

BLOOD AND PLASMA BANK

Submitted in partial fulfilment of requirements for the
award of degree of

B. TECH

COMPUTER SCIENCE AND ENGINEERING

SUBMITTED BY

1. ARCHIT JAIN (00355302717)
2. NISCHAY (01155302717)

Under the Guidance
Of

Mr. PRADEEP TYAGI



Department of Computer Science and Engineering

B.M. Institute of Engineering and Technology
Sector-10, Sonipat
(Affiliated to GGSIP University, Delhi)

Certificate

It is to certify that the project has been carried out by the students of 7th /8th semester **ARCHIT Jain (01055302717)** and **NISCHAY (01155302717)** under my guidance. The report covers all the aspects of the work done (including H/W & S/W, Coding etc.).

The project report is complete in all respects and I have understood the entire software.

Mr. PRADEEP TYAGI

Incharge Name

Certificate

It is to certify that the project has been carried out by the student of 7th semester “BLOOD AND PLASMA BANK” with the help of other team members Archit Jain, Nischay under the guidance of Computer Science and Engineering Department. The report covers all the aspects of the work done (including H/W & S/W, Coding etc.)

Mr.Pradeep Tyagi
H.O.D. COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**(AFFILIATED TO GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY, DELHI) DELHI –
110089**

CANDIDATE'S DECLARATION

It is hereby certified that the work which is being presented in the B. Tech Minor Project Report entitled "**BLOOD AND PLASMA BANK**" in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** and submitted in the **Department of Computer science and engineering , B.M.I.E.T (Affiliated to Guru Gobind Singh Indraprastha University, Delhi)** is an authentic record of our own work carried out during a period from **AUGUST,2020 to DECEMBER,2020** under the guidance of **PRADEEP TYAGI with H.O.D CSE DEPARTMENT**

The matter presented in the B. Tech Minor Project Report has not been submitted by me for the award of any other degree of this or any other Institute.

Archit Jain	Nischay Rajpal
0035530271	01155302717

This is to certify that the above statement made by the candidate is correct to the best of my knowledge. They are permitted to appear in the External Major Project Examination

Mr. Pradeep Tyagi

The B. Tech Minor Project Viva-Voce Examination of **Nikhil Jain (01055302717)** and **Nischay Rajpal(01155302717)** has been held on

Mr.Pradeep Tyagi

(Signature of External Examiner)

ACKNOWLEDGEMENT

We express our deep gratitude to **Mr. Pradeep Tyagi** Department of computer science and Engineering for his valuable guidance and suggestion throughout my project work. We are thankful for his valuable guidance.

We would like to extend my sincere thanks to HOD, for his time to time suggestions to complete my project work. I am also thankful to **Dr.HARISH MITTAL** for providing me the facilities to carry out my project work.

Archit Jain

00355302717

Nichay Rajpal

01155302717

ABSTRACT

Blood and Plasma donor application as the name suggests its an application that brings out blood and plasma donors in the nearby areas. Plasma donation would be added as updated version as plasma donation is going to be need of the hour. One can donate receive the blood according to the need and can too contact nearby hospitals for the same

Listing of the nearby hospitals is also one of the crucial features of the application

Technology used us android development and google API's for proper functioning of app. Moreover, the use of xml is to done for front end designing and backend will be by webhosting using php scripts with Mysql as query language. Logo designing software are also utilized.

Further more working with google map API is planned for mapping the hospitals and blood banks around.

TABLE OF CONTENTS

Certificate	i
Certificate	ii
Declaration	iii
Acknowledgement	iv
Abstract	v
Table of Contents	
Chapter– 1 Introduction	
1.1 General	1
1.2 Overview of the project	2
1.3 Problem Statement	3
1.4 Scope of Study	5
Chapter – 2 System Analysis	
2.1 General	7
2.2 SRS	17
2.3 Feasibility Study	22
Chapter – 3 System Design	
3.1 Design Methods	26
3.2 Various development approaches	27
Chapter –4 Testing	
4.1 Testing techniques and strategies	29
4.2 Debugging & Improvement	30
4.3 Finished Code	32
4.4 Outputs	36
Chapter – 5 Implementation	
5.1 System Implementation	43
5.2 Software Implementation	45
5.2.1 Current Uses	47
5.2.2 Ongoing research and improvement	47
Conclusion & Future Uses	48
References	49

Chapter 1

Introduction

1.1 General

Blood and plasma bank manager is the android native application to facilitate the blood and plasma donors and receiver. Android is one of the fastest growing technology among the youth thus is being a powerful tool for creating a social web and thus connecting more people with the application. Thus, the platform for the native application is android so that application can be socially benefited by more users.

Signing in to the application a user is madidate to provide our database with all his information so that he could be potential donor in respect to his age. Thus, filtering the user database using user's locality, area, address and mobile number so that contacting user could be easy to help the public around.

UI/UX (User interface and user experience) is kept simple so that the application is handy for user by all the age groups, with same ease

Design of the application is purely based upon Xml designing

Whereas advanced designing methods such as selecting colour pallet and material designing are also used for improving user interface. Red and Yellow colour pattens are used to firmly differentiate between blood and plasma.

Logo designing methods and software are also used to design the logo of the application.

1.2 Overview of the Project

The project here aims to achieve the objective of creating a Blood and plasma bank manager is to facilitate the blood and plasma donors and receiver. Android is one of the fastest growing technology among the youth thus is being a powerful tool for creating a social web and thus connecting more people with the application. Thus, the platform for the native application is android so that application can be socially benefited by more users.

Signing in to the application a user is mandated to provide our database with all his information so that he could be potential donor in respect to his age. Thus, filtering the user database using user's locality, area, address and mobile number so that contacting user could be easy to help the public around.

UI/UX (User interface and user experience) is kept simple so that the application is handy for user by all the age groups, with same ease

Design of the application is purely based upon Xml designing

Whereas advanced designing methods such as selecting colour pallet and material designing are also used for improving user interface. Red and Yellow colour patterns are used to firmly differentiate between blood and plasma.

Logo designing methods and software are also used to design the logo of the application. The php scripts are used to communicate between server and application by creating an API that used json objects for transferring the user data from database to application.

Free 000Webhosting is used to create admin panel that is to monitor database and filtering around the user data, to divide them in listed blood groups and areas.

1.3 Problem Statement

A problem statement is a concise description of an issue to be addressed or a condition to be improved upon. It identifies the gap between the current (problem) state and desired (goal) state of a process or product. Focusing on the facts, the problem statement should be designed to address the Five W's. The first condition of solving a problem is understanding the problem, which can be done by way of a problem statement.

Problem statements are widely used by businesses and organizations to execute process improvement projects. A simple and well-defined problem statement will be used by the project team to understand the problem and work toward developing a solution. It will also provide management with specific insights into the problem so that they can make appropriate project-approving decisions. As such, it is crucial for the problem statement to be clear and unambiguous.

The main purpose of the problem statement is to identify and explain the problem. This includes describing the existing environment, where the problem occurs, and what impacts it has on users, finances, and ancillary activities. Additionally, the problem statement is used to explain what the expected environment looks like. Defining the desired condition provides an overall vision for the process or product. It makes clear the purpose for initiating the improvement project and the goals that it is meant to accomplish.

Another important function of the problem statement is to be used as a communication device. Problem statement helps with obtaining buy-in from those involved in the project, before the project begins, the stakeholders verify the problem and goals are accurately described in the problem statement. Once this approval is received, the project team reviews it to ensure everyone understands the issue at hand and what they are trying to accomplish. This also helps define the project scope, which keeps the project concentrated on the overall goal.

The problem statement is referenced throughout the project to establish focus within the project team and verify they stay on track. At the end of the project, it is revisited to confirm the implemented solution indeed solves the problem. A well-defined problem statement can also aid in performing root-cause analysis to understand why the problem occurred and ensure measures can be taken to prevent it from happening in the future.

It is important to note that the problem statement does not define the solution or methods of reaching the solution. The problem statement simply recognizes the gap between the problem and goal states. It can be said that, “a problem well stated is half solved.” However, there are often multiple, viable solutions to a problem. Only after the problem statement is written and agreed upon should the solution(s) be discussed, and the resulting course of action determined.

The problem statement is the initial starting point for a project. It is basically a one to three-page statement that everyone on the project agrees with that describes what will be done at a high level. The problem statement is intended for a broad audience and should be written in non-technical terms. It helps the non- technical and technical personnel communicate by providing a description of a problem. It doesn’t describe the solution to the problem.

The input to requirement engineering is the problem statement prepared by customer. It may give an overview of the existing system along with broad expectations from the new system.

The first phase of requirements engineering begins with requirements elicitation i.e., gathering of information about requirements. Here, requirements are identified with the help of customer and existing system processes. So, from here begins the preparation of problem statement.

So, basically a problem statement describes what needs to be done without describing how.

This application will display potential donors in a designated locality with filtered blood groups.

1.4 Scope of Study

Blood and Plasma donations are part and parcel of life. Easily facilitating blood and plasma in a given area could be difficult if social network and traffic on the application would be less. Thus, scope of study would also be directed towards the social marketing of the application. Checking and validating credentials would also be done using the xml forms , only the valid mobile number and blood group would be allowed to login in to the user dashboard

Finding donors: -

This would be the main task of the application to find donors with the required blood group in the area, User credentials would be filtered at the back end and with the help of php scripts at the backend, user with the matching blood group would be listed as card, in the card view in user dash board. Along with the blood group and address, user phone number would also be popped and the function of calling that user is also provided

Requesting blood:-

The blood/ plasma requested would be displayed to the user as a card on card view in the user dashboard , the requested message would contain user calling number, address and a short note/message.

Home Page: -

Home page of the application displays all the requests of blood in that particular area that is being short listed using php back end scripts

Php scripts (Sample of filtering data)

```
<?php
require "init.php";

$city= $_POST["city"];
$blood_group= $_POST["blood_group"];

$sql = "SELECT name,number,city from user_table WHERE blood_group LIKE
'$blood_group' AND city LIKE '%$city%' ";

$result = mysqli_query($con,$sql);
$response = array();

while($row = mysqli_fetch_assoc($result))
{
    array_push($response,$row);
}
echo json_encode($response);

?>
```

Chapter 2

System Analysis

2.1 General

Systems development is systematic process which includes phases such as planning, analysis, design, deployment, and maintenance. Here, in this tutorial, we will primarily focus on –

- Systems analysis
- Systems design

1. Systems Analysis

It is a process of collecting and interpreting facts, identifying the problems, and decomposition of a system into its components.

System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives. It is a problem-solving technique that improves the system and ensures that all the components of the system work efficiently to accomplish their purpose.

Analysis specifies what the system should do.

2. Systems Design

It is a process of planning a new business system or replacing an existing system by defining its components or modules to satisfy the specific requirements. Before planning, you need to understand the old system thoroughly and determine how computers can best be used in order to operate efficiently.

System Design focuses on how to accomplish the objective of the system.

System Analysis and Design (SAD) mainly focuses on –

- Systems
- Processes
- Technology

3. *What is a System?*

The word System is derived from Greek word Systema, which means an organized relationship between any set of components to achieve some common cause or objective.

A system is “an orderly grouping of interdependent components linked together according to a plan to achieve a specific goal.”

Constraints of a System

A system must have three basic constraints –

- A system must have some structure and behaviour which is designed to achieve a predefined objective.
- Interconnectivity and interdependence must exist among the system components.
- The objectives of the organization have a higher priority than the objectives of its subsystems.

For example, traffic management system, payroll system, automatic library system, human resources information system.

4. *Properties of a System*

A system has the following properties –

1. Organization

Organization implies structure and order. It is the arrangement of components that helps to achieve predetermined objectives.

2. Interaction

It is defined by the manner in which the components operate with each other.

For example, in an organization, purchasing department must interact with production department and payroll with personnel department.

3. Interdependence

Interdependence means how the components of a system depend on one another. For proper functioning, the components are coordinated and linked together according to a specified plan. The output of one subsystem is the required by other subsystem as input.

4. Integration

Integration is concerned with how a system component are connected together. It means that the parts of the system work together within the system even if each part performs a unique function.

5. Central Objective

The objective of system must be central. It may be real or stated. It is not uncommon for an organization to state an objective and operate to achieve another.

The users must know the main objective of a computer application early in the analysis for a successful design and conversion

6. Outputs and Inputs

- The main aim of a system is to produce an output which is useful for its user.
- Inputs are the information that enters into the system for processing.
- Output is the outcome of processing.

7. Processor(s)

- The processor is the element of a system that involves the actual transformation of input into output.
- It is the operational component of a system. Processors may modify the input either totally or partially, depending on the output specification.
- As the output specifications change, so does the processing. In some cases, input is also modified to enable the processor for handling the transformation.

8. Control

- The control element guides the system.
- It is the decision-making subsystem that controls the pattern of activities governing input, processing, and output.
- The behaviour of a computer System is controlled by the Operating System and software. In order to keep system in balance, what and how much input is needed is determined by Output Specifications.

9. Feedback

- Feedback provides the control in a dynamic system.

- Positive feedback is routine in nature that encourages the performance of the system.
- Negative feedback is informational in nature that provides the controller with information for action.

10. Environment

- The environment is the “supersystem” within which an organization operates.
- It is the source of external elements that strike on the system.
- It determines how a system must function. For example, vendors and competitors of organization’s environment, may provide constraints that affect the actual performance of the business.

11. Boundaries and Interface

- A system should be defined by its boundaries. Boundaries are the limits that identify its components, processes, and interrelationship when it interfaces with another system.
- Each system has boundaries that determine its sphere of influence and control.
- The knowledge of the boundaries of a given system is crucial in determining the nature of its interface with other systems for successful design.

5. *Types of Systems*

The systems can be divided into the following types –

1. Physical or Abstract Systems

- Physical systems are tangible entities. We can touch and feel them.

- Physical System may be static or dynamic in nature. For example, desks and chairs are the physical parts of computer centre which are static. A programmed computer is a dynamic system in which programs, data, and applications can change according to the user's needs.
- Abstract systems are non-physical entities or conceptual that may be formulas, representation or model of a real system.

Open or Closed Systems

- For example, an information system which must adapt to the changing environmental conditions.
- A closed system does not interact with its environment. It is isolated from environmental influences. A completely closed system is rare in reality.

2. Adaptive and Non-Adaptive System

- Adaptive System responds to the change in the environment in a way to improve their performance and to survive. For example, human beings, animals.
- Non-Adaptive System is the system which does not respond to the environment. For example, machines.

3. Permanent or Temporary System

- Permanent System persists for long time. For example, business policies.
- Temporary System is made for specified time and after that they are demolished. For example, A DJ system is set up for a program and it is dissembled after the program.

4. Natural and Manufactured System

- Natural systems are created by the nature. For example, Solar system, seasonal system.
- Manufactured System is the man-made system. For example, Rockets, dams, trains.

5. Deterministic or Probabilistic System

- Deterministic system operates in a predictable manner and the interaction between system components is known with certainty. For example, two molecules of hydrogen and one molecule of oxygen makes water.
- Probabilistic System shows uncertain behaviour. The exact output is not known. For example, Weather forecasting, mail delivery.

6. Social, Human-Machine, Machine System

- Social System is made up of people. For example, social clubs, societies.
- In Human-Machine System, both human and machines are involved to perform a particular task. For example, Computer programming.
- All the tasks are performed by the machine. For example, an autonomous robot.

7. Man-Made Information Systems

- It is an interconnected set of information resources to manage data for particular organization, under Direct Management Control (DMC).
- This system includes hardware, software, communication, data, and application for producing information according to the need of an organization.

Man-made information systems are divided into three types –

- Formal Information System – It is based on the flow of information in the form of memos, instructions, etc., from top level to lower levels of management.
- Informal Information System – This is employee-based system which solves the day to day work related problems.
- Computer Based System – This system is directly dependent on the computer for managing business applications. For example, automatic library system, railway reservation system, banking system, etc.

6. *Systems Models*

1. Schematic Models

- A schematic model is a 2-D chart that shows system elements and their linkages.
- Different arrows are used to show information flow, material flow, and information feedback.

2. Flow System Models

- A flow system model shows the orderly flow of the material, energy, and information that hold the system together.
- Program Evaluation and Review Technique (PERT), for example, is used to abstract a real-world system in model form.

3. Static System Models

- They represent one pair of relationships such as *activity–time* or *cost–quantity*.
- The Gantt chart, for example, gives a static picture of an activity-time relationship.

4. Dynamic System Models

- Business organizations are dynamic systems. A dynamic model approximates the type of organization or application that analysts deal with.
- It shows an ongoing, constantly changing status of the system. It consists of –
 - Inputs that enter the system
 - The processor through which transformation takes place
 - The program(s) required for processing

7. Categories of Information

There are three categories of information related to managerial levels and the decision managers make.

Volume of Information	Type of Information	Information Level	Management Level	System Support
Low Consensed	Unstructured	Strategic Information	Upper	DSS
Medium Moderately Processed	Moderately Structured	Management Control Information	Middle	MIS
Large Detail Reports	Highly Structured	Operational Information	Lower	DPS

1. Strategic Information

- This information is required by topmost management for long range planning policies for next few years. For example, trends in revenues, financial investment, and human resources, and population growth.
- This type of information is achieved with the aid of Decision Support System (DSS).

2. Managerial Information

- This type of Information is required by middle management for short and intermediate range planning which is in terms of months. For example, sales analysis, cash flow projection, and annual financial statements.
- It is achieved with the aid of Management Information Systems (MIS).

3. Operational information

- This type of information is required by low management for daily and short-term planning to enforce day-to-day operational activities. For example, keeping employee attendance records, overdue purchase orders, and current stocks available.
- It is achieved with the aid of Data Processing Systems (DPS).

2.2 SRS

A software requirements specification (SRS) is a description of a software system to be developed. It is modelled after business requirements specification (CONOPS), also known as a stakeholder requirements specification (SRS). The software requirements specification lays

out functional and non-functional requirements, and it may include a set of use cases that describe user interactions that the software must provide to the user for perfect interaction.

Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers on how the software product should function (in a market-driven project, these roles may be played by the marketing and development divisions). Software requirements specification is a rigorous assessment of requirements before the more specific system design stages, and its goal is to reduce later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules. Used appropriately, software requirements specifications can help prevent software project failure.¹

The software requirements specification document lists sufficient and necessary requirements for the project development. To derive the requirements, the developer needs to have clear and thorough understanding of the products under development. This is achieved through detailed and continuous communications with the project team and customer throughout the software development process.

The SRS may be one of a contract's deliverable data item descriptions or have other forms of organizationally mandated content.

Table of Contents for an SRS Document:

- 1. Introduction**
 - 1.1 Purpose
 - 1.2 Document Conventions
 - 1.3 Intended Audience and Reading Suggestions
 - 1.4 Project Scope
 - 1.5 References
- 2. Overall Description**

- 2.1 Product Perspective
- 2.2 Product Features
- 2.3 User Classes and Characteristics
- 2.4 Operating Environment
- 2.5 Design and Implementation Constraints
- 2.6 Assumptions and Dependencies

3. System Features

- 3.1 Functional Requirements

4. External Interface Requirements

- 4.1 User Interfaces
- 4.2 Hardware Interfaces
- 4.3 Software Interfaces
- 4.4 Communication Interfaces

5. Non-Functional Requirements

- 5.1 Security Requirements
- 5.2 Software Quality Attributes

SRS Document for the Application Software – BLOOD AND PLASMA BANK MANAGER:

1. Introduction

1.1 Purpose: -

The purpose of this application is to socialize more blood and plasma donors and receivers.

1.2 Document Convention: -

For containing the data, we have used the user login credentials and details of their blood, phone numbers and addresses are being socialized.

1.3 Intended Audience and Reading Suggestion: -

This project is a prototype for the Blood Bank. This has been implemented under the guidance of college teachers

Project scope: -

The purpose of the BLOOD AND PLASMA BANK MANAGER is to ease the donation of blood and plasma so that a life can be saved at curtail point of time.

The system is based on android application with web hosts as a server to store user data.

1.4 References

- <https://www.github.com>
- Android programming for beginners' book by John Horton
- <https://www.browserstack.com>
- <https://www.developer.android.com>
- <https://www.Vogella.com>
- <https://www.android.com>
- <https://www.Learn Php.com>
- <https://www.CodeCourse.com>
- <https://www.SidePoiny PHP.com>
- <https://www.000WebHost.com>
- <https://www.androidcentral.com>
- <https://www.stackoverflow.com>

2. Overall Description

2.1 Product Perspective

- A donation bank portal stores the following information.
- Details:

It includes the display all the users information in it.

2.2 Product Features

- The major features of Blood and plasma manager are that its interface is user friendly and good looking just like a premium application.
- User can request and donate blood and plasma from thousands of people at the same time.
- Backend is keep in simple format, .php files are used.
- API's are used for securing the user data.

2.3 USER CLASS And CHARACTERISTICS

We have made a class team in which we have created the data member and member functions.

2.4 Operating Environment

- Distributed Database
- Client/Server System
- Operating System: Android.
- Platform: MIX

2.5 DESIGN and IMPLEMENTATION CONSTRAINTS

- The global schema, fragmentation schema, and allocation schema.
- Implement the database at least using a centralized database management system.

3. *System Features*

3.1 *Functional Requirements*

Distributed Database: - It implies that a single application should be able to operate transparently on data that is spread across a variety of different databases.

4. External Interface Requirements

4.1 User Interfaces

Front-end software: XML, MATERIAL DESIGNING, PICS ART

Back-end software: PHP, Webhost, MySQL, Java,.

4.2 Hardware Interfaces

Active Internet	:	Required
RAM	:	2Gb and more
Hard Disk	:	Requires space in megabytes,
Screen Size	:	Best exp in 5 inch and above
Keyboard	:	Mobile Keyboard

4.3 Software interface

Operating system	:	Android 7 or above
------------------	---	--------------------

4.4 Communication Interfaces

The software has online support and can only be used if there is an access to internet.

5. Non-Functional Requirements

5.1 Security Requirements

Security systems need database storage just like many other applications. However, the special requirements of the security market mean that vendors must choose their database partner carefully.

5.2 Software Quality Attributes

- Filtering donors on the basis location and blood group.
- Correctness: The user will search for the donors in a particular area with specific blood group
- Maintainability: The details of guests or customer will store in well-organized manner.
- Usability: The users can use this for donating and requesting blood and plasma.

SRS END

2.3 Feasibility Study

Economic Feasibility – Identifying & Forecasting Costs & Benefits

Costs and benefits of the proposed computer system must always be considered together, because they are interrelated and often interdependent. Although the systems analyst is trying to propose a system that fulfil various information requirements, decisions to continue with the proposed system will be based on a cost-benefit analysis, not on information requirements. In many ways, benefits are measured becomes apparent in the next section.

Forecasting

Systems analysts are required to predict certain key variables before the proposal is submitted to the client. To some degree, a systems analyst will rely on a what-if analysis, such as, “What if labour costs rise only 5 percent per year for the next three years, rather than 10 percent?” The systems analyst should realize, however, that he or she cannot rely on what-if analysis for everything if the proposal is to be credible, meaningful, and valuable.

The systems analyst has many forecasting models available. The main condition for choosing a model is the availability of historical data. If they are unavailable, the analyst must turn to one of the judgment methods: estimates from the sales force, surveys to estimate customer demand, Delphi studies (a consensus forecast developed independently by a group of experts through a series of iterations), creating scenarios, or drawing historical analogies.

If historical data are available, the next differentiation between classes of techniques involves whether the forecast is conditional or unconditional. Conditional implies that there is an association among variables in the model or that such a causal relationship exists. Common methods in this group include correlation, regression, leading indicators, econometrics, and input/output models.

Unconditional forecasting means the analyst isn't required to find or identify any causal relationships. Consequently, systems analysts find that these methods are low-cost, easy-to-implement alternatives. Included in this group are graphical judgment, moving averages, and analysis of time-series data. Because these methods are simple, reliable, and cost effective, the remainder of the section focuses on them.

4. Estimation of Trends

Trends can be estimated in a number of different ways. One way to estimate trends is to use a moving average. This method is useful because some seasonal, cyclical, or random patterns may be smoothed, leaving the trend pattern. The principle behind moving averages is to calculate the arithmetic mean of data from a fixed number of periods; a three-month moving average is simply the average of the last three months. For example, the average sales for January, February, and March are used to predict the sales for April.

Then the average sales for February, March, and April are used to predict the sales for May, and so on.

When the results are graphed, it is easily noticeable that the widely fluctuating data are smoothed. The moving average method is useful for its smoothing ability, but at the same time

it has many disadvantages. Moving averages are more strongly affected by extreme values than by using graphical judgment or estimating using other methods such as least squares.

5. Intangible Benefits

Some benefits that accrue to the organization from the use of the information system are difficult to measure but are important, nonetheless. They are known as intangible benefits.

Intangible benefits include improving the decision-making process, enhancing accuracy, becoming more competitive in customer service, maintaining a good business image, and increasing job satisfaction for employees by eliminating tedious tasks. As you can judge from the list given, intangible benefits are extremely important and can have far-reaching implications for the business as it relates to people both outside and within the organization.

Although intangible benefits of an information system are important factors that must be considered when deciding whether to proceed with a system, a system built solely for its intangible benefits will not be successful. You must discuss both tangible and intangible benefits in your proposal, because presenting both will allow decision makers in the business to make a well-informed decision about the proposed system.

6. Tangible Costs

The concepts of tangible and intangible costs present a conceptual parallel to the tangible and intangible benefits discussed already. Tangible costs are those that can be accurately projected by the systems analyst and the business's accounting personnel.

Included in tangible costs are the cost of equipment such as computers and terminals, the cost of resources, the cost of systems analysts' time, the cost of programmers' time, and other employee salaries. These costs are usually well established or can be discovered quite easily, and are the costs that will require a cash outlay of the business.

7. Intangible Costs

Intangible costs are difficult to estimate and may not be known. They include losing a competitive edge, losing the reputation for being first with an innovation or the leader in a field, declining company image due to increased customer dissatisfaction, and ineffective decision making due to untimely or inaccessible information.

Chapter 3

System Design

3.1 Design methods

Design methods are procedures, techniques, aids, or tools for designing. They offer a number of different kinds of activities that a designer might use within an overall design process. Conventional procedures of design, such as drawing, can be regarded as design methods, but since the 1950s new procedures have been developed that are more usually grouped together under the name of "design methods". What design methods have in common is that they "are attempts to make public the hitherto private thinking of designers; to *externalise* the design process".

Design methodology is the broader study of method in design: the study of the principles, practices and procedures of designing.

Design methods originated in new approaches to problem solving developed in the mid-20th Century, and also in response to industrialisation and mass-production, which changed the nature of designing. A "Conference on Systematic and Intuitive Methods in Engineering, Industrial Design, Architecture and Communications", held in London in 1962 is regarded as a key event marking the beginning of what became known within design studies as the "design methods movement", leading to the founding of the Design Research Society and influencing design education and practice.

3.2 Various development approaches

Software Development Approach

Software development is the process of creating new software solutions or modifying existing software solutions. Software can be designed and developed in many different ways. The approach taken to design and develop a software package can vary from the very simple, limited planning approach to a very detailed, formal and structured approach. In this topic you will study four commonly used approaches to software development.

- Structured Approach
- Rapid Application Development Approach
- End-User Approach

Selecting an appropriate development approach.

Why develops software? There are obviously millions of reasons to develop a new program; however, the concept of this topic looks at the method. Which approach and language is best? What happens if you get it wrong?

Structured approach

Definition

“Structured Approach to developing a software application involves the use of the program development cycle:

- Defining the problem
- Planning the solution
- Building the solution
- Checking the solution
- Modifying the solution”.

The stages of the development cycle are:

Defining the problem: - It is necessary to understand the problem in as much detail as possible.

Planning the solution: - It is important to begin the planning and design of a solution to the problem. Planning solution involves determining data types, developing algorithms and diagrams, whether the 'start from scratch approach' needed, determine the programming language used.

Building the solution: - Building the solution involves converting the specifications and algorithms developed during the planning stage into code.

Checking the solution: - Once the software solution has been coded it is necessary to check that it operates correctly.

Modifying the solution: - Modification of code may be necessary if errors are detected, or the users' feedback means that the program doesn't meet their needs, or the underlying processes change, or the existing system changes, etc. Accurate documentation is extremely important because the modifications may be made by programmer who were not part of the original team.

Chapter 4

Testing

4.1 Testing techniques and strategies

To perform testing in a planned and systematic manner, software testing strategy is developed. A testing strategy is used to identify the levels of testing which are to be applied along with the methods, techniques, and tools to be used during testing. This strategy also decides test cases, test specifications, test case decisions, and puts them together for execution.

Developing a test strategy, which efficiently meets the requirements of an organization, is critical to the success of software development in that organization. Therefore, a software testing strategy should contain complete information about the procedure to perform testing and the purpose and requirements of testing.

The choice of software testing strategy is highly dependent on the nature of the developed software. For example, if the software is highly data intensive then a strategy that checks structures and values properly to ensure that all inputs given to the software are correct and complete should be developed. Similarly, if it is transaction intensive then the strategy should be such that it is able to check the flow of all the transactions. The design and architecture of the software are also useful in choosing testing strategy. A number of software testing strategies are developed in the testing process. All these strategies provide the tester a template, which is used for testing. Generally, all testing strategies have following characteristics.

Testing proceeds in an outward manner. It starts from testing the individual units, progresses to integrating these units, and finally, moves to system testing.

Testing techniques used during different phases of software development are different.

Testing is conducted by the software developer and by an ITG.

Testing and debugging should not be used synonymously. However, any testing strategy must accommodate debugging with itself.

Types of Software Testing Strategies

There are different types of software testing strategies, which are selected by the testers depending upon the nature and size of the software. The commonly used software testing strategies are listed below.

Analytic testing strategy: This uses formal and informal techniques to access and prioritize risks that arise during software testing. It takes a complete overview of requirements, design, and implementation of objects to determine the motive of testing. In addition, it gathers complete information about the software, targets to be achieved, and the data required for testing the software.

Model-based testing strategy: This strategy tests the functionality of the software according to the real-world scenario (like software functioning in an organization). It recognizes the domain of data and selects suitable test cases according to the probability of errors in that domain.

Methodical testing strategy: It tests the functions and status of software according to the checklist, which is based on user requirements. This strategy is also used to test the functionality, reliability, usability, and performance of the software.

Process-oriented testing strategy: It tests the software according to already existing standards such as the IEEE standards. In addition, it checks the functionality of the software by using automated testing tools.

Dynamic testing strategy: This tests the software after having a collective decision of the testing team. Along with testing, this strategy provides information about the software such as test cases used for testing the errors present in it.

Philosophical testing strategy: It tests the software assuming that any component of the software can stop functioning anytime. It takes help from software developers, users and systems analysts to test the software.

4.2 Debugging & Improvement

Debugging, in computer programming and engineering, is a multistep process that involves identifying a problem, isolating the source of the problem, and then either correcting the problem or determining a way to work around it. The final step of debugging is to test the correction or workaround and make sure it works.

The debugging process: -

In software development, debugging involves locating and correcting code errors in a computer program. Debugging is part of the software testing process and is an integral part of the entire software development lifecycle. The debugging process starts as soon as code is written and continues in successive stages as code is combined with other units of programming to form a software product. In a large program that has thousands and thousands of lines of code, the debugging process can be made easier by using strategies such as unit tests, code reviews and pair programming.

Once an error has been identified, it is necessary to actually find the error in the code. At this point, it can be useful to look at the code's logging and use a stand-alone debugger tool or the debugging component of an integrated development environment (IDE). Invariably, the bugs in the functions that get most use are found and fixed first. In some cases, the module that presents the problem is obvious, while the line of code itself is not. In that case, unit tests – such as JUnit

and xUnit, which allow the programmer to run a specific function with specific inputs -- can be helpful in debugging.

The standard practice is to set up a "breakpoint" and run the program until that breakpoint, at which time program execution stops. The debugging component of an IDE typically provides the programmer with the capability to view memory and see variables, run the program to the next breakpoint, execute just the next line of code, and, in some cases, change the value of variables or even change the contents of the line of code about to be executed.

Common debugging tools

Source code analysers, which include security, common code errors and complexity analysers, can also be helpful in debugging. A complexity analyser can find modules that are so intricate as to be hard to understand and test. Some tools can actually analyse a test run to see what lines of code were not executed, which can aid in debugging. Other debugging tools include advanced logging and simulators that allow the programmer to model how an app on a mobile device will display and behave.

Finding and removing software errors.

Some tools, especially open-source tools and scripting languages, do not run in an IDE and require a more manual approach to debugging. Such techniques include dropping values to a log, extensive "print" statements added during code execution or hard-coded "wait" commands that simulate a breakpoint by waiting for keyboard input at specific times.

The use of the word bug as a synonym for error originated in engineering. The term's application to computing and the inspiration for using the word debugging as a synonym for Troubleshooting has been attributed to Admiral Grace Hopper, a pioneer in computer programming, who was also known for her dry sense of humour. When an actual bug (a moth) got caught between electrical relays and caused a problem in the U.S. Navy's first computer,

Admiral Hopper and her team "debugged" the computer and saved the moth. It now resides in the Smithsonian Museum.

4.3 Finished Code (Home Page)

```
package com.e.blood_bank;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.preference.PreferenceManager;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.android.volley.AuthFailureError;
import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.e.blood_bank.Adapter.RequestAdapter;
import com.e.blood_bank.DataModela.RequestDataModel;
import com.e.blood_bank.Utills.Endpoints;
import com.e.blood_bank.Utills.VolleySingleton;
import com.google.android.material.bottomnavigation.BottomNavigationView;
import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;

import java.lang.reflect.Type;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Objects;

public class MainActivity extends AppCompatActivity {
```



```

private Toolbar top;
private BottomNavigationView bottom;
private RecyclerView recyclerView;
private List<RequestDataModel> requestDataModels;
private RequestAdapter requestAdapter;
private Button button;

```

```

@Override

```

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

```

```

    //initialize

```

```

    top = findViewById(R.id.topCont);
    bottom = findViewById(R.id.bottombar);
    recyclerView = findViewById(R.id.Recycle);
    requestDataModels = new ArrayList<>();
    requestAdapter = new RequestAdapter(requestDataModels,this);
    button = findViewById(R.id.Request);

```

```

    //On Click

```

```

    top.setOnMenuItemClickListener(new Toolbar.OnMenuItemClickListener() {
        @Override
        public boolean onMenuItemClick(MenuItem item) {

            if (item.getItemId() == R.id.logout) {
                Intent intent1 = new Intent(MainActivity.this, WelcomeScreen.class);
                startActivity(intent1);
                MainActivity.this.finish();
                Toast.makeText(MainActivity.this, "LOGGING OUT...",
Toast.LENGTH_SHORT).show();

                } else if (item.getItemId() == R.id.Settings) {
                    Intent intent2 = new Intent(MainActivity.this, SettingActivity.class);
                    startActivity(intent2);
                    Toast.makeText(MainActivity.this, "SOON IN ACTION...",
Toast.LENGTH_SHORT).show();

                } else if(item.getItemId()== R.id.search) {
                    Intent intent3 = new Intent(MainActivity.this, SearchActivity.class);
                    startActivity(intent3);

                }
            }
        }
    });

```

```

        return false;
    }
});

bottom.setOnNavigationItemSelectedListener(new
BottomNavigationView.OnNavigationItemSelectedListener() {
    @Override
    public boolean onNavigationItemSelectedListener(@NonNull MenuItem menuItem) {
        if(menuItem.getItemId()==R.id.plasma)
        {
            Toast.makeText(MainActivity.this, "SOON IN ACTION...",
Toast.LENGTH_SHORT).show();
            Intent intent = new Intent(MainActivity.this,PlasmaActivity.class);
            startActivity(intent);
            MainActivity.this.finish();
        }
        else if(menuItem.getItemId()==R.id.map)
        {
            Toast.makeText(MainActivity.this, "SOON IN ACTION...",
Toast.LENGTH_SHORT).show();
            Intent intent = new Intent(MainActivity.this,MapActivity.class);
            startActivity(intent);

        }
        return false;
    }
});

button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(MainActivity.this,Request_Blood.class);
        startActivity(intent);
    }
});

//Recycler view

LinearLayoutManager layoutManager= new
LinearLayoutManager(this,RecyclerView.VERTICAL,false);
recyclerView.setLayoutManager(layoutManager);
recyclerView.setAdapter(requestAdapter);
populateHomePage();
TextView pick_location = findViewById(R.id.Location);
String location =
PreferenceManager.getDefaultSharedPreferences(this).getString("city","No_city_found");

```

```

        if(!location.equals("No_city_found"))
        {
            pick_location.setText(location);
        }

    }

    private void populateHomePage()
    {
        final String city =
PreferenceManager.getDefaultSharedPreferences(getApplicationContext()).getString("city","
no_city");

        StringRequest stringRequest = new StringRequest(Request.Method.POST,
Endpoints.getreq, new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {

                Gson gson= new Gson();
                Type type = new TypeToken<List<RequestDataModel>>(){}.getType();
                List<RequestDataModel> dataModels = gson.fromJson(response,type);

                requestDataModels.addAll(dataModels);
                requestAdapter.notifyDataSetChanged();

            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                Toast.makeText(MainActivity.this, "Oops something went wrong :",
Toast.LENGTH_SHORT).show();
                Log.d("Volley", Objects.requireNonNull(error.getMessage()));
            }
        })
    {
        @Override
        protected Map<String, String> getParams() throws AuthFailureError {
            Map<String,String> params = new HashMap<>();
            // params.put("city",city);
            return params;
        }
    };
    VolleySingleton.getInstance(this).addToRequestQueue(stringRequest);

}

```

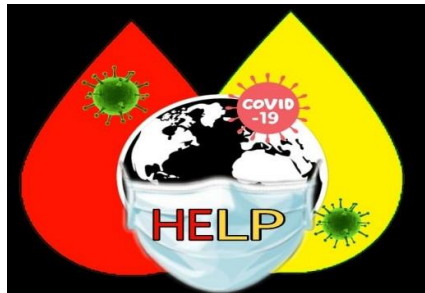
```
}
```

Finished Code (End Points)

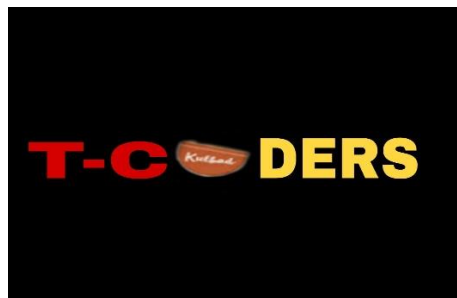
```
package com.e.blood_bank.Utills;

public class Endpoints {
    private static final String base_url
    ="https://tcoders1.000webhostapp.com/";
    public static final String register_url
    =base_url+"register.php";
    public static final String login_url =base_url+"login.php";
    public static final String upload_url =
    base_url+"upload_url.php";
    public static final String getreq =
    base_url+"get_requests.php";
    public static final String search_donors =
    base_url+"search_donors.php";
}
```

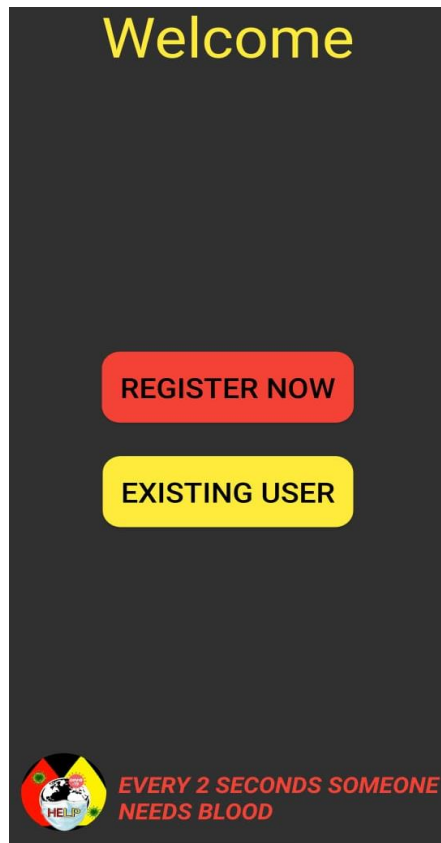
Outputs



(Logo)




(T-coders logo)



(Welcome Screen)




(Splash Screen)



NOTE:- MAKE SURE YOU ARE 18+ FOR DONATING


ENTER YOUR MOBILE NUMBER

ENTER YOUR PASSWORD 

FORGOT PASWORD?

LOGIN

(Login)




NOTE:- DETAILS ENTERED WILL BE PUBLICLY SHARED

ENTER YOUR NAME

ENTER YOUR CITY

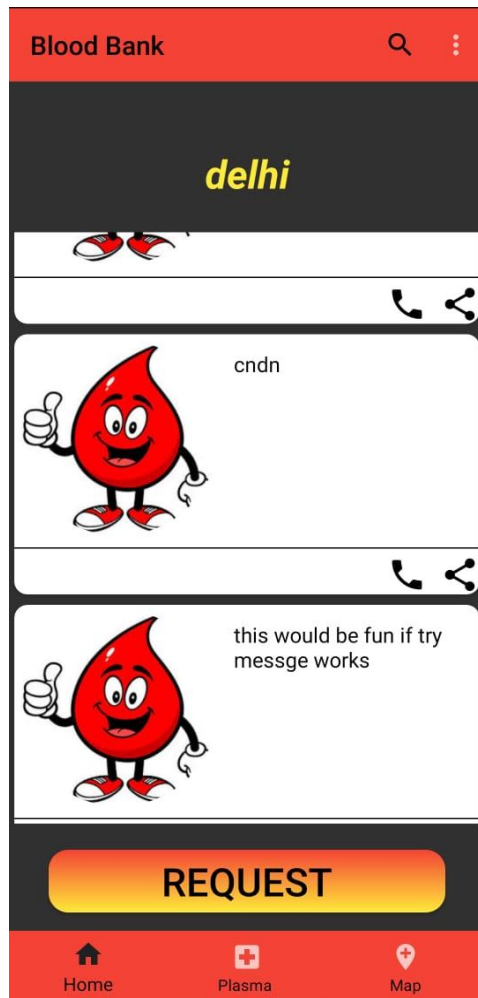
ENTER YOUR NUMBER

ENTER YOUR BLOOD GROUP

ENTER YOUR PASSWORD 

REGISTER


(Register)



(Home Page)



(Request Page)



Be calm you will find help here

Choose city

ENTER CITY NAME

Blood group


ENTER BLOOD GROUP

FIND DONORS


(Find Donors)

Search Results

Donors in delhi with blood group b+



Name archi
city: delhi



Name archit
city: delhi

(Search Donors)



(Share Request)



A great start is half the work

Name Your Project

blood-bank

bmiet1999

☒ Show password [GENERATE PASSWORD](#)

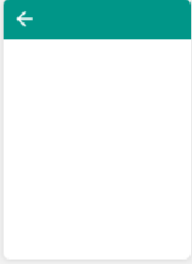
SUBMIT

(000WebHost)

Create New Project

×

Configure your project



Empty Activity

Creates a new empty activity

Name

Package name

Save location

Language

Minimum API level

i Your app will run on approximately **73.7%** of devices.
[Help me choose](#)

☐ This project will support instant apps

☒ Use androidx.* artifacts

Previous

Next

Cancel

Finish

(Project Prerequisites)

Chapter 5

Implementation

5.1 System Implementation

In pursuit of our number one project goal – to achieve exceptional results for our customers – we follow a standard system implementation framework, to ensure that the Lucas system is configured and implemented quickly and cleanly. The Lucas project management approach and implementation process have been developed and refined over the course of hundreds of projects.

First, we work with you to determine the exact scope and plan for the project. Next, we configure the system to meet the needs that we jointly identify and test it to make sure that it will. Finally, we install the system and train your team to guarantee a successful go-live. While we have completed projects in as few as 30 days, a typical implementation timeline is between 60 and 90 days, depending on the complexity of the system and the availability of the customer team.

8. Project Initiation and System Design

Following operational analysis and process design, and upon agreement to proceed with the system implementation, Lucas will assemble a project team and begin the system design process. During initial project kick-off meetings we will discuss project communications, and provide an initial project plan and milestone schedule. We will also begin to discuss on-site installation and training plans.

The Lucas project team will work with your team to create detailed functional requirements for your mobile work execution system, including the mobile user workflows, management reporting screens, interfaces, business logic, and any customer-specific functionality that will be incorporated in the system.

Upon completion of the functional requirements phase we will provide an updated process flow document and final interface specification. Our engineering team will then complete a detailed system design, which will determine the specific business rules and configuration options that will be incorporated into your system, and develop test plans and initial system documentation.

9. *Configuration and Testing*

Following detailed design, the Lucas team will assemble your system using our pre-built functional blocks that are designed to accommodate customer-specific business rules and configurations without custom development.

As the functional blocks are configured and assembled, Lucas will unit test the individual modules. We will develop integration test plans in consultation with your team, and we will jointly define a final installation and start up plan. The customer project team will continue to prepare the site for installation as the Lucas system is transferred to the Lucas QA department for functional and performance testing. Upon completion of remote testing, the system is ready for installation and roll out.

5.2 Software implementation

10. Structured Programming

In the process of coding, the lines of code keep multiplying, thus, size of the software increases. Gradually, it becomes next to impossible to remember the flow of program. If one forgets how software and its underlying programs, files, procedures are constructed it then becomes very difficult to share, debug and modify the program. The solution to this is structured programming. It encourages the developer to use subroutines and loops instead of using simple jumps in the code, thereby bringing clarity in the code and improving its efficiency. Structured programming also helps programmer to reduce coding time and organize code properly.

Structured programming states how the program shall be coded. Structured programming uses three main concepts:

- Top-down analysis - A software is always made to perform some rational work. This rational work is known as problem in the software parlance. Thus it is very important that we understand how to solve the problem. Under top-down analysis, the problem is broken down into small pieces where each one has some significance. Each problem is individually solved and steps are clearly stated about how to solve the problem.
- Modular Programming - While programming, the code is broken down into smaller group of instructions. These groups are known as modules, subprograms or subroutines. Modular programming based on the understanding of top-down analysis. It discourages jumps using 'goto' statements in the program, which often makes the program flow non-traceable. Jumps are prohibited and modular format is encouraged in structured programming.

- Structured Coding - In reference with top-down analysis, structured coding sub-divides the modules into further smaller units of code in the order of their execution. Structured programming uses control structure, which controls the flow of the program, whereas structured coding uses control structure to organize its instructions in definable patterns.

11. Functional Programming

Functional programming is a style of programming language, which uses the concepts of mathematical functions. A function in mathematics should always produce the same result on receiving the same argument. In procedural languages, the flow of the program runs through procedures, i.e. the control of program is transferred to the called procedure. While control flow is transferring from one procedure to another, the program changes its state. In procedural programming, it is possible for a procedure to produce different results when it is called with the same argument, as the program itself can be in different state while calling it. This is a property as well as a drawback of procedural programming, in which the sequence or timing of the procedure execution becomes important.

Functional programming provides means of computation as mathematical functions, which produces results irrespective of program state. This makes it possible to predict the behaviour of the program.

12. Programming style

Programming style is a set of coding rules followed by all the programmers to write the code. When multiple programmers work on the same software project, they frequently need to work with the program code written by some other developer. This becomes tedious or at times

impossible, if all developers do not follow some standard programming style to code the program.

An appropriate programming style includes using function and variable names relevant to the intended task, using well-placed indentation, commenting code for the convenience of reader and overall presentation of code. This makes the program code readable and understandable by all, which in turn makes debugging and error solving easier. Also, proper coding style helps ease the documentation and updating.

5.2.1 Current uses

The current uses of this embodies the apparent use and approaches of the metrological department.

5.2.2 Ongoing research and development

Development of portal for blood and plasma donors, address and blood group is seamless, from global to regional and local scales, which can be used for better and benefit assessment and decision making.

CONCLUSION AND FUTURE USES

Blood and plasma donation bank is a powerful application for donating or requesting blood or plasma from the society, as youth is more connected on social platforms, thus this application gives user a opportunity to share the details on social media so that a social connection can be created and needful person can be saved.

Future scope of the application is for medical purposes, the user information by this application can be further used for online lab testing, as in blood samples can be collected from home itself for testing purposes, this will further help to prevent communicable deceases as there would be least chances of social gatherings around hospitals.

.

REFERENCES

- <https://www.github.com>
- Android programming for beginners' book by John Horton
- <https://www.browserstack.com>
- <https://www.developer.android.com>
- <https://www.Vogella.com>
- <https://www.android.com>
- <https://www.Learn Php.com>
- <https://www.CodeCourse.com>
- <https://www.SidePoint PHP.com>
- <https://www.000WebHost.com>
- <https://www.androidcentral.com>
- <https://www.stackoverflow.com>