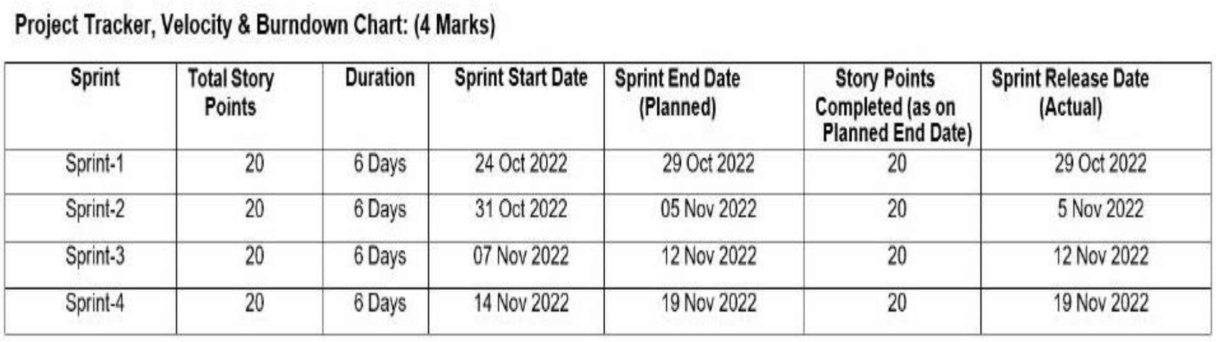
Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Donor Registration	USN-1	As a user, I can register in the donor application by entering my name, phone no, Email id, blood group, Aadhar no	20	High	Rakesh, Yogeshwaran
Sprint-1	Login	USN-2	As an admin, I can log into the application byentering email & password	20	High	Rakesh, Yogeshwaran
Sprint-1	View Donor List	USN-5	As a user, I can view all the donor list and contact them directly	20	High	Rakesh, Yogeshwaran
Sprint -2	Confirmation	USN-3	As a user, I can receive confirmation mail.	20	Medium	Ranjith, Praveen, Surya Prakash
Sprint - 2	Dashboard	USN-4	As a user, I can view dashboard and select	20	Medium	Ranjith, Praveen, Surya Prakash
Sprint-2	Search Donor	USN-6	As a user, I can search for the donor	20	Medium	Ranjith, Praveen, Surya Prakash



Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-3	Modify data	USN-7	As an admin, I can modify the User data.	20	High	Praveen, Surya Prakash, Rakesh
Sprint-3	Send mail	USN-8	As a user, I can send mail to donors using SendGrid.	20	High	Praveen, Surya Prakash, Rakesh
Sprint-4	Home page	USN-9	As a user I can view the home page and select the desired option.	20	Medium	Ranjith, Yogeshwaran

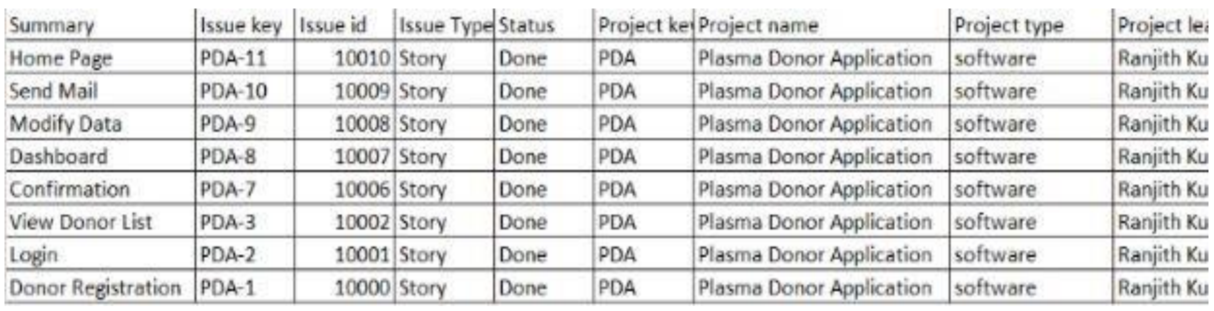
6.2 Sprint Delivery Schedule:



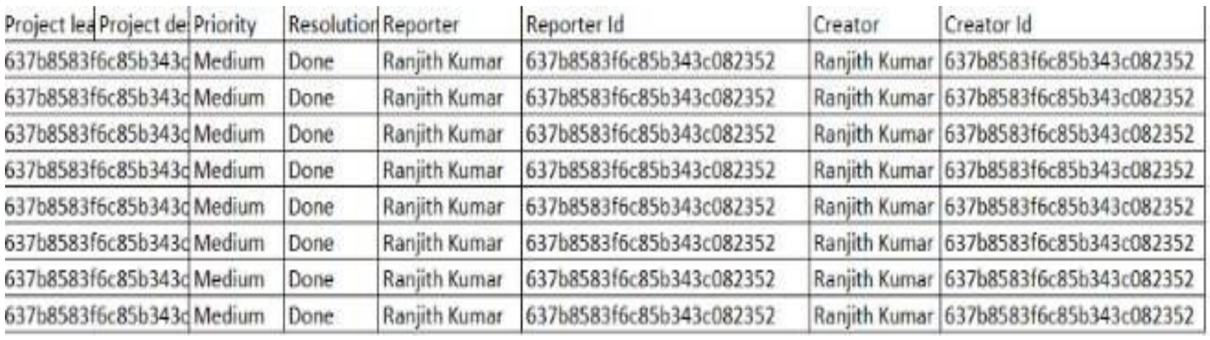
Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	5 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3 Reports from JIRA:



Summary	Issue key	Issue id	Issue Type	Status	Project ke	Project name	Project type	Project le
Home Page	PDA-11	10010	Story	Done	PDA	Plasma Donor Application	software	Ranjith Ku
Send Mail	PDA-10	10009	Story	Done	PDA	Plasma Donor Application	software	Ranjith Ku
Modify Data	PDA-9	10008	Story	Done	PDA	Plasma Donor Application	software	Ranjith Ku
Dashboard	PDA-8	10007	Story	Done	PDA	Plasma Donor Application	software	Ranjith Ku
Confirmation	PDA-7	10006	Story	Done	PDA	Plasma Donor Application	software	Ranjith Ku
View Donor List	PDA-3	10002	Story	Done	PDA	Plasma Donor Application	software	Ranjith Ku
Login	PDA-2	10001	Story	Done	PDA	Plasma Donor Application	software	Ranjith Ku
Donor Registration	PDA-1	10000	Story	Done	PDA	Plasma Donor Application	software	Ranjith Ku



Project lea	Project de	Priority	Resolution	Reporter	Reporter Id	Creator	Creator Id
637b8583f6c85b343c		Medium	Done	Ranjith Kumar	637b8583f6c85b343c082352	Ranjith Kumar	637b8583f6c85b343c082352
637b8583f6c85b343c		Medium	Done	Ranjith Kumar	637b8583f6c85b343c082352	Ranjith Kumar	637b8583f6c85b343c082352
637b8583f6c85b343c		Medium	Done	Ranjith Kumar	637b8583f6c85b343c082352	Ranjith Kumar	637b8583f6c85b343c082352
637b8583f6c85b343c		Medium	Done	Ranjith Kumar	637b8583f6c85b343c082352	Ranjith Kumar	637b8583f6c85b343c082352
637b8583f6c85b343c		Medium	Done	Ranjith Kumar	637b8583f6c85b343c082352	Ranjith Kumar	637b8583f6c85b343c082352
637b8583f6c85b343c		Medium	Done	Ranjith Kumar	637b8583f6c85b343c082352	Ranjith Kumar	637b8583f6c85b343c082352
637b8583f6c85b343c		Medium	Done	Ranjith Kumar	637b8583f6c85b343c082352	Ranjith Kumar	637b8583f6c85b343c082352
637b8583f6c85b343c		Medium	Done	Ranjith Kumar	637b8583f6c85b343c082352	Ranjith Kumar	637b8583f6c85b343c082352

7. Coding and Solutioning:

7.1 Features:

SendGrid

SendGrid service integrate in minutes with our email API and trust your emails reach the inbox

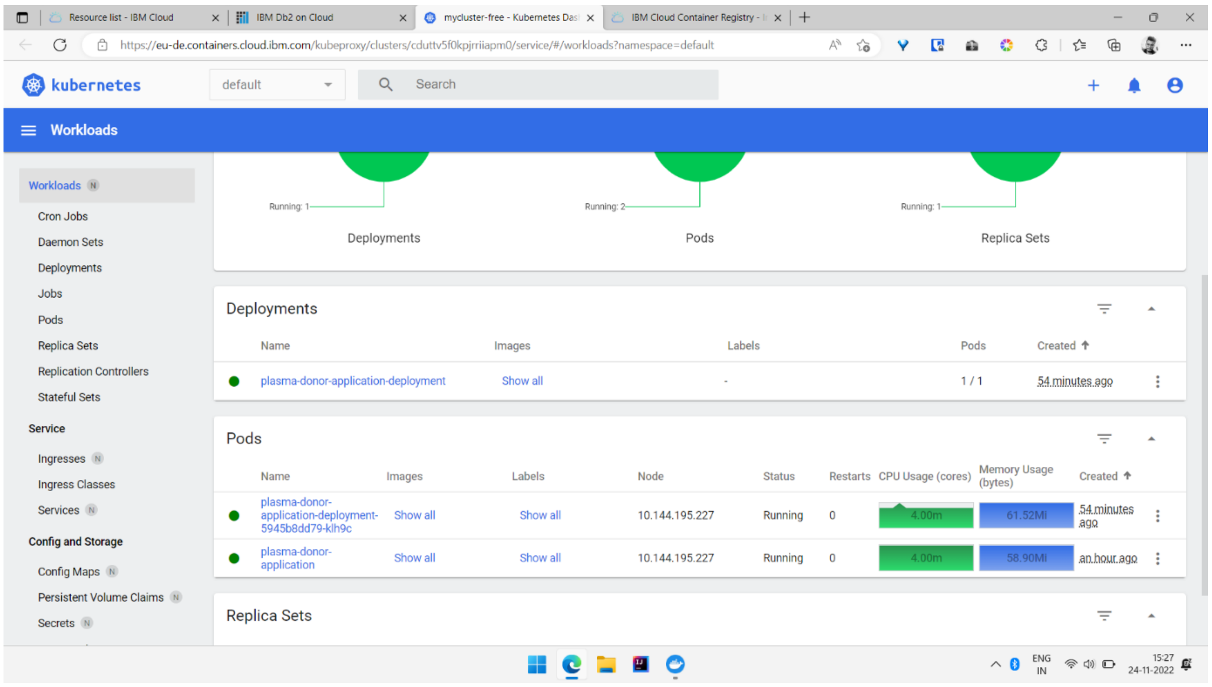
```
# sendgrid integration
def mailtest_registration(to_email):
    sg = sendgrid.SendGridAPIClient(api_key= 'SG.9_tPZuieRP-
tHkezgkD_ZA.qpw1oJcv4Ig6fT-Vz4mIMVbdnJ5HPPfcvlDyacxC-iE' )
    from_email = Email("rakeshprasanna72@gmail.com")
    subject = "Registration Successfull!"
    content = Content("text/plain", "You have successfully registered as user.
Please Login using your Username and Password to donate/request for Plasma.")
    mail = Mail(from_email, to_email, subject, content)
    response = sg.client.mail.send.post(request_body=mail.get())
    print(response.status_code)
    print(response.body)
    print(response.headers)

#for donor
def mailtest_donor(to_email):
    sg = sendgrid.SendGridAPIClient(api_key= 'SG.9_tPZuieRP-
tHkezgkD_ZA.qpw1oJcv4Ig6fT-Vz4mIMVbdnJ5HPPfcvlDyacxC-iE' )
    from_email = Email("rakeshprasanna72@gmail.com")
    subject = "Thankyou for Registering as Donor!"
    content = Content("text/plain", "Every donor is an asset to the nation who
saves people's lives, and you're one of them.We appreciate your efforts. Thank
you!!")
    mail = Mail(from_email, to_email, subject, content)
    response = sg.client.mail.send.post(request_body=mail.get())
    print(response.status_code)
    print(response.body)
    print(response.headers)
```

```
#for request

def mailtest_request(to_email):
    sg = sendgrid.SendGridAPIClient(api_key= 'SG.9_tPZuieRP-
tHkezgkD_ZA.qpw1oJcv4Ig6fT-Vz4mIMVbdnJ5HPPfcvlDyacxC-iE' )
    from_email = Email("rakeshprasanna72@gmail.com")
    subject = "Request Submitted!"
    content = Content("text/plain", "Your request has been successfully
submitted. Please be patient, your requested donor will get back to you
soon.")
    mail = Mail(from_email, to_email, subject, content)
    response = sg.client.mail.send.post(request_body=mail.get())
    print(response.status_code)
    print(response.body)
    print(response.headers)

#for request sending to donor

def mailtest_requesttodonor(to_email):
    sg = sendgrid.SendGridAPIClient(api_key= 'SG.9_tPZuieRP-
tHkezgkD_ZA.qpw1oJcv4Ig6fT-Vz4mIMVbdnJ5HPPfcvlDyacxC-iE' )
    from_email = Email("rakeshprasanna72@gmail.com")
    subject = "Requesting Plasma"
    content = Content("text/plain", "Your registration has been requested by a
recipient, we will share futher details in future. Stay connected!!")
    mail = Mail(from_email, to_email, subject, content)
    response = sg.client.mail.send.post(request_body=mail.get())
    print(response.status_code)
    print(response.body)
    print(response.headers)
```

7.2 Features:

Kubernetes:

Kubernetes has been used to deploy the application we built to the IBM Cloud



https://eu-de.containers.cloud.ibm.com/kubeproxy/clusters/cduttv5f0kpjrriiapm0/service/#/workloads?namespace=default

Running: 1 Deployments
Running: 2 Pods
Running: 1 Replica Sets

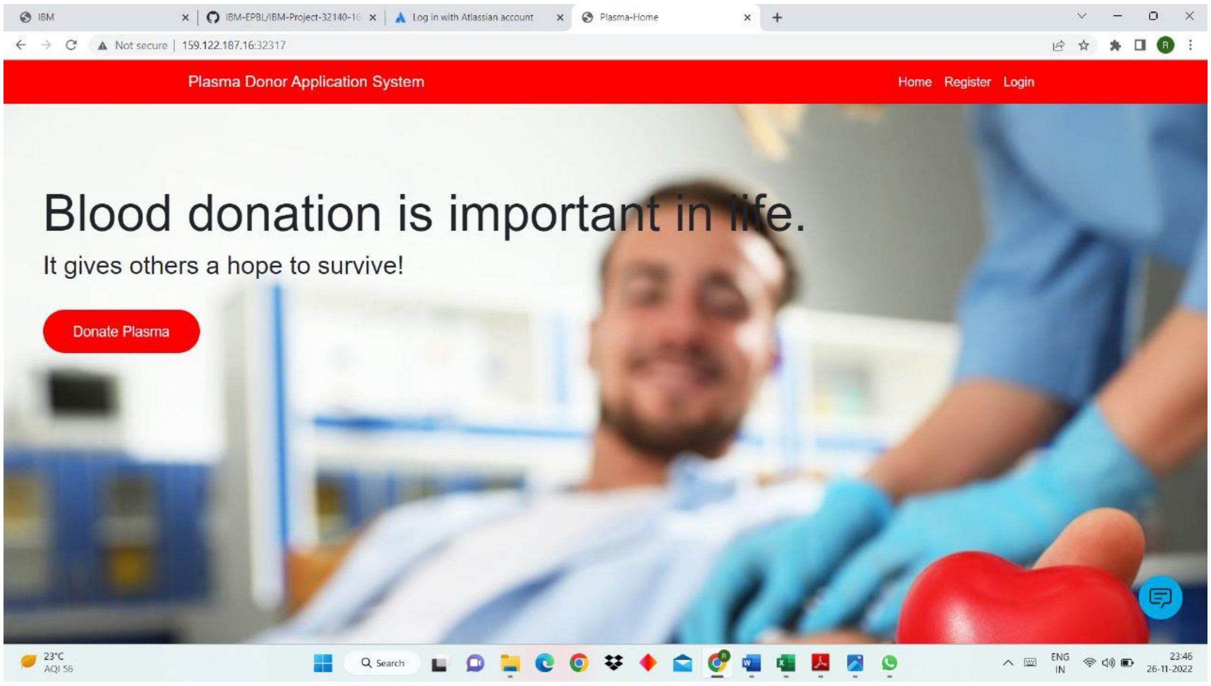
Deployments

Name	Images	Labels	Pods	Created
plasma-donor-application-deployment	Show all	-	1 / 1	54 minutes ago

Pods

Name	Images	Labels	Node	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)	Created
plasma-donor-application-deployment-5945b8dd79-klh9c	Show all	Show all	10.144.195.227	Running	0	4.00m	61.52Mi	54 minutes ago
plasma-donor-application	Show all	Show all	10.144.195.227	Running	0	4.00m	58.90Mi	an hour ago

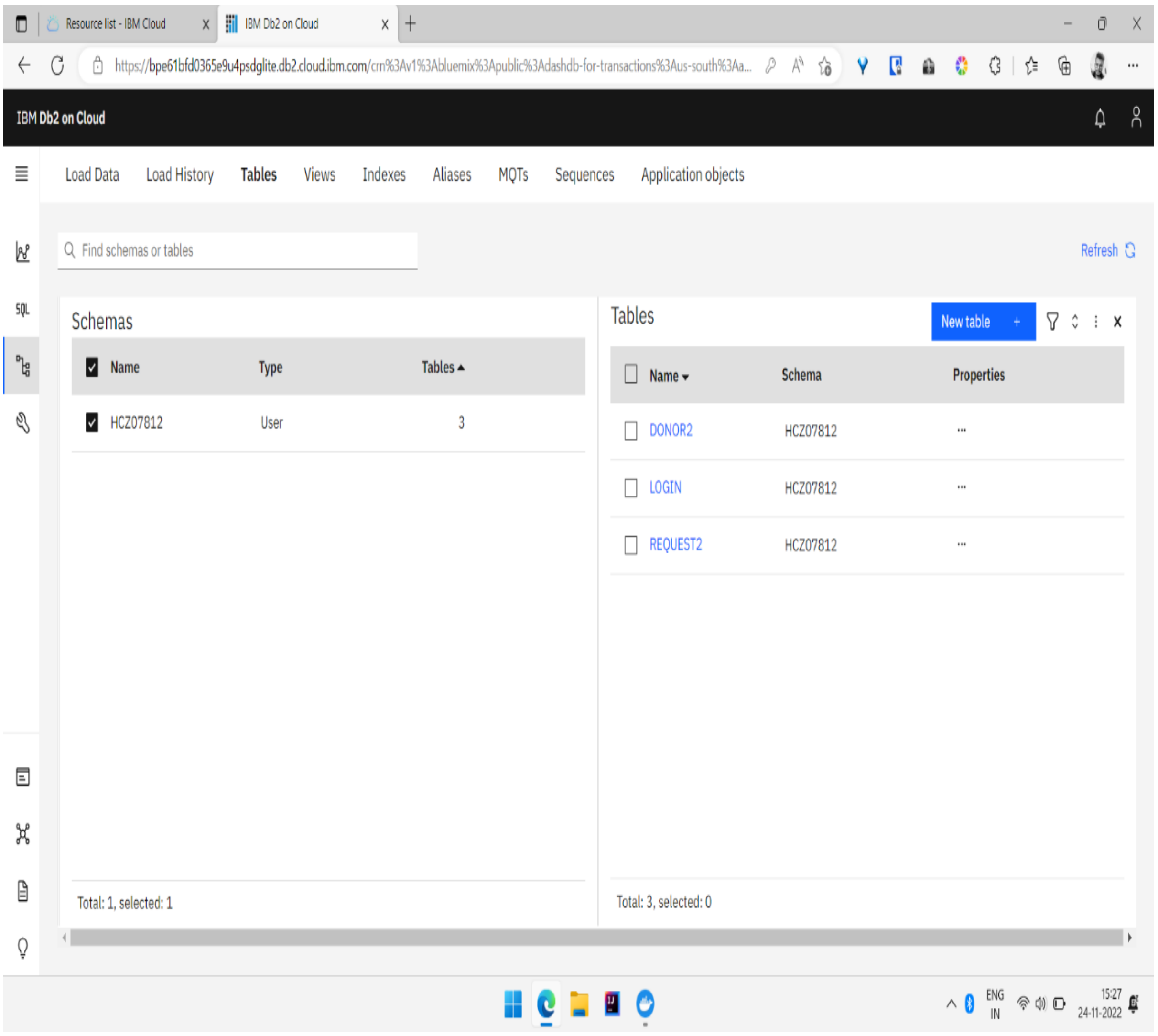
Replica Sets


Not secure | 159.122.187.16:32317
Plasma Donor Application System
Home
Register
Login
Blood donation is important in life.
It gives others a hope to survive!
Donate Plasma

7.3 Database Schema(if Applicable):

Database

IBM Cloud Database help to integrate data from different sources across on-premises and cloud environments.



Schemas

Name	Type	Tables ▲
✓ HCZ07812	User	3

Total: 1, selected: 1

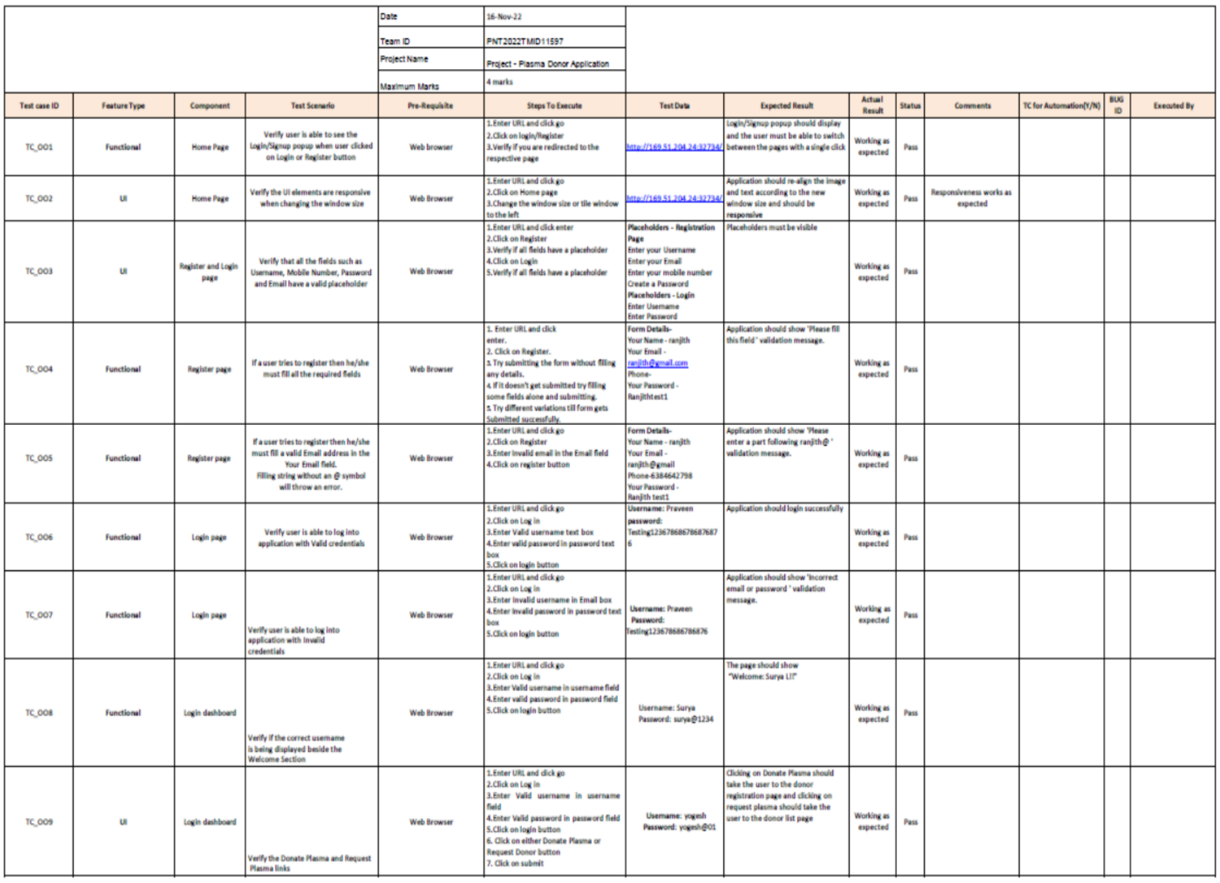
Tables

Name ▾	Schema	Properties
DONOR2	HCZ07812	...
LOGIN	HCZ07812	...
REQUEST2	HCZ07812	...

Total: 3, selected: 0

8. Testing:

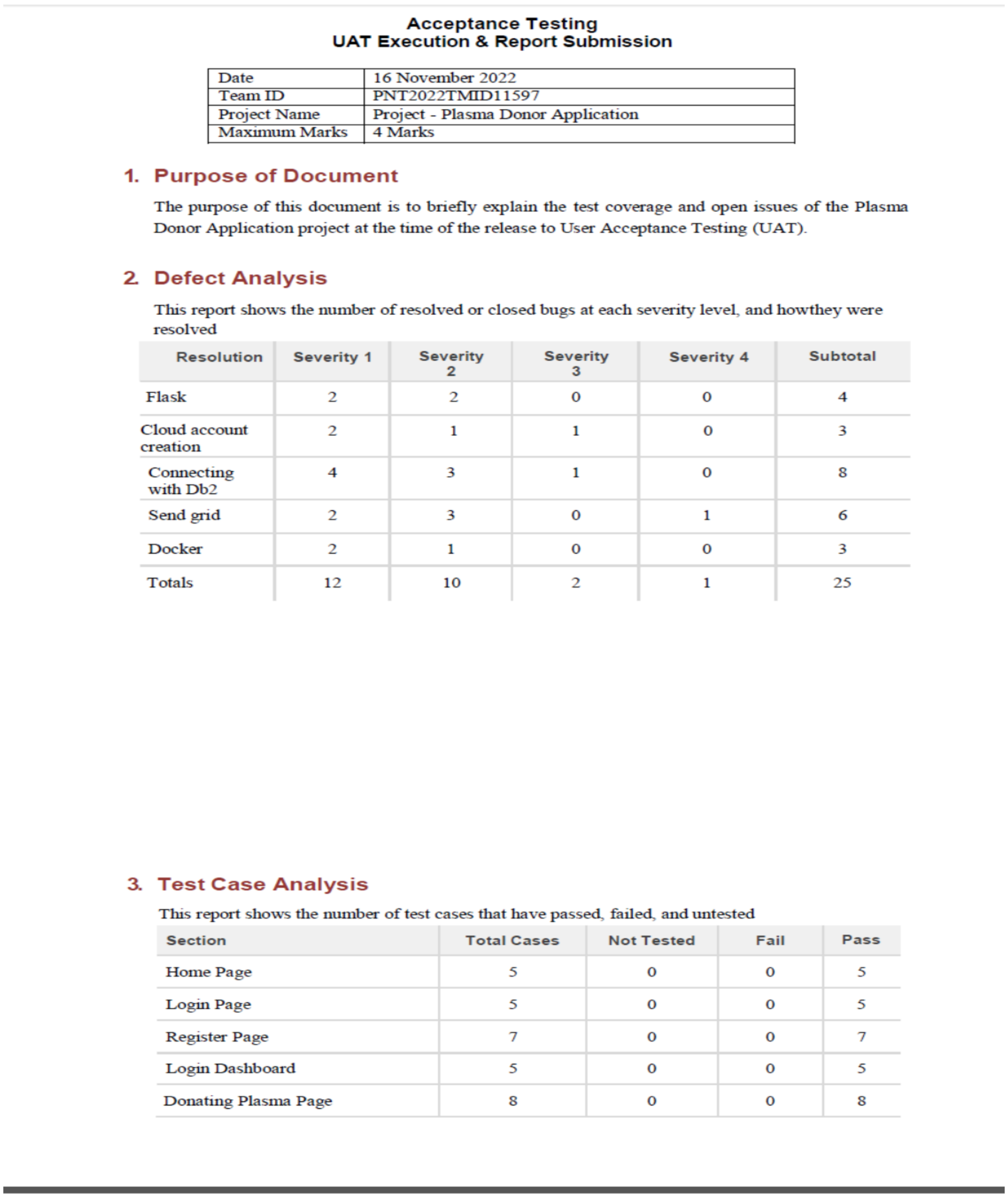
8.1 Test Cases:



Date	16-Nov-22
Team ID	PNT2022TMID11597
Project Name	Project - Plasma Donor Application
Maximum Marks	4 marks

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
TC_OO1	Functional	Home Page	Verify user is able to see the Login/Signup popup when user clicked on Login or Register button	Web browser	1.Enter URL and click go 2.Click on login/Register 3.Verify if you are redirected to the respective page	http://169.51.204.24:32734/	Login/Signup popup should display and the user must be able to switch between the pages with a single click	Working as expected	Pass				
TC_OO2	UI	Home Page	Verify the UI elements are responsive when changing the window size	Web Browser	1.Enter URL and click go 2.Click on Home page 3.Change the window size or tile window to the left	http://169.51.204.24:32734/	Application should re-align the image and text according to the new window size and should be responsive	Working as expected	Pass	Responsiveness works as expected			
TC_OO3	UI	Register and Login page	Verify that all the fields such as Username, Mobile Number, Password and Email have a valid placeholder	Web Browser	1.Enter URL and click enter 2.Click on Register 3.Verify if all fields have a placeholder 4.Click on Login 5.Verify if all fields have a placeholder	**Placeholders - Registration Page** Enter your Username Enter your Email Enter your mobile number Create a Password **Placeholders - Login** Enter Username Enter Password	Placeholders must be visible	Working as expected	Pass				
TC_OO4	Functional	Register page	If a user tries to register then he/she must fill all the required fields	Web Browser	1. Enter URL and click enter. 2. Click on Register. 3. Try submitting the form without filling any details. 4. If it doesn't get submitted try filling some fields alone and submitting. 5. Try different variations till form gets Submitted successfully.	Form Details- Your Name - ranjith Your Email - ranjith@gmail.com Phone- Your Password - Ranjithtest1	Application should show 'Please fill this field ' validation message.	Working as expected	Pass				
TC_OO5	Functional	Register page	If a user tries to register then he/she must fill a valid Email address in the Your Email field. Filling string without an @ symbol will throw an error.	Web Browser	1.Enter URL and click go 2.Click on Register 3.Enter invalid email in the Email field 4.Click on register button	Form Details- Your Name - ranjith Your Email - ranjith@gmail Phone-6384642798 Your Password - Ranjith test1	Application should show 'Please enter a part following ranjith@ ' validation message.	Working as expected	Pass				
TC_OO6	Functional	Login page	Verify user is able to log into application with Valid credentials	Web Browser	1.Enter URL and click go 2.Click on Log in 3.Enter Valid username text box 4.Enter valid password in password text box 5.Click on login button	Username: Praveen password: Testing123678686786876876	Application should login successfully	Working as expected	Pass				
TC_OO7	Functional	Login page	Verify user is able to log into application with invalid credentials	Web Browser	1.Enter URL and click go 2.Click on Log in 3.Enter invalid username in Email box 4.Enter invalid password in password text box 5.Click on login button	Username: Praveen Password: Testing12367868678687б	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass				
TC_OO8	Functional	Login dashboard	Verify if the correct username is being displayed beside the Welcome Section	Web Browser	1.Enter URL and click go 2.Click on Log in 3.Enter Valid username in username field 4.Enter valid password in password field 5.Click on login button	Username: Surya Password: surya@1234	The page should show "Welcome: Surya L!"	Working as expected	Pass				
TC_OO9	UI	Login dashboard	Verify the Donate Plasma and Request Plasma links	Web Browser	1.Enter URL and click go 2.Click on Log in 3.Enter Valid username in username field 4.Enter Valid password in password field 5.Click on login button 6. Click on either Donate Plasma or Request Donor button 7. Click on submit	Username: yogesh Password: yogesh@01	Clicking on Donate Plasma should take the user to the donor registration page and clicking on request plasma should take the user to the donor list page	Working as expected	Pass				

8.2 User Acceptance Testing:



Acceptance Testing
UAT Execution & Report Submission

Date	16 November 2022
Team ID	PNT2022TMID11597
Project Name	Project - Plasma Donor Application
Maximum Marks	4 Marks

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the Plasma Donor Application project at the time of the release to User Acceptance Testing (UAT).

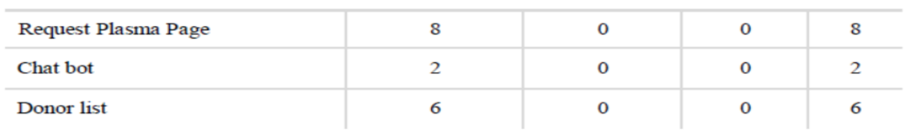
2. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and howthey were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
Flask	2	2	0	0	4
Cloud account creation	2	1	1	0	3
Connecting with Db2	4	3	1	0	8
Send grid	2	3	0	1	6
Docker	2	1	0	0	3
Totals	12	10	2	1	25

3. Test Case Analysis

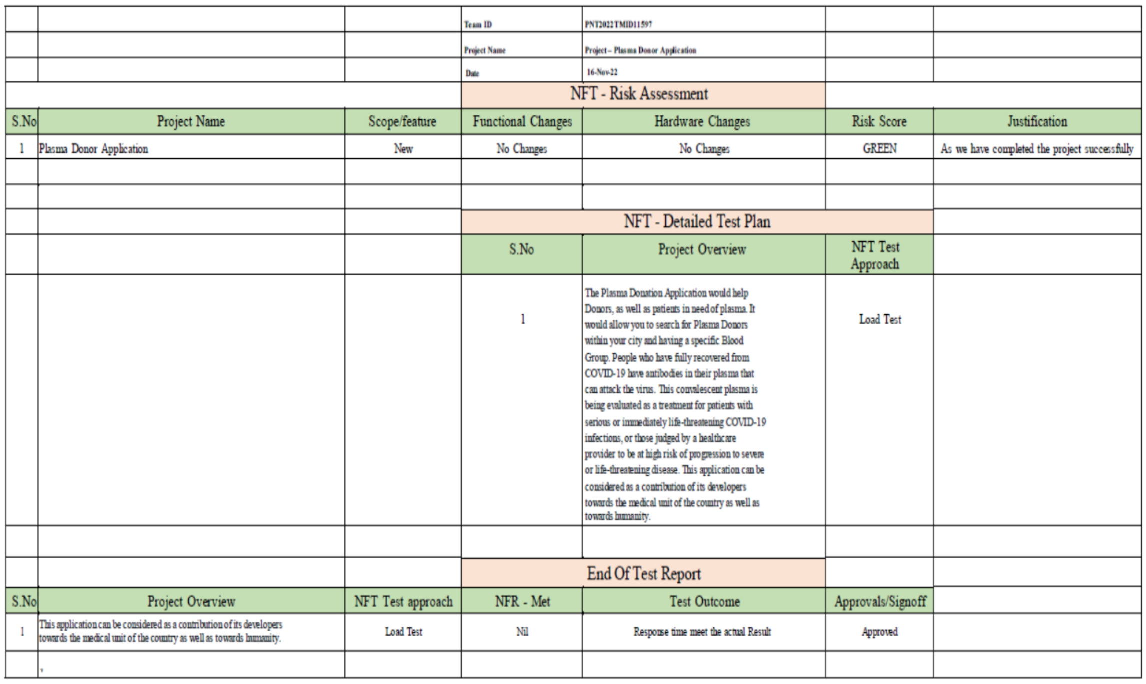
This report shows the number of test cases that have passed, failed, and untested



Section	Total Cases	Not Tested	Fail	Pass
Home Page	5	0	0	5
Login Page	5	0	0	5
Register Page	7	0	0	7
Login Dashboard	5	0	0	5
Donating Plasma Page	8	0	0	8
Request Plasma Page	8	0	0	8
Chat bot	2	0	0	2
Donor list	6	0	0	6

9. Results:

9.1 Performance Metrics:



			Team ID	PNT2022TMID11597		
			Project Name	Project – Plasma Donor Application		
			Date	16-Nov-22		
			NFT - Risk Assessment			
S.No	Project Name	Scope/feature	Functional Changes	Hardware Changes	Risk Score	Justification
1	Plasma Donor Application	New	No Changes	No Changes	GREEN	As we have completed the project successfully
			NFT - Detailed Test Plan			
			S.No	Project Overview	NFT Test Approach	
			1	The Plasma Donation Application would help Donors, as well as patients in need of plasma. It would allow you to search for Plasma Donors within your city and having a specific Blood Group. People who have fully recovered from COVID-19 have antibodies in their plasma that can attack the virus. This convalescent plasma is being evaluated as a treatment for patients with serious or immediately life-threatening COVID-19 infections, or those judged by a healthcare provider to be at high risk of progression to severe or life-threatening disease. This application can be considered as a contribution of its developers towards the medical unit of the country as well as towards humanity.	Load Test	
			End Of Test Report			
S.No	Project Overview	NFT Test approach	NFR - Met	Test Outcome	Approvals/Signoff	
1	This application can be considered as a contribution of its developers towards the medical unit of the country as well as towards humanity.	Load Test	Nil	Response time meet the actual Result	Approved	

10. Advantages and Disadvantages:

10.1 Advantages:

- The main advantage is that it is relatively simple way to collect data from many people quickly and at zero cost.
- Good Validity - people can fulfil and request their needs directly .
- A second advantage is that data can be collected in various ways to suit the researcher’s needs.
- The application can collect data from a large number of people and stored in the database.
- It helps people to help others who has medical needs.
- It is a relatively safe process.

10.2 Disadvantages:

- The main disadvantage is that questionnaires might be the possibility of providing invalid answers. Fixed choice questions lack flexibility.
- There is a chance that some questions will be ignored or left unanswered.
- Self-reported answers may be exaggerated; respondents may be too embarrassed to reveal private details.
- Low response rate.

11. Conclusion:

PLASMA DONOR APPLICATION this project "PLASMA DONOR" deals with notifying the concerned donor upon request by the Recipient in need of Plasma. This project provides quick access to donors for an immediate requirement of blood. In case of an emergency/surgery, blood procurement is always a major problem which consumes a lot of time. This helps serve the major time-lapse in which a life can be saved!

12. Future Scope:

The Plasma Donation App would help Donors, as well as patients in need of plasma. It would allow you to search Plasma Donors within your city and having a specific Blood Group. People who have fully recovered from COVID-19 have antibodies in their plasma that can attack the virus. The proposed plasma Donating Web Application project could ensure the necessity of plasma and plasma donation by saving the World.

13. Appendix:

Source code:

admin_login.html:

```
{% extends 'base.html'%}

<!--title tag-->
{% block title %}
<title>Admin-LogIn</title>
{% endblock %}
```

```
<!---Login Content-->
{% block content %}
<!---Registration form-->
<div class="container">
    <div class="text-center mt-5"><h2>LogIn as Admin</h2></div>

</div>

<div class="container mt-5">
    <div class="row justify-content-center">
        <div class="col-sm-6 ">
            <div class="card">
                <div class="card-body">
                    <!----Form content---->
                    <form action="/" method="post">

                        <div class="form-group">
                            <label for="email">Email</label>
                            <input type="email" class="form-control" name=""
id="email" required placeholder="Enter your Email">
                        </div>
                        <div class="form-group">
                            <label for="password">Password</label>
                            <input type="password" class="form-control"
name="" id="password" placeholder="Enter Password" required>
                        </div>





                        <!--button-->
                        <div class="form-group text-center">

                            <button type="submit" class="btn btn-
success">Submit</button>
                        </div>


                    </form>
                </div>
            </div>
        </div>
    </div>
</div>

{% endblock %}
```

base.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no">
    <!--font awesome-->
    <script src="https://kit.fontawesome.com/15af226b72.js"
crossorigin="anonymous"></script>

    <!-- Bootstrap CSS -->
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/css/bootstrap.min.css"
integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">

    <!--Google Font-->
    <style>
        .card {
            border-radius: 25px !important;
            background-color: rgba(255, 255, 255, 0.141) !important;
            backdrop-filter: blur(5px) !important;
        }
        @import
url('https://fonts.googleapis.com/css2?family=Montserrat:ital,wght@0,100;0,200
;0,300;0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600
;1,700;1,800;1,900&display=swap');
    </style>
    <!--contains style for all pages-->
    <link rel="stylesheet" href="{{ url_for('static',filename='style.css')}}">
    <script src="https://kit.fontawesome.com/000fb23390.js"
crossorigin="anonymous"></script>
    {% block link %}
    {% endblock %}
    {% block title %}
    {% endblock %}
</head>
<body>

    <!-- Header -->
    <header>
<nav class="navbar navbar-expand-lg navbar-dark bg-primary">
    <div class="container">
    <a href="{{ url_for('index')}}" class="navbar-brand"><i class="fa-solid 
fa-droplet" id="icon"></i>
        Plasma Donor Application System</a>

    <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#myNavBar"
        aria-controls="myNavBar" aria-expanded="false" aria-label="Toggle 
navigation">

        <span class="navbar-toggler-icon"></span>
    </button>

    <div class="collapse navbar-collapse" id="myNavBar">
        <ul class="navbar-nav ml-auto">
            <li class="nav-item active">
                <a href="{{ url_for('home_page') }}" class="nav-link">Home</a>
            </li>


            <li class="nav-item">
                <a href="{{ url_for('signin') }}" class="nav-link" 
style="color:white ;">Register</a>
            </li>
            <li class="nav-item">
                <a href="{{ url_for('login') }}" class="nav-link" 
style="color:white ;">Login</a>
            </li>

            </ul>
        </div>
    </div>

    </nav>
</header>
    <!-- End Header -->



    <!--Future contents-->
    {% block content %}
    {% endblock %}


    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" 
integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" 
crossorigin="anonymous"></script>
```

```
    <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.14.7/dist/umd/popper.min.js"
integrity="sha384-
UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
crossorigin="anonymous"></script>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/js/bootstrap.min.js"
integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
crossorigin="anonymous"></script>
    <script>
        window.watsonAssistantChatOptions = {
          integrationID: "18cd43fb-1fde-4fb9-b9d2-a7c1095e980b", // The ID of
this integration.
          region: "au-syd", // The region your integration is hosted in.
          serviceInstanceID: "05ba06eb-03f7-412f-b60f-5a4a4f83de97", // The ID
of your service instance.
          onLoad: function(instance) { instance.render(); }
        };
        setTimeout(function(){
          const t=document.createElement('script');
          t.src="https://web-
chat.global.assistant.watson.appdomain.cloud/versions/" +
(window.watsonAssistantChatOptions.clientVersion || 'latest') +
"/WatsonAssistantChatEntry.js";
          document.head.appendChild(t);
        });
      </script>
</body>
</html>
```

donor.html:

```
{% extends 'base.html'%}

<!--title tag-->
{% block title %}
<title>Plasma-Donor</title>
<style>
  body{
   background-image:url('static\\bg.jpg');
   }
   .bg-primary{
     background-color:red !important;
   }
   .card-body{
    border-radius: 25px;
    background-color: rgba(255, 255, 255, 0.141);
    backdrop-filter: blur(5px);
   }
</style>
{% endblock %}

<!---Donor Content-->
{% block content %}
<!---Donor table-->
<div class="container mt-3">
    <div class="row justify-content-center">
        <div class="col-sm-12 ">
          <div class="msg">{{ msg }}</div>
            <div class="">
                <div class="">
                    <h6 style="text-align: center; margin-top: 50px;color:
red;">Note: Please note the donor email from the table you want to
request.</h6>
  <table class="table table-hover table-bordered" style="margin:100px 0px;
text-align: center;">
    <thead class="thead-light">
      <tr>

        <th scope="col">Email</th>
        <th scope="col">Age</th>
        <th scope="col">Gender</th>
        <th scope="col">Blood Group</th>
        <th scope="col">Area</th>
        <th scope="col">City</th>
        <th scope="col">District</th>
        <th scope="col">Make a Request</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        {% for row in donor2 %}
        <td>{{ row["EMAIL"] }}</td>
        <td>{{ row["AGE"] }}</td>
        <td>{{ row["GENDER"] }}</td>
        <td>{{ row["BLOOD"] }}</td>
        <td>{{ row["AREA"] }}</td>
        <td>{{ row["CITY"] }}</td>
        <td>{{ row["DISTRICT"] }}</td>
        <td>
        <a href="{{url_for('request_page')}}" class="btn-sm btn-info"
style="color: white; text-decoration:none">Request</a>
        </td>
      </tr>
      {% endfor %}
    </tbody>
  </table>
</div>
</div>
</div>
</div>
</div>
{% endblock %}
```

home.html:

```
{% extends 'base.html' %}

<!--title tag-->
{% block title %}
<title>Plasma-Home</title>
<style>
    body{
    background-image:url('static\\bg.jpg');
    }

    .heading{
        padding-top: 30px;
        text-align: center;
        font-weight: 500;
    }

    .profile-area{
    padding:30px 0;
    }
    .card{
      box-shadow: 0 0 30px rgba(0,0,0,0.1);
      overflow:hidden;
      border-radius:15px;
      margin-top:30px;
    }
    .img1 img{
        height:100px;
        margin-left:auto;
        margin-right:auto;
        /* border-top-right-radius:15px;
        border-top-left-radius:15px; */
        width:100%;
    }

    .img2 img{
      margin-left: auto;
      text-align: center;
      border-radius: 50%;
      width: 100px;
    }
    .card:hover .img2 img{
        border-color:bg-primary;
        transition:.7s
    }
    .main-text{
        padding: 30px 0;
        text-align:center;
        }
    .main-text h2{
        top:22px;
        text-transform:uppercase;
        font-weight: 900;
        font-size:20px;
        margin: 0 0 10px;
    }
    .main-text p{
        font-size:16px;
        padding: 0 35px;
    }
    .space{
    margin-bottom:20px;
    }

    .bg-primary{
      background-color:red !important;
    }
</style>
{% endblock %}

{% block link %}
<link rel="stylesheet" href="{{ url_for('static',filename='home.css')}}">

{% endblock %}

{% block content %}

    <div class="landing">

        <div class="landing-text" data-aos="fade-up" data-aos-duration="1000">
            <h1>Blood donation is important in life. <span style="color:
#e0501b; font-size: size 6vw;"></span></h1>
            <h2>It gives others a hope to survive!</h2>
            <div class="btn">
```

```
                <a href="{{ url_for('signin') }}" style="text-decoration:
none;">Donate Plasma</a>
            </div>
        </div>

    </div>
    </div>
    </div>
    </div>





{% endblock %}
```

login.html:

```
{% extends 'base.html'%}

<!--title tag-->
{% block title %}
<title>Plasma-LogIn</title>
<style>
    body{
     background-image:url('static\\bg.jpg');
     }
     .bg-primary{
       background-color:red !important;
     }
     .card-body{
        border-radius: 25px;
        background-color: rgba(255, 255, 255, 0.141);
        backdrop-filter: blur(5px);
     }
 </style>
{% endblock %}

<!---Login Content-->
{% block content %}
<!---Login form-->
<div class="container">
    <div class="text-center mt-5"><h3>LogIn using UserName and
Password</h3></div>

</div>

<div class="container mt-5">
```

```
    <div class="row justify-content-center">
        <div class="col-sm-6 ">
            <div class="card">
                <div class="card-body">

                    <!----Form content---->
                    <form action="/login" method="POST">
                        <div class="msg" style="color: green;">{{ msg }}</div>

                        <div class="form-group">
                            <label for="username">User Name</label>
                            <input type="text" class="form-control"
name="username" id="username" required placeholder="Enter UserName">
                        </div>
                        <div class="form-group">
                            <label for="password">Password</label>
                            <input type="password" class="form-control"
name="password" id="password" placeholder="Enter Password" required>
                        </div>


                        <!--button-->
                        <div class="form-group text-center">
                            <input type="submit" value="LogIn" class="btn btn-
success">
                        </div>
                        <br>
                        <div style="text-align: center;">
                            <p>Don't have an account <a href="{{
url_for('signin') }}">Register here</a>
                        </div>

                    </form>
                </div>
            </div>
        </div>
    </div>
</div>

{% endblock %}
```

register.html:

```
{% extends 'base.html' %}

<!--title tag-->
{% block title %}
<title>Plasma-Register</title>
<style>
    body{
     background-image:url('static\\bg.jpg');
     }
     .bg-primary{
       background-color:red !important;
     }
     .card-body{
        border-radius: 25px;
        background-color: rgba(255, 255, 255, 0.141);
        backdrop-filter: blur(5px);
     }
 </style>
{% endblock %}

{% block content %}

<!---Registration form-->

<div class="container mt-5<div class="row"<div class="col-sm-12"><div
class="card">
<div class="card-body">
        <form action="/adddonor" method="post">
        <h4 style="text-align: center;">Donating Plasma</h4>
        <div class="form-group ">
            <label for="name">Full Name</label>
            <input type="text" class="form-control" name="name" id="name"
placeholder="Enter your Name"
                            required>

        </div>
        <!--Splitting into two grids-->
        <div class="form-row">
            <div class="col-sm-6">
                <div class="form-group mr-4 ">
                    <label for="mobile">Mobile Number</label>
                    <input type="tel" class="form-control" name="mobile"
id="mobile" required
                    placeholder="Enter your Mobile No.">

                </div>
            </div>
            <div class="col-sm-6">
                <div class="form-group">
                    <label for="email">Email</label>
                    <input type="email" class="form-control" name="email"
id="email" required
                    placeholder="Enter your Email">

                </div>

            </div>
        </div>

        <div class="form-row">
            <div class="col-sm-4">
                <div class="form-group mr-4">
                    <label for="age">Age</label>
                    <input type="number" class="form-control" name="age"
id="age" required
                    placeholder="Enter your Age">

                </div>
            </div>

            <div class="col-sm-4">
                <div class="form-group mr-4">
                <label for="gender" class="form-label">Gender</label><br>
                <select id="gender" class="form-control" name="gender">
                    <option selected>Select your Gender</option>
                    <option>Male</option>
                    <option>Female</option>
                    <option>Other</option>
                </select>
            </div>
            </div>


            <div class="col-sm-4">
                <div class="form-group mr-4">
                <label for="blood-group" class="form-label">Blood
Group</label><br>
                <select id="blood-group" class="form-control" name="blood">
                    <option selected>Select your blood group</option>
                    <option>O+</option>
                    <option>O-</option>
                    <option>A+</option>
                    <option>A-</option>
                    <option>B+</option>
                    <option>B-</option>
                    <option>AB+</option>
                    <option>AB-</option>
                </select>
            </div>
            </div>
        </div>


            <div class="form-row">
                <div class="col-sm-4">
                    <div class="form-group mr-4 ">
                        <label for="Area">Area</label>
                        <input type="text" class="form-control" name="area"
id="Area" required
                        placeholder="Enter your Area Name">

                    </div>
                </div>
                <div class="col-sm-4">
                    <div class="form-group">
                        <label for="city">City</label>
                        <input type="text" class="form-control" name="city"
id="city" required
                        placeholder="Enter your City Name">

                    </div>

                </div>
                <div class="col-sm-4">
                    <div class="form-group mr-4 ">
                        <label for="district">District</label>
                        <input type="text" class="form-control"
name="district" id="district" required
                        placeholder="Enter your District Name">

                    </div>
                </div>
            </div>



                        <!--button-->
                <div class="form-group text-center">
                    <input type="reset" value="Reset" class="btn btn-dark mr-
2">

                    <input type="submit" value="Submit" class="btn btn-
success">

                </div>
            </form>
        </div>
    </div>
</div>
</div>
</div>




{% endblock %}
```

Request.html:

```
{% extends 'base.html'%}

<!--title tag-->
{% block title %}
<title>Plasma-Request</title>
<style>
    body{
     background-image:url('static\\bg.jpg');
     }
     .bg-primary{
       background-color:red !important;
     }
     .card-body{
        border-radius: 25px;
        background-color: rgba(255, 255, 255, 0.141);
        backdrop-filter: blur(5px);
     }
 </style>
{% endblock %}

<!---Login Content-->
{% block content %}
<!---Registration form-->


<div class="container mt-5" id="request-form">
  <div class="row justify-content-center">
      <div class="col-sm-12 ">
          <div class="card">
            <div class="msg" style="color: green;">{{ msg }}</div>
              <div class="card-body">
                <h4 style="text-align: center;">Request for Plasma</h4>
                  <!----Form content---->
                  <form action="/request_page" method="post">
```

```
                    <div class="form-row">
                        <div class="col-sm-6">
                            <div class="form-group mr-4 ">
                                <label for="drmail">Enter Donor Mail</label>
                                <input type="email" class="form-control"
name="drmail" id="drmail" required
                                placeholder="Enter Donor mail from the table">

                            </div>
                        </div>
                        <div class="col-sm-6">
                            <div class="form-group">
                                <label for="hospitalname">Hospital
Name</label>
                                <input type="text" class="form-control"
name="hospitalname" id="hospitalname" required
                                placeholder="Enter Hospital Nmae">

                            </div>

                        </div>
                    </div>

                    <div class="form-row">
                        <div class="col-sm-4">
                            <div class="form-group mr-4 ">
                                <label for="fullname">FullName</label>
                                <input type="text" class="form-control"
name="recname" id="fullname" required
                                placeholder="Enter your FullName">

                            </div>
                        </div>
                        <div class="col-sm-4">
                            <div class="form-group">
                                <label for="mobile">Mobile Number</label>
                                <input type="tel" class="form-control"
name="recmobile" id="mobile" required
                                placeholder="Enter your Mobile Number">

                            </div>

                        </div>
                        <div class="col-sm-4">
                            <div class="form-group mr-4 ">
                                <label for="recmail">Your Mail</label>
                                <input type="email" class="form-control"
name="recmail" id="recmail" required
                                placeholder="Enter your Email">

                            </div>
                        </div>
                    </div>


                    <div class="form-row">
                        <div class="col-sm-4">
                            <div class="form-group mr-4">
                                <label for="age">Age</label>
                                <input type="number" class="form-control"
name="recage" id="age" required
                                placeholder="Enter your Age">

                            </div>
                        </div>

                        <div class="col-sm-4">
                            <div class="form-group mr-4">
                            <label for="gender" class="form-
label">Gender</label><br>
                            <select id="gender" class="form-control"
name="recgender">
                              <option selected>Select your Gender</option>
                              <option>Male</option>
                              <option>Female</option>
                              <option>Other</option>
                            </select>
                        </div>
                        </div>


                        <div class="col-sm-4">
                            <div class="form-group mr-4">
                            <label for="blood-group" class="form-label">Blood
Group</label><br>
                            <select id="blood-group" class="form-control"
name="recbloodgroup">
                              <option selected>Select your blood
group</option>
                              <option>O+</option>
                              <option>O-</option>
                              <option>A+</option>
                              <option>A-</option>
                              <option>B+</option>
                              <option>B-</option>
                              <option>AB+</option>
                              <option>AB-</option>
                            </select>
                        </div>
                        </div>
                      </div>

                        <div class="form-row">
                            <div class="col-sm-4">
                                <div class="form-group mr-4 ">
                                    <label for="Area">Area</label>
                                    <input type="text" class="form-control"
name="recarea" id="Area" required
                                    placeholder="Enter your Area Name">

                                </div>
                            </div>
                            <div class="col-sm-4">
                                <div class="form-group">
                                    <label for="city">City</label>
                                    <input type="text" class="form-control"
name="reccity" id="city" required
                                    placeholder="Enter your City Name">

                                </div>

                            </div>
                            <div class="col-sm-4">
                                <div class="form-group mr-4 ">
                                    <label for="district">District</label>
                                    <input type="text" class="form-control"
name="recdistrict" id="district" required
                                    placeholder="Enter your District Name">

                                </div>
                            </div>
                        </div>


                      <!--button-->
                    <div class="form-group text-center modal-footer">
                        <button type="reset" class="btn btn-secondary" data-
dismiss="modal">Reset</button>
                        <button type="submit" class="btn btn-
success">Request</button>
                    </div>
```

```
                  </form>
              </div>
          </div>
      </div>
  </div>
</div>



{% endblock %}
```

signin.html:

```
{% extends 'base.html'%}

<!--title tag-->
{% block title %}
<title>Plasma-Signin</title>
<style>
   body{
    background-image:url('static\\bg.jpg');
    }
    .bg-primary{
      background-color:red !important;
    }
    .card-body{
      border-radius: 25px;
      background-color: rgba(255, 255, 255, 0.141);
      backdrop-filter: blur(5px);
    }
</style>
{% endblock %}

<!---Login Content-->
{% block content %}
<!---Registration form-->


<div class="container mt-5" id="request-form">
  <div class="row justify-content-center">
      <div class="col-sm-6 ">
          <div class="card">
              <div class="card-body">
                <h4 style="text-align: center;">Register as a user</h4>
                  <!----Form content---->
                  <form action="/signin" method="post">
                    <div class="form-group">
                      <label for="your-name">User Name</label>
                      <input type="text" class="form-control" name="username"
id="your-name" required
                      placeholder="Enter your UserName">
                  </div>

                      <div class="form-group">
                          <label for="email">Your Email</label>
                          <input type="email" class="form-control"
name="usermail" id="email" required
                          placeholder="Enter your Email">
                      </div>
                      <div class="form-group">
                          <label for="phone">Phone</label>
                          <input type="tel" class="form-control"
name="usercontact" id="phone" placeholder="Enter your mobile number"
                          required>
                      </div>
                      <div class="form-group">
                        <label for="password">Your Password</label>
                        <input type="password" class="form-control"
name="password" id="password" placeholder="Create a Password"
                        required>
                    </div>


                      <!--button-->
                      <div class="form-group text-center modal-footer">
                        <input type="reset" value="Reset" class="btn btn-dark
mr-2">

                        <input type="submit" value="Register" class="btn btn-
success">

                      </div>



                        <div>
                            <p style="text-align: center;">Already a user? <a
href="{{ url_for('login') }}">LogIn</a></p>
                        </div>

                  </form>
              </div>
          </div>
      </div>
  </div>
</div>



{% endblock %}
```

success.html:

```
{% extends 'base.html'%}

<!--title tag-->
{% block title %}
<title>Plasma-Success</title>
<style>
  body{
   background-image:url('static\\bg.jpg');
   }
   .bg-primary{
     background-color:red !important;
   }
   .card-body{
    border-radius: 25px;
    background-color: rgba(255, 255, 255, 0.141);
    backdrop-filter: blur(5px);
   }
</style>
{% endblock %}

{% block link %}
{% endblock %}

<!---User Content-->
{% block content %}



        <div class="container-fluid">
            <div class="text-center">

            <style>

            .hide {
              display: none;
            }
            .myDIV:hover + .hide {
              display: block;
              color: red;
            text-align: top;
            }


            </style>


            <div class="container">
            <div class="row p-1">
              <div class="col-sm-12">
                <div class="card" style="margin-top: 70px;">
                  <div class="msg">{{ msg }}</div>
                  <div class="card-body">


            <div class="myDIV">

                    <h2>Thank you for donating plasma.</h2>
                      </div>
                            </div>
                              </div>
                                  </div>
                                        </div></div> </div>


{% endblock%}
```

user_profile.html:

```
{% extends 'base.html'%}

<!--title tag-->
{% block title %}
<title>Plasma-Profile</title>
<style>
    .profile-area{
    padding:70px 0;
    border: 2px solid rgba(233, 226, 226,0.4);
    margin-right: 190px;
    margin-left: 190px;
    margin-top: 50px;
    }
.card{
    box-shadow: 0 0 30px rgba(0,0,0,0.1);
    overflow:hidden;
    border-radius:15px;
    margin-top:30px;
    background-image: linear-gradient(#ff0000,#ce3a3a);
    height: 200px;

    }
    .card:hover{
        border-color:red;
        transform: rotate(1deg);
        transition:.7s
    }
    .main-text, .card-body{
        text-align: center;
        padding-top: 69px;
    }
    .btn{
        padding: 10px 10px;
        margin-bottom: 20px;
    }
   body{
    background-image:url('static\\bg.jpg');
    }
    .bg-primary{
      background-color:red !important;
    }
    .card-body{
        border-radius: 25px;
        background-color: rgba(255, 255, 255, 0.141);
        backdrop-filter: blur(5px);
    }

    .profile-area{
    padding:70px 0;
    border: 0 ;
    margin-right: 190px;
    margin-left: 190px;
    margin-top: 50px;
    }

</style>
{% endblock %}

{% block link %}

{% endblock %}
<h2 class="page-header  text-center" style="margin-top: 30px;
color:black">Your Profile</h2>


<!---User Content-->
{% block content %}
<div class="container">
<h3 class="text-danger" style="margin-top: 50px;">Welcome :
{{session["username"]}}!!</h3>
</div>
<div class = "profile-area">
<div class = "container">
```

```
    <div class="msg">{{ msg }}</div>
    <div class="row">

    <div class = "col-12 col-md-6 col-lg-6">
        <div class = "card">
        <div class = "main-text card-body">
            <div><a href="{{ url_for('register') }}" class="btn btn-
light">Donate Plasma</a></div>
        </div>
        </div>
    </div>
    <div class = "col-12 col-md-6 col-lg-6">
        <div class = "card">
        <div class = "main-text card-body">
            <div><a href="{{ url_for('donorlist') }}" class="btn btn-
light">Request Plasma</a></div>
        </div>
        </div>
    </div>
    </div>
    </div>
    </div><br>
    <div class="container">
    <a href="{{url_for('logout')}}" class="btn btn-danger">Log Out</a></div>



{% endblock%}
```

home.css:

```
@media only screen and (max-width: 500px) {
    /* For mobile phones: */
    .landing, .landing-text h1,.landing-text h3, .landing-text .btn,img ,
.landing-text .btn a{
      width: 100%;
      height: auto;
    }
  }
 .landing{
    margin-top: 100px;
    margin-left: 50px;
    margin-right: 50px;
 }

 .landing-text h1{
    font-size: 65px;
}
```

```css
.landing-text h3{
    margin: 6px;
    font-size: 15px;
    line-height: 1.8;
    color: #777777;
    margin-right: 20px;
}

.landing-text .btn{
    width: 200px;
    margin-top: 30px;
    padding: 14px 20px 12px 20px;
    background-color: red;
    border-radius: 45px;
    text-align: center;

}

.landing-text .btn a{
    font-size: 18px;
    color: #fff;
}

  img {
    float: right;
  }
```

style.css:

```css
*{
    font-family: 'Montserrat', sans-serif;
}


#icon{
    color: red;
    padding-right: 2px;
}
```

app.py:

```
from flask import Flask,render_template,request,url_for,flash,redirect,session
import ibm_db
import sendgrid
import re
from sendgrid.helpers.mail import *


app = Flask(__name__)
app.secret_key="1"

conn = ibm_db.connect("DATABASE=BLUDB;HOSTNAME=b1bc1829-6f45-4cd4-bef4-10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32304;PROTOCOL=TCPIP;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=hcz07812;PWD=9ETLzrZz4t34xNVw;","","")

@app.route("/")
def index():
    return render_template('home.html')

@app.route("/home")
def home_page():
    return render_template('home.html')
#-------------------------------------------------------

@app.route("/login",methods = ['POST', 'GET'])
def login():
    global userid
    msg = ''
    if request.method == 'POST' :
        username = request.form['username']
        password = request.form['password']
        sql = "SELECT * FROM LOGIN WHERE username =? AND password=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.bind_param(stmt,2,password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print (account)
        if account:
            session['loggedin'] = True
            session['id'] = account['USERNAME']
            userid=  account['USERNAME']
            session['username'] = account['USERNAME']
            msg = 'Logged in successfully !'

            return render_template('user_profile.html', msg = msg)
        else:
```

```python
            msg = 'Incorrect username / password !'
    return render_template('login.html', msg = msg)

#---------------------------------------------------------
# After login
@app.route('/afterlogin')
def afterlogin():
    return render_template("user_profile.html")

#-------------------------------------------------------

@app.route("/signin",methods = ['POST', 'GET'])
def signin():
    msg = ''
    if request.method == 'POST' :
        username = request.form['username']
        usermail = request.form['usermail']
        usercontact = request.form['usercontact']
        password = request.form['password']
        sql = "SELECT * FROM LOGIN WHERE username =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            msg = 'Account already exists !'
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', usermail):
            msg = 'Invalid email address !'
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg = 'name must contain only characters and numbers !'
        else:
            mailtest_registration(usermail)
            insert_sql = "INSERT INTO LOGIN VALUES (?, ?, ?, ?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, username)
            ibm_db.bind_param(prep_stmt, 2, usermail)
            ibm_db.bind_param(prep_stmt, 3, usercontact)
            ibm_db.bind_param(prep_stmt, 4, password)
            ibm_db.execute(prep_stmt)
            msg = 'You have successfully registered !'
            mailtest_registration(usermail)
            return render_template('login.html', msg = msg)
    elif request.method == 'POST':
        msg = 'Please fill out the form !'

    return render_template('signin.html', msg = msg)
#-------------------------------------------------------------------
```

```
# sendgrid integration
def mailtest_registration(to_email):
    sg = sendgrid.SendGridAPIClient(api_key= 'SG.9_tPZuieRP-
tHkezgkD_ZA.qpw1oJcv4Ig6fT-Vz4mIMVbdnJ5HPPfcvlDyacxC-iE' )
    from_email = Email("rakeshprasanna72@gmail.com")
    subject = "Registration Successfull!"
    content = Content("text/plain", "You have successfully registered as user.
Please Login using your Username and Password to donate/request for Plasma.")
    mail = Mail(from_email, to_email, subject, content)
    response = sg.client.mail.send.post(request_body=mail.get())
    print(response.status_code)
    print(response.body)
    print(response.headers)

#for donor
def mailtest_donor(to_email):
    sg = sendgrid.SendGridAPIClient(api_key= 'SG.9_tPZuieRP-
tHkezgkD_ZA.qpw1oJcv4Ig6fT-Vz4mIMVbdnJ5HPPfcvlDyacxC-iE' )
    from_email = Email("rakeshprasanna72@gmail.com")
    subject = "Thankyou for Registering as Donor!"
    content = Content("text/plain", "Every donor is an asset to the nation who
saves people's lives, and you're one of them.We appreciate your efforts. Thank
you!!")
    mail = Mail(from_email, to_email, subject, content)
    response = sg.client.mail.send.post(request_body=mail.get())
    print(response.status_code)
    print(response.body)
    print(response.headers)

#for request

def mailtest_request(to_email):
    sg = sendgrid.SendGridAPIClient(api_key= 'SG.9_tPZuieRP-
tHkezgkD_ZA.qpw1oJcv4Ig6fT-Vz4mIMVbdnJ5HPPfcvlDyacxC-iE' )
    from_email = Email("rakeshprasanna72@gmail.com")
    subject = "Request Submitted!"
    content = Content("text/plain", "Your request has been successfully
submitted. Please be patient, your requested donor will get back to you
soon.")
    mail = Mail(from_email, to_email, subject, content)
    response = sg.client.mail.send.post(request_body=mail.get())
    print(response.status_code)
    print(response.body)
    print(response.headers)

#for request sending to donor

def mailtest_requesttodonor(to_email):
    sg = sendgrid.SendGridAPIClient(api_key= 'SG.9_tPZuieRP-
tHkezgkD_ZA.qpw1oJcv4Ig6fT-Vz4mIMVbdnJ5HPPfcvlDyacxC-iE' )
    from_email = Email("rakeshprasanna72@gmail.com")
    subject = "Requesting Plasma"
    content = Content("text/plain", "Your registration has been requested by a
recipient, we will share futher details in future. Stay connected!!")
    mail = Mail(from_email, to_email, subject, content)
    response = sg.client.mail.send.post(request_body=mail.get())
    print(response.status_code)
    print(response.body)
    print(response.headers)
#------------------------------------------------------------------------------
--
@app.route("/register")
def register():
    return render_template('register.html')

@app.route("/adddonor",methods = ['POST','GET'])
def adddonor():

    if request.method == 'POST':
        name = request.form['name']
        mobile = request.form['mobile']
        email = request.form['email']
        age = request.form['age']
        gender = request.form['gender']
        blood = request.form['blood']
        area = request.form['area']
        city = request.form['city']
        district = request.form['district']

        sql = "SELECT * FROM DONOR2 WHERE name =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,name)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)

        if account:
            return render_template('donor.html', msg="You are already a
member, please login using your details")
        else:
            mailtest_donor(email)
            insert_sql = "INSERT INTO DONOR2 VALUES (?,?,?,?,?,?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, name)
            ibm_db.bind_param(prep_stmt, 2, mobile)
            ibm_db.bind_param(prep_stmt, 3, email)
            ibm_db.bind_param(prep_stmt, 4, age)
            ibm_db.bind_param(prep_stmt, 5, gender)
            ibm_db.bind_param(prep_stmt, 6, blood)
            ibm_db.bind_param(prep_stmt, 7, area)
            ibm_db.bind_param(prep_stmt, 8, city)
            ibm_db.bind_param(prep_stmt, 9, district)
            ibm_db.execute(prep_stmt)
        return render_template('success.html', msg="Registered successfuly..")
#-----------------------------------------------------------------------------
------------

@app.route('/donorlist')
def donorlist():
    donor2 = []
    sql = "SELECT * FROM DONOR2"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        donor2.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)
    if donor2:
        return render_template("donor.html", donor2 = donor2)

#----------------------------------------------------------------------------
@app.route("/request_page", methods = ['GET','POST'])
def request_page():
    msg = ''
    if request.method == 'POST' :
        drmail = request.form['drmail']
        hospitalname = request.form['hospitalname']
        recname = request.form['recname']
        recmobile = request.form['recmobile']
        recmail = request.form['recmail']
        recage = request.form['recage']
        recgender = request.form['recgender']
        recbloodgroup = request.form['recbloodgroup']
        recarea = request.form['recarea']
        reccity = request.form['reccity']
        recdistrict = request.form['recdistrict']
        sql = "SELECT * FROM REQUEST2 WHERE recname =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,recname)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            msg = 'Request already exists !'
        else:
            mailtest_request(recmail)
            mailtest_requesttodonor(drmail)
            insert_sql = "INSERT INTO REQUEST2 VALUES (?, ?, ?, ?, ?, ?, ?, ?, 
?, ?, ?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, drmail)
            ibm_db.bind_param(prep_stmt, 2, hospitalname)
            ibm_db.bind_param(prep_stmt, 3, recname)
            ibm_db.bind_param(prep_stmt, 4, recmobile)
            ibm_db.bind_param(prep_stmt, 5, recmail)
            ibm_db.bind_param(prep_stmt, 6, recage)
            ibm_db.bind_param(prep_stmt, 7, recgender)
            ibm_db.bind_param(prep_stmt, 8, recbloodgroup)
            ibm_db.bind_param(prep_stmt, 9, recarea)
            ibm_db.bind_param(prep_stmt, 10, reccity)
            ibm_db.bind_param(prep_stmt, 11, recdistrict)
            ibm_db.execute(prep_stmt)
            msg = 'Your request has been submitted!'
            return render_template('request.html', msg = msg)
    elif request.method == 'POST':
        msg = 'Please fill out the form !'

    return render_template('request.html', msg = msg)


#-----------------------------------------
@app.route('/logout')
def logout():
    session.clear()
    return redirect(url_for("index"))

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```

Docker file:

```
FROM python:3.9.5
WORKDIR /app
COPY requirements.txt ./
RUN pip install -r requirements.txt
COPY . .
EXPOSE 5000
CMD ["python","./app.py"]
```

requirements.txt:

```
flask
ibm_db
sendgrid
```

Kubernetes:

dashboard-adminuser.yaml:

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: admin-user
  namespace: kubernetes-dashboard
---
apiVersion: v1
kind: Secret
metadata:
  name: admin-user-token
  namespace: kubernetes-dashboard
  annotations:
    kubernetes.io/service-account.name: admin-user
type: kubernetes.io/service-account-token

---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-user
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: admin-user
  namespace: kubernetes-dashboard
```

flask_deployment.yaml:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: plasma-donor-application-deployment

spec:
  replicas: 1
  selector:
    matchLabels:
      app: plasma-donor-application
  template:
    metadata:
      labels:
        app: plasma-donor-application

    spec:
      containers:
        - name: plasma-donor-application
          image: us.icr.io/plasma_donor_ns/plasma
          imagePullPolicy: Always
          ports:
            - containerPort: 5000
              protocol: TCP
```

flask_ingress.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: plasma-donor-application-deployment

spec:
  replicas: 1
  selector:
    matchLabels:
      app: plasma-donor-application
  template:
    metadata:
      labels:
        app: plasma-donor-application

    spec:
      containers:
        - name: plasma-donor-application
          image: us.icr.io/plasma_donor_ns/plasma
          imagePullPolicy: Always
          ports:
            - containerPort: 5000
              protocol: TCP
```

flask_service.yaml:

```
apiVersion: v1
kind: Service
metadata:
  name: plasma-donor-application-deployment
spec:
  type: ClusterIP
  ports:
    - port: 5000
  selector:
    app: plasma-donor-application
```

ibm_deployment.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flask-app

spec:
  replicas: 3
  selector:
    matchLabels:
      app: flask-app
  template:
    metadata:
      labels:
        app: flask-app

    spec:
      containers:
        - name: repo2
          image: icr.io/plasmadonorapp1/repo2
          imagePullPolicy: Always
          ports:
            - containerPort: 5000
              protocol: TCP
```

GitHub and Project Demo Link:

Project Demo Video Link:

https://drive.google.com/file/d/1xiv1SDeQsR8rPPCvMF0r4GibGI-pYIxY/view?usp=sharing

GitHub Link:

https://github.com/IBM-EPBL/IBM-Project-32140-1660208230