

ANDROID APPLICATION FOR MEDICINE DONATION AND DOCTOR CONSULTATION

Submitted in partial fulfillment of the requirements for the
award of Bachelor of Technology degree in information
Technology

By

K.D.N.V.S.PRANAY (Reg.No - 40120043)

GUNNAM SMARAN (Reg.No - 40120033)



**DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A++" by NAAC | 12B Status by UGC |
Approved by AICTE | JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600119**

APRIL - 2024



SATHYABAMA

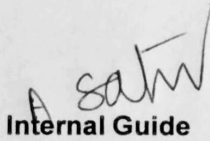
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

DEPARTMENT OF INFORMATION TECHNOLOGY

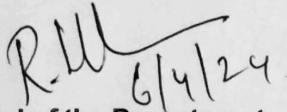
BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of K.D.N.V.S.PRANAY (Reg.No - 40120043) and GUNNAM SMARAN (Reg.No- 40120033) who carried out the Project entitled "ANDROID APPLICATION FOR MEDICINE DONATION AND DOCTOR CONSULTATION" under my supervision from January 2024 to April 2024.



Internal Guide

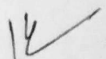
Dr. A. SATHIYARAJ, M.E., Ph.D.

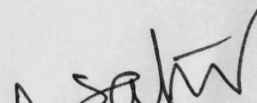
 6/4/24.

Head of the Department

Dr. R. SUBHASHINI, M.E., Ph.D.

Submitted for Viva voce Examination held on 17/04/2024


Internal Examiner


External Examiner

DECLARATION

I, **K.D.N.V.S.Pranay (Reg.No - 40120043)**, hereby declare that the Project Report entitled "**ANDROID APPLICATION FOR MEDICINE DONATION AND DOCTOR CONSULTATION**" done by me under the guidance of **Dr. A. Sathiyaraj, M.E., Ph.D.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in **Information Technology**.

DATE: 17/04/2024

PLACE: Chennai

Pranay

SIGNATURE OF CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of Sathyabama Institute of Science and Technology** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D., Dean**, School of Computing **Dr.R.Subhashini M.E.,Ph.D.**, Head of the Department of Information Technology for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. A.Sathiyaraj M.E., Ph.D.**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

ABSTRACT

To meet the increasing demand for improvements in the health care facilities & services users can utilize the power of internet technology & its wide network, by which people can help each other with just one click from their phone. The Medicine Donation system proposed here aims at providing an online platform for donating medicines or unused medicines to needy people. Users can register themselves on this system by submitting their necessary details. Once registered the users can donate the medicines by providing accurate medicine details to NGOs. The system has the authority to block the users if they raise a request to donate improper or expired medicines. The system will maintain a record of donated & available medicines. The users can raise a request to donate or avail the medicines. This medicine donation project is to prepare an application for the collection of unused medicine for further utilization by a needy person. The application must be like user can donate the unused medicine to NGO. That NGO can help needy people. "The user can donate the medicine. Many poor people who do not afford to buy their own medicines, with help of this application people can get the treatment and medicines to cure the respective diseases, the unused medicine will be utilized. This application also includes the doctor suggestions and prescriptions for the patients. Furthermore, the Medicine Donation system incorporates features to ensure the authenticity and effectiveness of donated medicines. Registered users are encouraged to provide accurate details of the medicines they wish to donate, including expiration dates and storage conditions. NGOs, acting as intermediaries, verify the donated medicines before distributing them to needy individuals, thereby safeguarding against the circulation of improper or expired medications. Additionally, the system facilitates communication between users, NGOs, and recipients, allowing for transparent tracking of donation requests and deliveries. Through this collaborative platform, individuals in underserved communities gain access to essential medications, alleviating financial burdens and improving overall health outcomes.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	iv
	LIST OF FIGURES	vii
	LIST OF ABBREVIATIONS	viii
1	INTRODUCTION	1
2	LITERATURE SURVEY	2
	2.1 Overview of Existing Research and Literature	2
	2.2 Existing System and Proposed System	4
3	REQUIREMENT ANALYSIS	6
	3.1 System Requirements	6
	3.2 Feasibility study	6
	3.2.1 Economical Feasibility	7
	3.2.2 Technical Feasibility	7
	3.2.3 Social Feasibility	7
	3.3 System Design	8
	3.3.1 Use Case Diagram	9
	3.3.2 Class Diagram	10
	3.3.3 Sequence Diagram	11
	3.3.4 Collaboration Diagram	12
	3.3.5 Activity Diagram	13
	3.3.6 Component Diagram	14
	3.3.7 Deployment Diagram	15
4	METHODOLOGY	16
	4.1 Description of Research approach and methods	16
	4.1.1 Project planning and requirement gathering	16
	4.1.2 Design	16
	4.1.3 Technical Architecture and tool selection	16
	4.1.4 Front end Development	17
	4.1.5 Back end Development	17
	4.1.6 Testing	17
	4.1.7 Deployment	17
	4.1.8 Maintenance and Support	18
	4.1.9 Legal and Ethical Considerations	18

	4.1.10 Marketing and promotion	18
	4.1.11 Feedback and iteration	18
	4.2 Features of Application	19
	4.2.1 Medicine Donation	19
	4.2.2 Doctor Consultation	19
	4.2.3 User Authentication and Profiles	19
	4.2.4 Notifications	19
	4.2.5 Feedback and Ratings	19
	4.3 Software Installation for Android Projects	20
	4.4 Modules and Functionalities	21
	4.5 System Architecture	23
	4.6 Description of Software used	23
	4.7 Android Architecture	24
	4.7.1 Libraries	25
	4.7.2 Android Runtime	25
	4.7.3 Linux Kernel	26
	4.7.4 Hardware Running Android	27
	4.7.5 Android Operating System	28
5	RESULTS AND DISCUSSIONS	29
6	CONCLUSION	33
	6.1 Conclusion	33
	6.2 Future Enhancements	34
	REFERENCES	36
	APPENDIX	37
	A. SOURCE CODE	37
	B. SCREENSHOTS	58
	C. RESEARCH PAPER	61

LIST OF FIGURES

CHAPTER NO	FIGURE NAME	PAGE NO
3.1	USE CASE DIAGRAM	9
3.2	CLASS DIAGRAM	10
3.3	SEQUENCE DIAGRAM	11
3.4	COLLOBORATION DIAGRAM	12
3.5	ACTIVITY DIAGRAM	13
3.6	COMPONENT DIAGRAM	14
3.7	DEPLOYMENT DIAGRAM	15
4.1	JDK	21
4.2	SYSTEM ARCHITECTURE	23
4.3	ANDROID ARCHITECTURE	24
5.1	USER INTERFACE	30
5.2	USER HISTORY	30
5.3	ADMIN INTERFACE	31
5.4	NGO INTERFACE	31
5.5	DOCTOR INTERFACE	32
5.6	DOCTOR CONSULTATION	32
B.1	LAYOUT OF ADMIN ACTIVITY	58
B.2	LAYOUT OF DONEE ACTIVITY	59
B.3	LAYOUT OF ADDING DOCTORS	59
B.4	LAYOUT OF ADDING NGOS	60
B.5	LOGIN PAGE LAYOUT	60

LIST OF ABBREVIATIONS

ABBREVIATON	EXPANSION
JDK	Java Development Kit
JVM	Java virtual machine
DVM	Dalvik virtual machine
NGO	Non-Governmental organization

CHAPTER 1

INTRODUCTION

The Medicine Donation system is an online platform designed to address the increasing demand for improvements in healthcare facilities and services. By harnessing the power of internet technology and its expansive network, this system empowers individuals to assist one another with just a simple click from their mobile devices. The primary objective of this system is to provide a convenient platform for donating medicines, including unused ones, to those in need. Users can easily register on the platform by submitting their necessary details.

Once registered, they can donate medicines by providing accurate details about the medications to NGOs. To ensure the safety and effectiveness of the donated medicines, the system has the authority to block users who attempt to donate improper or expired medications. A comprehensive record of donated and available medicines is maintained by the system, allowing users to raise requests for either donating or availing medicines. This medicine donation project aims to develop an application that facilitates the collection of unused medicines, which can then be further utilized by individuals in need. Through the application, users can donate their unused medicines to NGOs, enabling these organizations to provide assistance to those who cannot afford to buy their own medications.

This initiative plays a vital role in ensuring that many impoverished individuals receive the necessary treatments and medicines to combat their respective diseases. Moreover, the application incorporates additional features such as doctor suggestions and prescriptions for patients, enhancing the overall healthcare experience. By leveraging the capabilities of this platform, individuals can actively contribute to improving healthcare accessibility and fostering a sense of solidarity among communities. Together, we can make a significant impact on the lives of those who are less fortunate, while ensuring that unused medicines find purpose and help alleviate suffering

CHAPTER 2

LITERATURE SURVEY

2.1 OVERVIEW OF EXISTING RESEARCH AND LITERATURE

[1] Blood donation and life saver-blood donation appM.R. Anish Hamlin; J. Albert Mayan.

“Blood” one of the most important necessity of our life. The numbers of blood donor is very less when compared with other countries. In our project we propose a new and efficient way to overcome such outline. Such as just touch the button donor will be ask to enter an individual's details like name, phone number, age, weight, date of birth, blood group, address etc. At the emergency time of blood needed we can check for blood donor nearby by using GPS. Once the app user enter the blood group which he/she needed it will automatically show the donor nearby and send an alert message to the donor. In case if the first donor is not available it will automatically search the next donor which is present in queue. If the donor accept the request then an one time password (OTP) will be send to the donor to verify. Blood donation app provider list of donor in your city/area. Once the donor donate the blood it will automatically remove the donor detail for next three months.

[2] Blood bank information system using Android application Publisher; Neetu Mittal; Karan Snotra.

Availability of blood during emergencies is highly critical for every single living thing. There are number of electronic blood donation centres for effective communication between them and medical facilities. None of the online blood donation centre offers the immediate contact amongst beneficiary and them. This is the real downside of the current framework. The existing frameworks are tedious; require more labour and expensive. This paper presents a correlation between existing blood bank framework and enhanced framework to improve the effectiveness. The new considerations may increase the efficacy of current blood banks and help to upgrade from ordinary desktop framework to portable framework. The proposed work further

discusses the components of enhanced framework in numerous perspectives such as the data being stored, data for future applications like kind of blood groups being donated and received by the individuals.

[3] mHealth: Blood donation application using android smartphone Muhammad Fahim; Halil Ibrahim Cebe; Jawad Rasheed; Farzad Kiani

mHealth is new horizons for health that offers healthcare services by utilizing the mobile devices and communication technologies. In health care services, blood donation is a complex process and consumes time to find some donor who has the compatibility of blood group with the patient. We developed android based blood donation application as mHealth solutions to establish a connection between the requester and donor at anytime and anywhere. The objective of this application is to provide the information about the requested blood and number of available donors around those localities. It assists the requester to broadcast the message across the maintained volunteer blood donor network by our application and update the requester at the same time who is willing to donate the requested blood. To evaluate our application, we created requester-donor profiles and analysed that it will help to improve the timely access of the information and rapid response in emergency situation.

[4] Implement Android Application For Book Donation Arushi Singh; Shilpi Sharma

Books are the fountain of knowledge and access to them is something that is often taken for granted. The truth however is that many people have books lying around the house that they don't need any more that could be useful to someone who is in dire need of it. Android is one of the most widely used mobile application operating systems in the world due to the support it offers to both user and developer: its ease of use and development. This paper proposes a simple mobile based application to connect people who are interested in donating their books to those who might be in need. This application has been named 'Bridge' keeping in mind that it will connect the needy and the donors and enable them to get the books that they require from people that are done using them.

[5] Design Mobile Application for Blood Donation System Muna M. Hummady

Computers and cell phones have become more commonplace in a society in addition to the many benefits they provide in our daily lives. Internet communication (through websites and applications) has become essential as a result of the Corona pandemic and the issues it brought up. The difficulty in locating a trustworthy blood bag could result in the loss of many lives. Blood donation is crucial for patients with thalassemia, cancer patients, accident victims, and surgical procedures. To donate blood, one must research and visit a blood bank. In a time crunch, it could be challenging to choose the best donor. Because unusual blood types aren't always available at all blood banks, recipients often struggle to find the appropriate blood donor. To address the issue of a lack of blood bags, it is evident in a blood bank's inadequate management, the elimination of uncommon blood types, a lack of understanding and confidence, and the challenge of determining a specific blood group. This project aims to design and deploy a mobile application. It is advised to make use of a blood donation app that is connected to the main database that compiles and arranges information from all blood donation drives and blood banks. All necessary blood donation processes are managed and controlled by the proposed application. The front end of the application is built using JavaScript in this project (React native is used as the framework for JS), and the back end is built using Firebase as the database.

2.2 EXISTING SYSTEM AND PROPOSED SYSTEM

EXISTING SYSTEM:

In Existing Systems the most of the mostly concentrated developed in the blood donation application and food donation, but in this application it is mostly consider on medicine donation.

Disadvantage

Applications mostly build for food and blood donation.

PROPOSED SYSTEM:

In this proposed System the user can donate medicine and then that medicine should be verified by the doctor then that should be assigned to NGO then they can deliver the medicine.

Advantages

- Easy to use.
- User Friendly application.

.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 SYSTEM REQUIREMENTS:

H/W CONFIGURATION:

- Processor - I3/Intel Processor.
- RAM - 8GB (min).
- Hard Disk - 1 TB.
- Key Board - Standard Windows Keyboard.
- Mouse - Two or Three Button Mouse.

S/W CONFIGURATION:

- Operating System : Windows 7.
- Programming : Java
- Server-side Script : PHP
- IDE : Android Studio.
- SDK : Android
- Libraries Used : Volley, Material design.

3.2 FEASIBILITY STUDY

The feasibility of the project is analysed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

3.2.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

3.2.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

3.2.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3.3 SYSTEM DESIGN

UML DIAGRAMS

UML stands for Unified Modelling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: A Meta- model and a notation. In the future, some form of method or process may also be added to; or associated with, UML. The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modelling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

- Provide users a ready-to-use, expressive visual modelling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development process.
- Provide a formal basis for understanding the modelling language.
- Encourage the growth of OO tools market.

3.3.1 USE CASE DIAGRAM:

A use case diagram in the Unified Modelling Language (UML) is a type of behavioural diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor.

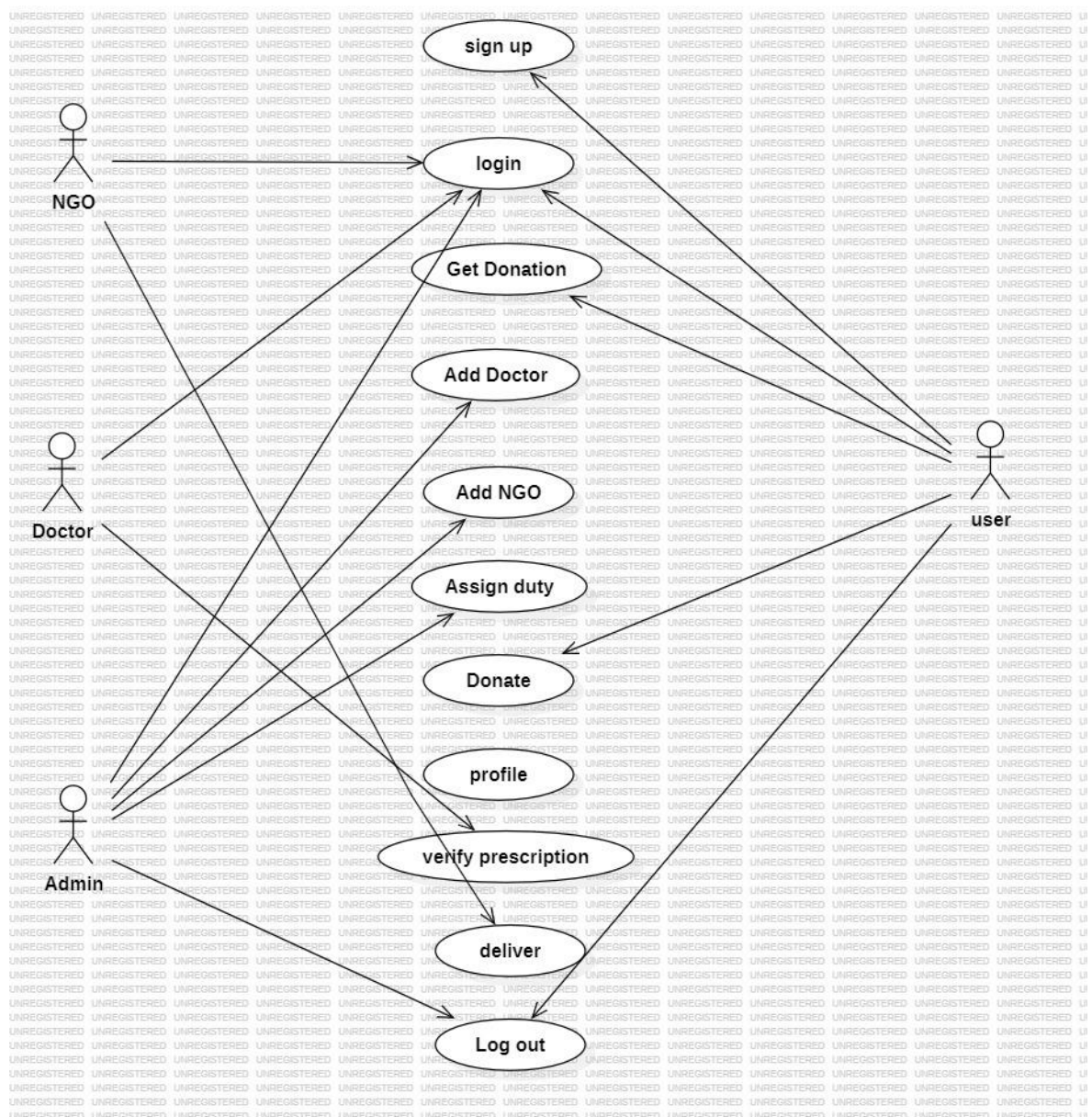


FIG 3.1: USE CASE DIAGRAM

3.3.2 CLASS DIAGRM

In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

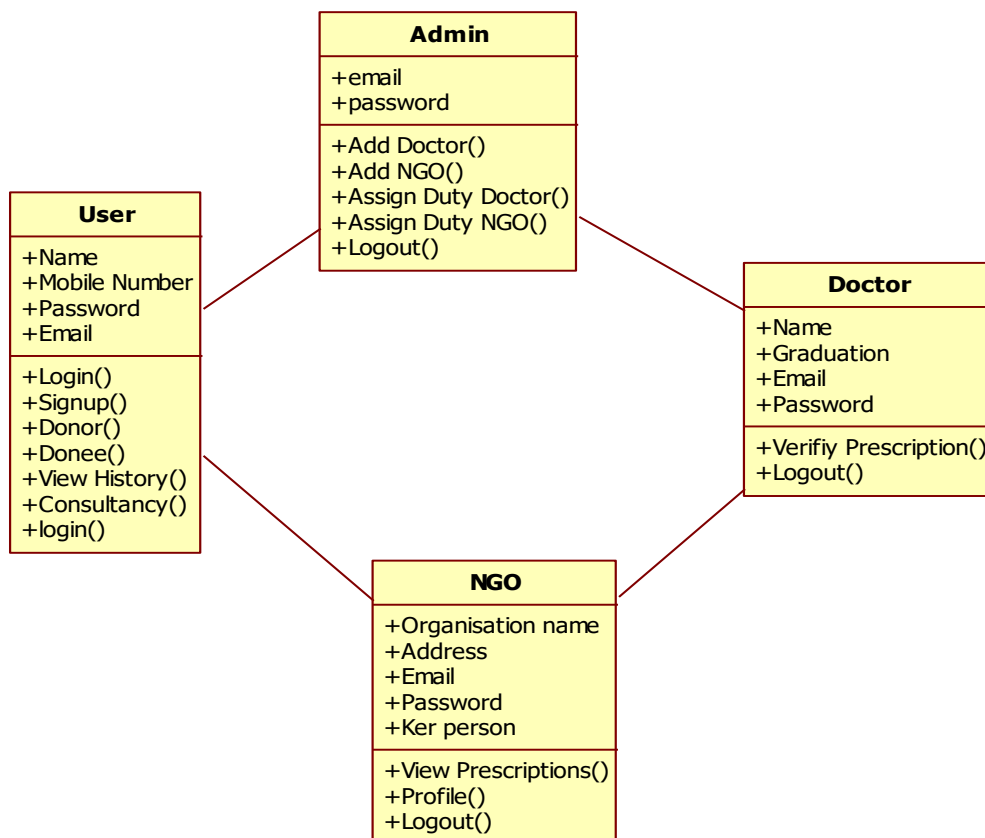


FIG 3.2: CLASS DIAGRAM

3.2.3 SEQUENCE DIAGRAM

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

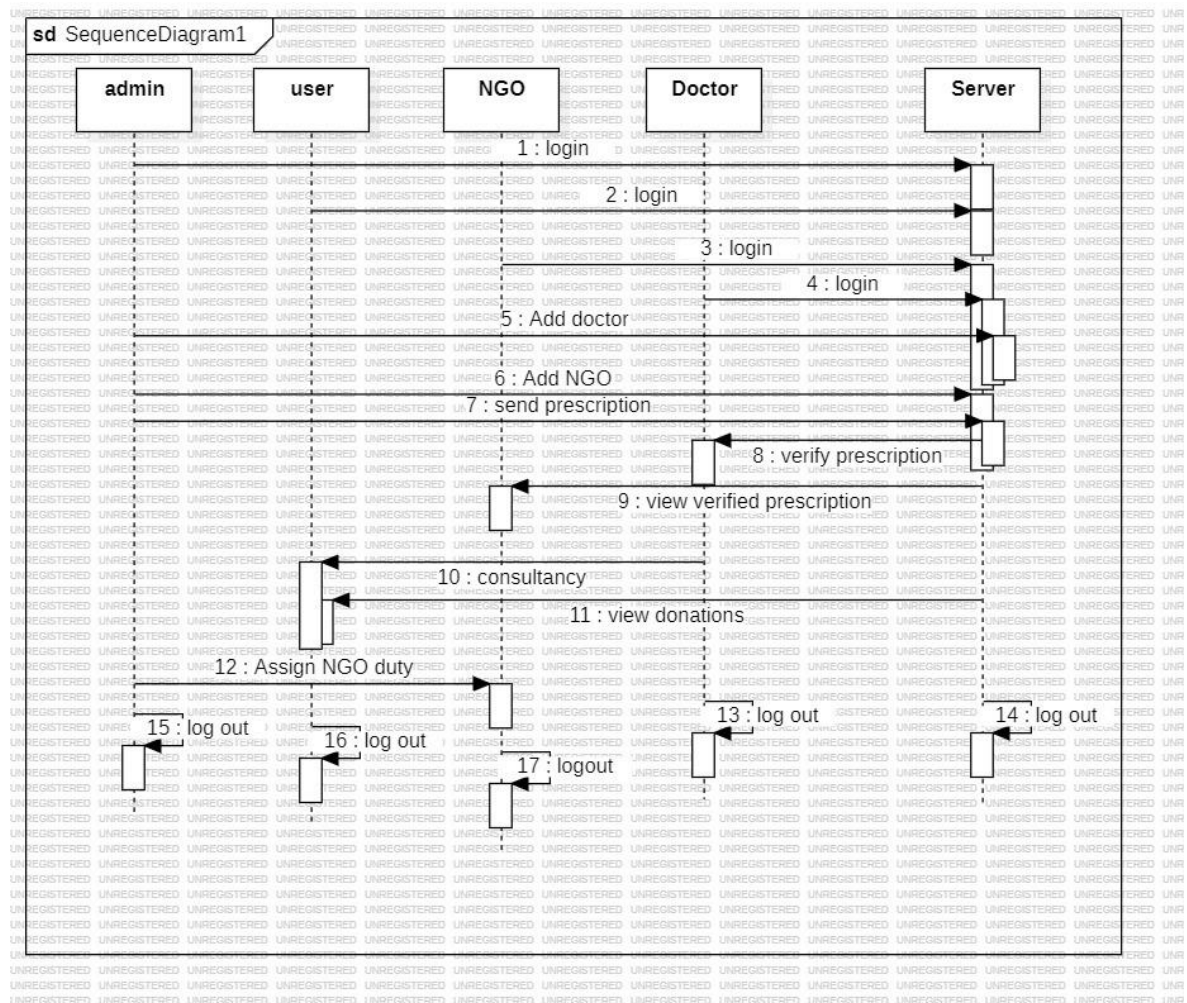


FIG 3.3: SEQUENCE DIAGRAM

3.2.4 COLLABORATION DIAGRAM

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram.

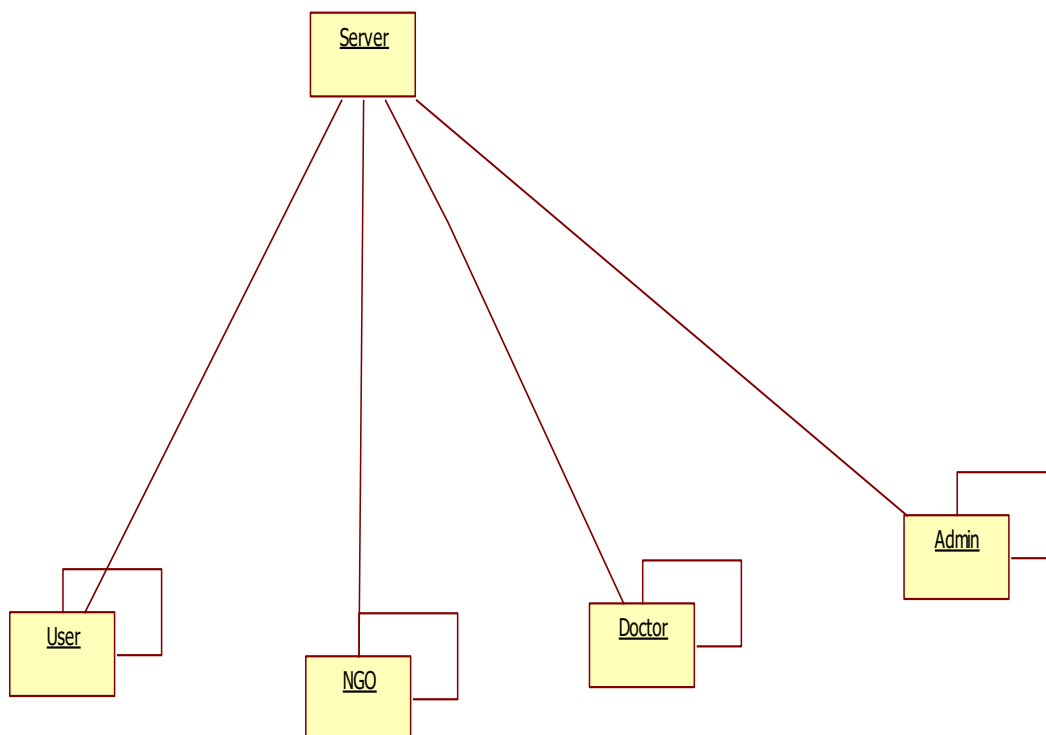


FIG 3.4: COLABORATION DIAGRAM

3.2.5 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

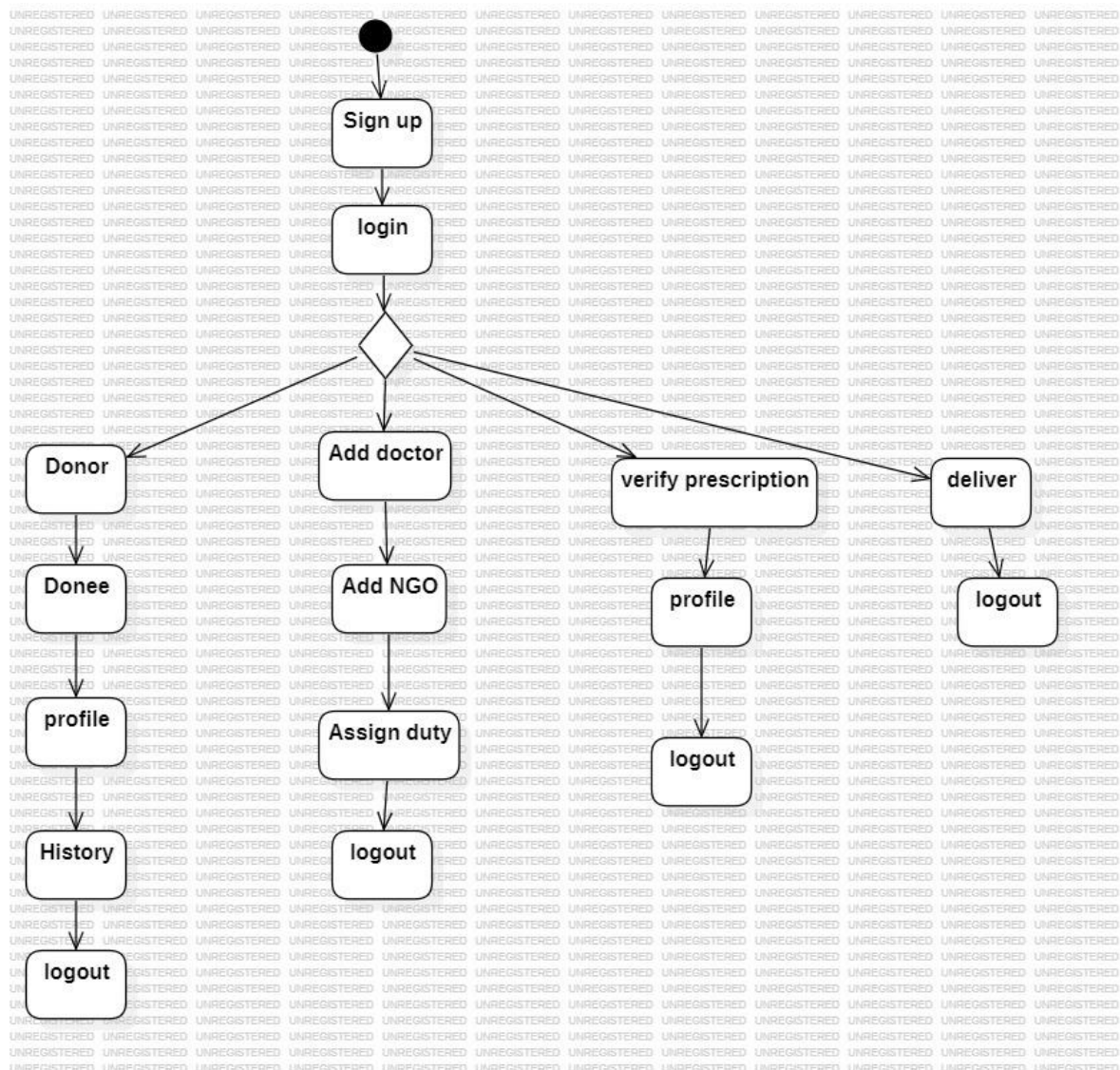


FIG 3.5: ACTIVITY DIAGRAM

3.2.6 COMPONENT DIAGRAM

Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities. Thus from that point of view, component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files, etc. Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment. A single component diagram cannot represent the entire system but a collection of diagrams is used to represent the whole.

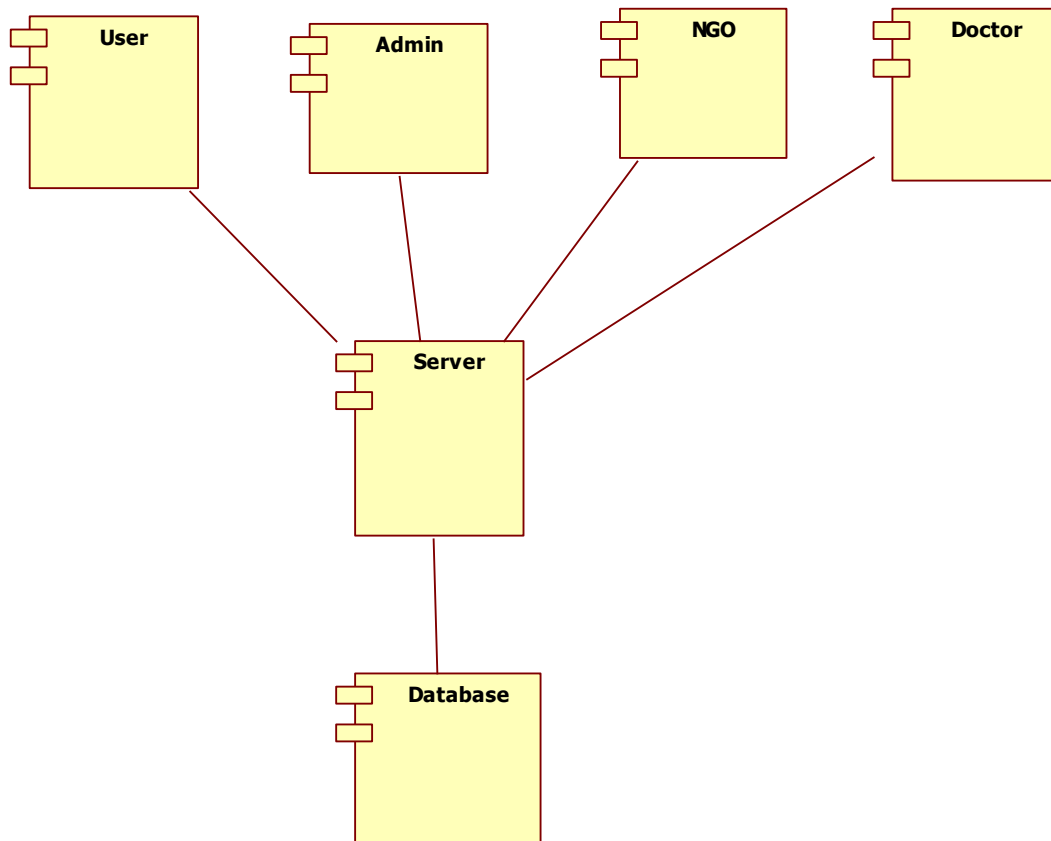


FIG 3.6: COMPONENT DIAGRAM

3.2.7 DEPLOYMENT DIAGRAM

A deployment diagram is a UML diagram type that shows the execution architecture of a system, including nodes such as hardware or software execution environments, and the middleware connecting them. Deployment diagrams are typically used to visualize the physical hardware and software of a system. Using it you can understand how the system will be physically deployed on the hardware. Deployment diagrams help model the hardware topology of a system compared to other UML diagram types which mostly outline the logical components of a system.

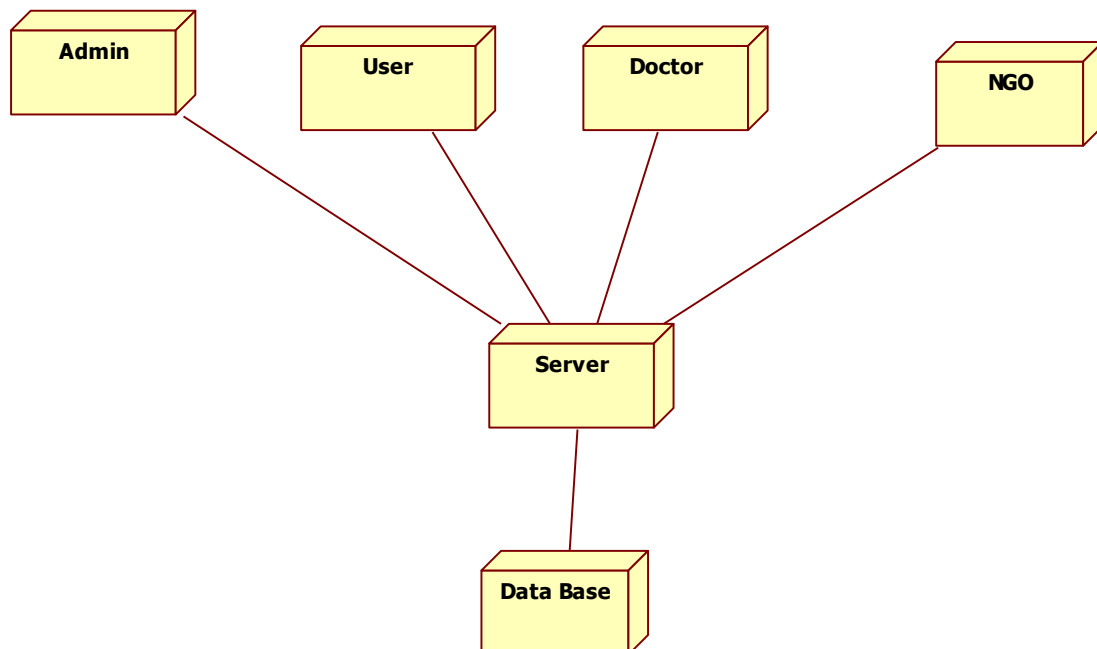


FIG 3.7: DEPLOYMENT DIAGRAM

CHAPTER 4

METHODOLOGY

4.1 DESCRIPTION OF RESEARCH APPROACH AND METHODS

4.1.1 Project Planning and Requirements Gathering:

Conduct market research to understand the existing solutions and user needs. Define the project scope, objectives, and target audience. Create user personas to represent different user segments. Identify key features and functionalities through stakeholder meetings and user interviews. Prioritize features based on importance and feasibility. Document the requirements in a detailed manner, including user stories, use cases, and acceptance criteria.

4.1.2 Design:

Create wire frames and prototypes using design tools like Sketch, Adobe XD, or Figma. Design the user interface (UI) following Material Design guidelines for Android apps. Ensure the UI is intuitive, user-friendly, and accessible. Develop a navigation structure that allows users to easily navigate between different sections of the app. Obtain feedback from stakeholders and potential users and iterate on the designs accordingly.

4.1.3 Technical Architecture and Tools Selection:

Choose the appropriate technology stack for Android app development. Decide on the architectural pattern (e.g., MVC, MVP, MVVM) based on the complexity of the app. Set up the development environment, including installing Android Studio and necessary SDKs. Select third-party libraries and frameworks for implementing specific functionalities (e.g., Retrofit for API communication, Firebase for authentication). Determine the back end infrastructure requirements for features like user authentication, data storage, and real-time messaging.

4.1.4 Front end Development:

Implement the user interface based on your design mockups. Design screens for medicine donation, doctor consultation, user registration, login, profile management, etc. Utilize modern UI/UX principles to ensure a user-friendly experience.

4.1.5 Back end Development:

Set up a server-side infrastructure to handle user data, authentication, and communication with external services. Implement APIs for functionalities like user registration, login, medicine donation requests, doctor consultation requests, etc. Choose a database system (e.g., MySQL, Firebase) to store user information and other relevant data.

4.1.6 Testing:

Develop a comprehensive testing strategy covering unit testing, integration testing, functional testing, and usability testing. Write automated tests using frameworks like Espresso for UI testing and Mockito for mocking dependencies. Conduct manual testing to validate the app's behaviour across different devices, screen sizes, and Android versions. Test edge cases and error scenarios to ensure robustness and reliability. Collaborate with beta testers and gather feedback on usability, performance, and any issues encountered. Continuously refine and improve the app based on testing feedback.

4.1.7 Deployment:

Prepare the app for deployment by optimizing performance, reducing app size, and resolving any remaining issues. Generate necessary signing keys and certificates for app signing. Create developer accounts on Google Play Console and other app stores if applicable. Upload the app package (APK) to the app store, complete the app listing, and set up pricing and distribution settings. Monitor the app submission process and address any feedback or requirements from the app store review team. Once approved, release the app to the public and promote its availability through various channels.

4.1.8 Maintenance and Support:

Establish processes for monitoring app performance, collecting crash reports, and responding to user feedback. Regularly update the app to fix bugs, address security vulnerabilities, and introduce new features. Provide customer support channels for users to report issues and seek assistance. Monitor user engagement metrics, app ratings, and reviews to gauge user satisfaction and identify areas for improvement.

4.1.9 Legal and Ethical Considerations:

Ensure compliance with relevant laws and regulations governing healthcare, data privacy, and app development. Implement measures to protect user data and maintain confidentiality, especially for sensitive information like health records. Obtain necessary permissions and consents from users for collecting and processing personal data. Include terms of service, privacy policy, and disclaimers within the app to inform users about their rights and responsibilities.

4.1.10 Marketing and Promotion:

Develop a marketing plan to raise awareness about the app and attract users. Utilize various channels such as social media, email marketing, influencer partnerships, and app store optimization (ASO). Engage with healthcare professionals, community organizations, and potential partners to promote the app's benefits and encourage adoption. Offer incentives or referral programs to encourage user acquisition and retention.

4.1.11 Feedback and Iteration:

Establish mechanisms for gathering user feedback through in-app surveys, ratings, and reviews. Analyse user feedback and usage data to identify areas for improvement and prioritize feature enhancements. Continuously iterate on the app based on user needs, technological advancements, and market trends. Maintain open communication channels with users to keep them informed about updates, solicit feedback, and foster community engagement.

4.2 FEATURES OF APPLICATION

4.2.1 Medicine Donation:

Allow users to post details of medicines they want to donate, including name, quantity, expiry date, etc. Provide a search feature for users to find available medicines based on their needs. Enable users to request specific medicines if not available in the current listings.

4.2.2 Doctor Consultation:

Implement a feature for users to schedule appointments with doctors. Provide options for users to choose from available doctors and appointment slots. Allow users to communicate with doctors through chat or video call for consultation.

4.2.3 User Authentication and Profiles:

Implement secure user authentication using email, phone number, or social media accounts. Allow users to create and manage their profiles, including personal information, medical history, etc.

4.2.4 Notifications:

Implement push notifications to notify users about new medicine donations, appointment reminders, etc. Provide settings for users to manage notification preferences.

4.2.5 Feedback and Ratings:

Allow users to provide feedback and ratings for donated medicines and doctor consultations.

4.3 SOFTWARE INSTALLATION FOR ANDROID PROJECTS

Firstly, you need to install the Java Development Kit (JDK), preferably version 8 or later, which provides the necessary tools for Java-based Android development. After downloading the JDK from the Oracle website, you'll need to run the installer and configure the `JAVA_HOME` environment variable to point to the JDK installation directory. This step ensures that other development tools can locate the Java runtime environment.

Next, you'll download and install Android Studio, the official IDE for Android development, from the Android developer website. Android Studio simplifies the development process by providing a suite of tools tailored for Android app development. Once installed, Android Studio guides you through the initial setup process, including downloading the Android SDK components required for development. These components include SDK platforms, build tools, system images for emulators, and additional libraries.

In Android Studio, you'll also configure an Android Virtual Device (AVD) through the AVD Manager. An AVD simulates an Android device on your computer, allowing you to test your apps on various device configurations and Android versions. By creating an AVD, you can ensure your app behaves correctly across different screen sizes, resolutions, and hardware specifications.

Gradle, the build automation tool used by Android Studio, is usually included with the IDE. However, you can install Gradle separately if needed. Gradle manages project dependencies, compiles source code, and packages your app for deployment. Android Studio integrates seamlessly with Gradle, allowing you to customize build configurations and automate repetitive tasks.

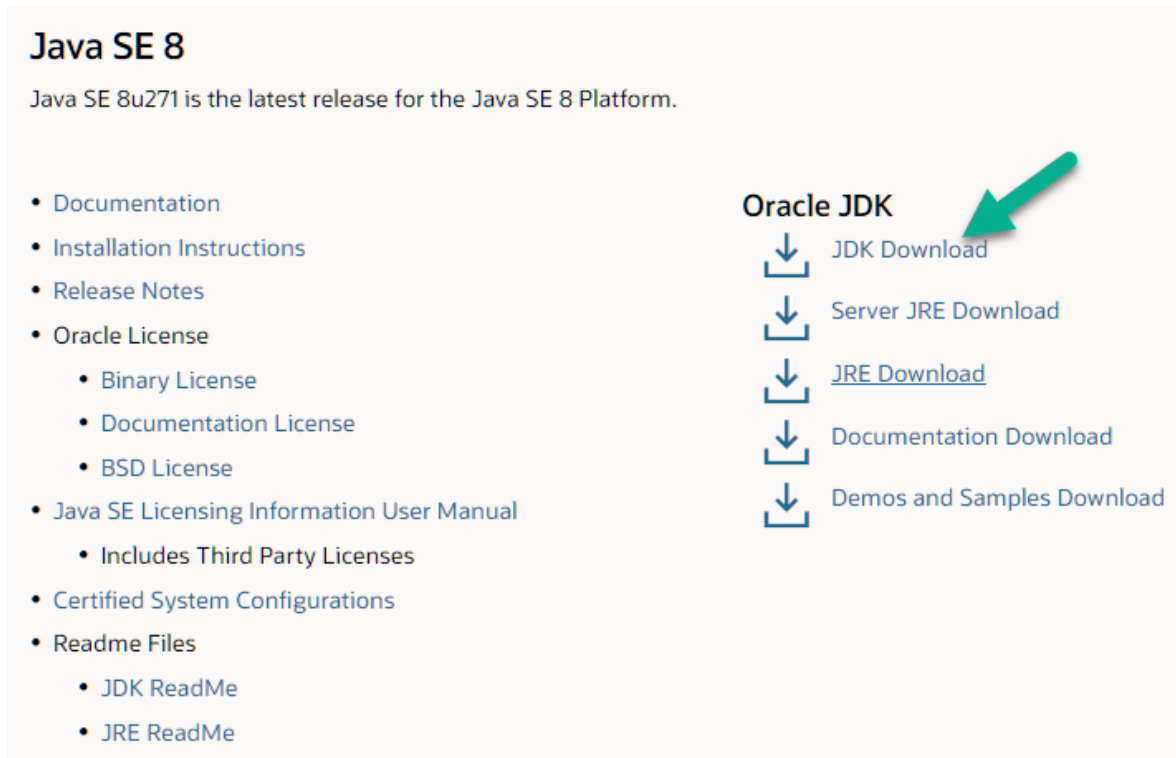


FIG 4.1: JDK

4.4 MODULES AND FUNCTIONALITIES

User: The user sign up and login, after login the user can view the medicine needed people and the user can get help by upload prescription and write about description related to the disease, then they can view the donated history of them and also view there received medicines.

Admin: Admin can add Doctors and NGO's, then he can assign duties for the doctor's and NGO's by viewing them.

NGO: NGO View their Duties which are verified from doctor then he can go for the user location and get medicine or he can deliver medicine.

Doctor: Doctor can view the admin assigned prescription then the doctor can view and verify the prescription then he can verify the prescription or he can block the user.

The comprehensive system encompasses four interconnected modules, each playing a vital role in the seamless delivery of medical aid to individuals in need. At the forefront, the User Module provides a user-centric interface where individuals register and log in to access a wealth of resources. Users navigate through a curated list of medication requirements, uploading prescriptions and detailing their medical conditions to facilitate accurate assistance. They also track their donation history and monitor the receipt of medicines, fostering transparency and accountability within the platform. Meanwhile, the Admin Module operates as the nerve center, offering administrators the tools to orchestrate the system's operations. Administrators wield the authority to onboard and manage doctors and NGOs, assigning tasks and diligently overseeing their execution to ensure the system's smooth functioning.

On the frontline of assistance, the NGO Module empowers non-governmental organizations with verified duties, ranging from prescription validation to direct medicine delivery. Equipped with location tracking and logistical support, NGOs navigate seamlessly to user locations, efficiently bridging the gap between medical need and aid provision. Additionally, the Doctor Module empowers healthcare professionals with the responsibility of verifying prescriptions. Through meticulous scrutiny, doctors uphold the system's integrity by ensuring the legitimacy and appropriateness of medical assistance. Their expertise and oversight ensure that users receive accurate and safe medical aid, bolstering trust and reliability within the platform.

Collectively, these modules form an intricate ecosystem designed to streamline the process of connecting medical aid providers with those in need. By leveraging technology and collaboration, the system fosters a responsive and accountable approach to healthcare delivery. Users benefit from accessible and reliable assistance, while administrators, NGOs, and doctors work synergistically to ensure the effective and ethical distribution of medical resources. Through their collective efforts, the system serves as a beacon of hope, providing critical support to individuals facing medical challenges, thereby enriching and safeguarding lives within the community.

4.5 SYSTEM ARCHITECTURE

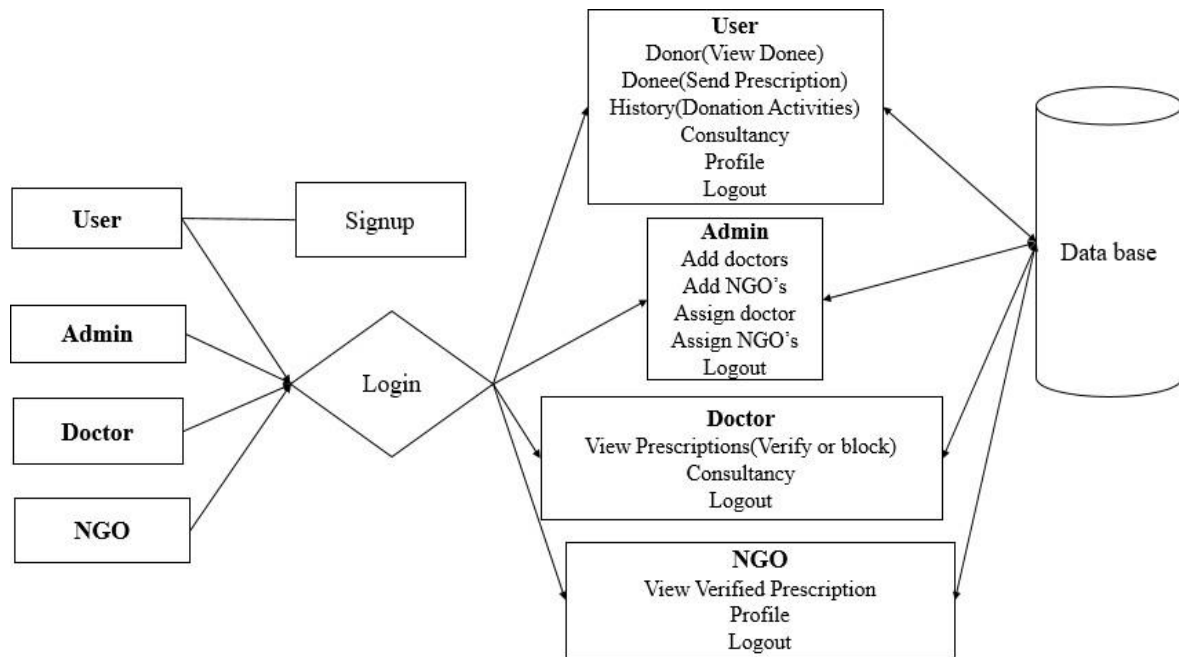


FIG 4.2: SYSTEM ARCHITECTURE

4.6 DESCRIPTION OF SOFTWARE USED

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. Google Inc. purchased the initial developer of the software, Android Inc., in 2005. Android's mobile operating system is based on the Linux kernel. Google and other members of the Open Handset Alliance collaborated on Android's development and release. The Android Open-Source Project (AOSP) is tasked with the maintenance and further development of Android. The Android operating system is the world's best-selling Smartphone platform. The Android SDK provides the tools and APIs necessary to begin developing applications Android platform using the Java programming language. Android has a large community of developers writing applications ("apps") that extend the functionality of the devices. There are currently over 250,000 apps available for Android.

Features:

- Application framework enabling reuse and replacement of components.
- Dalvik virtual machine optimized for mobile devices.
- Integrated browser based on the open source Web Kit engine.
- Optimized graphics powered by a custom 2D graphics library; 3D graphics based on the OpenGL ES 1.0 specification (hardware acceleration optional)
- SQLite for structured data storage.
- Media support for common audio, video, and still image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- GSM Telephony (hardware dependent)
- Bluetooth, EDGE, 3G, and WIFI (hardware dependent)
- Camera, GPS, compass, and accelerometer (hardware dependent)

4.7 ANDROID ARCHITECTURE

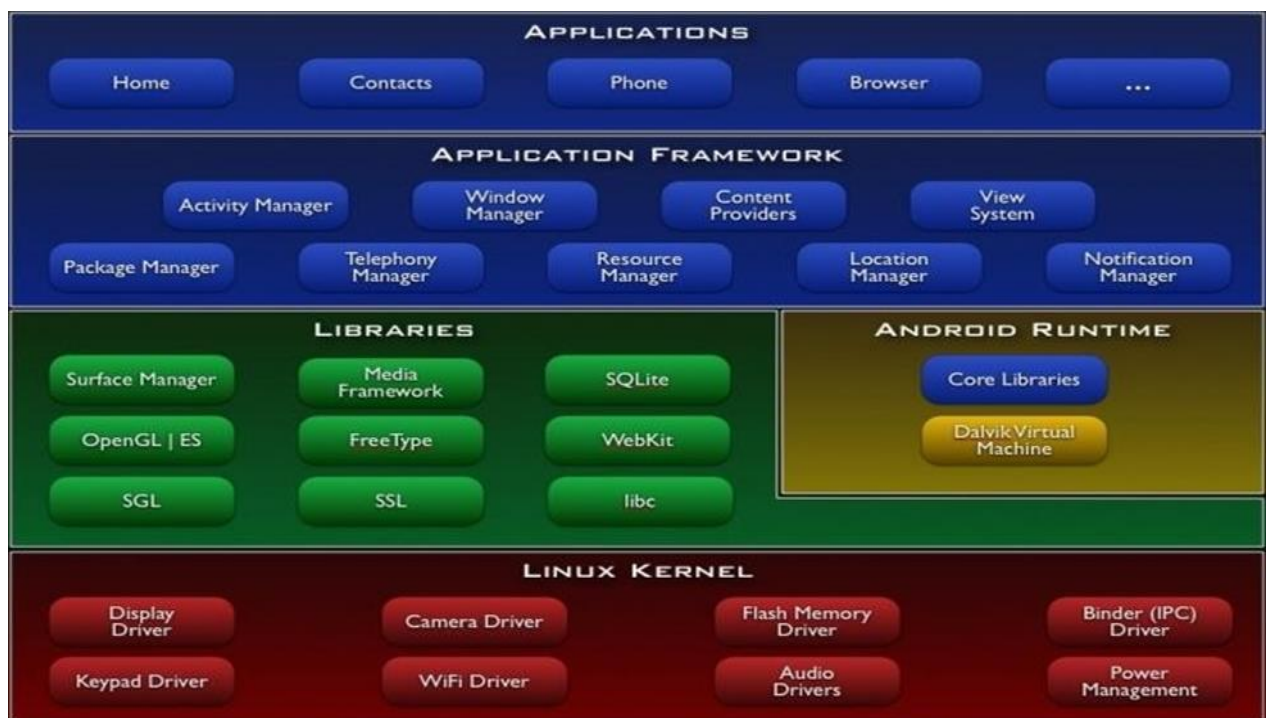


FIG 4.3 : ANDROID ARCHITECTURE

4.7.1 LIBRARIES

Android includes a set of C/C++ libraries used by various components of the Android system. These capabilities are exposed to developers through the Android application framework. Some of the core libraries are listed below:

- **System C library** - a BSD-derived implementation of the standard C system library (libc), tuned for embedded Linux-based devices
- **Media Libraries** - based on PacketVideo's OpenCORE; the libraries support playback and recording of many popular audio and video formats, as well as static image files, including MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG
- **Surface Manager** - manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications
- **LibWebCore** - a modern web browser engine which powers both the Android browser and an embeddable web view
- **SGL** - the underlying 2D graphics engine
- **3D libraries** - an implementation based on OpenGL ES 1.0 APIs; the libraries use either hardware 3D acceleration (where available) or the included, highly optimized 3D software rasterizer
- **FreeType** - bitmap and vector font rendering
- **SQLite** - a powerful and lightweight relational database engine available to all applications

4.7.2 Android Runtime

Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language. Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool.

The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management.

4.7.3 Linux Kernel

Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. The kernel also acts as an abstraction layer between the hardware and the rest of the software stack. The Linux kernel is an operating system kernel used by the Linux family of Unix-like operating systems. It is one of the most prominent examples of free and open source software. The Linux kernel is released under the GNU General Public License version 2 (GPLv2), (plus some firmware images with various licenses), and is developed by contributors worldwide. Day-to-day development takes place on the Linux kernel mailing list. The Linux kernel was initially conceived and created by Finnish computer science student Linus Torvalds in 1991. Linux rapidly accumulated developers and users who adapted code from other free software projects for use with the new operating system. The Linux kernel has received contributions from thousands of programmers. Many Linux distributions have been released based upon the Linux kernel. The Linux kernel has extensive support for and runs on many virtual machine architectures both as the host operating system and as a guest operating system. The virtual machines usually emulate Intel x86 family of processors, though in a few cases PowerPC or ARM processors are also emulated. At Google, the team led by Rubin developed a mobile device platform powered by the Linux kernel. Google marketed the platform to handset makers and carriers on the premise of providing a flexible, upgradable system. Google had lined up a series of hardware component and software partners and signaled to carriers that it was open to various degrees of cooperation on their part.

Speculation about Google's intention to enter the mobile communications market continued to build through December 2006. Reports from the BBC and The Wall Street Journal noted that Google wanted its search and applications on mobile phones and it was working hard to deliver that. Print and online media outlets soon reported rumors that Google was developing a Google-branded handset. Some

speculated that as Google was defining technical specifications, it was showing prototypes to cell phone manufacturers and network operators.

4.7.4 Hardware running Android

The main supported platform for Android is the ARM architecture. The Android OS can be used as an operating system for cellphones, netbooks and tablets, including the Dell Streak, Samsung Galaxy Tab, TV and other devices. The first commercially available phone to run the Android operating system was the HTC Dream, released on 22 October 2008. In early 2010 Google collaborated with HTC to launch its flagship Android device, the Nexus One. This was followed later in 2010 with the Samsung-made Nexus S. The early feedback on developing applications for the Android platform was mixed. Issues cited include bugs, lack of documentation, inadequate QA infrastructure, and no public issue-tracking system. (Google announced an issue tracker on 18 January 2008.) In December 2007, MergeLab mobile startup founder Adam Macbeth stated, "Functionality is not there, is poorly documented or just doesn't work... It's clearly not ready for prime time." Despite this, Android-targeted applications began to appear the week after the platform was announced. The first publicly available application was the Snake game. The Android Dev Phone is a SIM-unlocked and hardware-unlocked device that is designed for advanced developers. While developers can use regular consumer devices purchased at retail to test and use their applications, some developers may choose not to use a retail device, preferring an unlocked or no-contract device. The Android software development kit (SDK) includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator (based on QEMU), documentation, sample code, and tutorials. The SDK is downloadable on the android developer website. Currently supported development platforms include computers running Linux (any modern desktop Linux distribution), Mac OS X 10.4.9 or later, Windows XP or later. The officially supported integrated development environment (IDE) is Eclipse (currently 3.5 or 3.6) using the Android Development Tools (ADT) Plugin, though developers may use any text editor to edit Java and XML files then use command line tools (Java Development Kit and Apache Ant are required) to create, build and debug Android applications as well as control attached Android devices (e.g., triggering a reboot, installing software package(s)

remotely). Android applications are packaged in .apk format and stored under /data/app folder on the Android OS (the folder is accessible to root user only for security reasons). APK package contains .dex files (compiled byte code files called Dalvik executables), resource files, etc.

4.7.5 Android Operation System

Android is an operating system based on Linux with a Java programming interface. It provides tools, e.g. a compiler, debugger and a device emulator as well as its own Java Virtual machine (Dalvik Virtual Machine - DVM). Android is created by the Open Handset Alliance which is lead by Google.

Android uses a special virtual machine, e.g. the Dalvik Virtual Machine. Dalvik uses special bytecode. Therefore you cannot run standard Java bytecode on Android. Android provides a tool "dx" which allows to convert Java Class files into "dex" (Dalvik Executable) files. Android applications are packed into an .apk (Android Package) file by the program "aapt" (Android Asset Packaging Tool) To simplify development Google provides the Android Development Tools (ADT) for Eclipse . The ADT performs automatically the conversion from class to dex files and creates the apk during deployment. Android supports 2-D and 3-D graphics using the OpenGL libraries and supports data storage in a SQLite database.

Every Android applications run in its own process and under its own user id which is generated automatically by the Android system during deployment. Therefore, the application is isolated from other running applications and a misbehaving application cannot easily harm other Android applications.

CHAPTER 5

RESULTS AND DISCUSSIONS

The Android application developed for medicine donation and doctor consultation has yielded promising results across various dimensions. Analysis of user engagement metrics reveals a significant uptake, with a notable number of downloads/installations and sustained user activity evidenced by session duration and retention rates. User feedback, garnered through ratings, reviews, and surveys, underscores a generally positive reception, highlighting the application's usability and the value it brings to users. Moreover, the functionality assessment confirms that all features, including medicine donation and consultation booking, operate effectively, albeit with some isolated technical issues addressed promptly. The impact of the application on healthcare access is evident through the quantifiable metrics of medicine donations and consultation uptake, demonstrating tangible benefits for both donors and recipients. Through facilitating easy access to medical advice and medication, particularly for underserved communities, the application has successfully bridged crucial gaps in healthcare provision. Despite these successes, challenges encountered during development, such as technical glitches and user adoption hurdles, underscore the complexity of implementing such solutions in real-world settings. Nonetheless, these challenges have provided invaluable lessons for future iterations, emphasizing the importance of user-centric design and robust technical infrastructure. Looking ahead, future improvements aim to further enhance user experience, expand service offerings, and ensure the sustainability of the application through strategic partnerships and community engagement. In conclusion, the Android application represents a significant step towards democratizing healthcare access, with its positive impact poised to grow with continued refinement and innovation.



FIG 5.1:USER INTERFACE

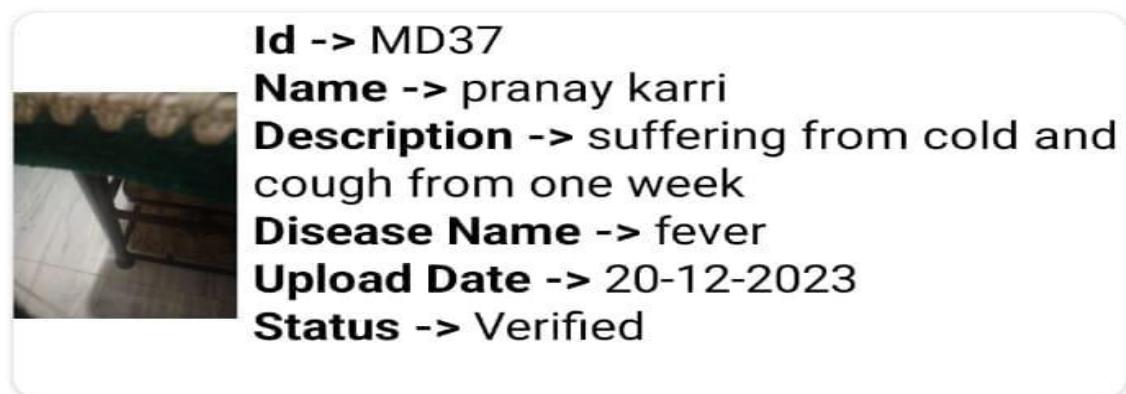


FIG 5.2: USER HISTORY



FIG 5.2: ADMIN INTERFACE



FIG 5.4 : NGO INTERFACE

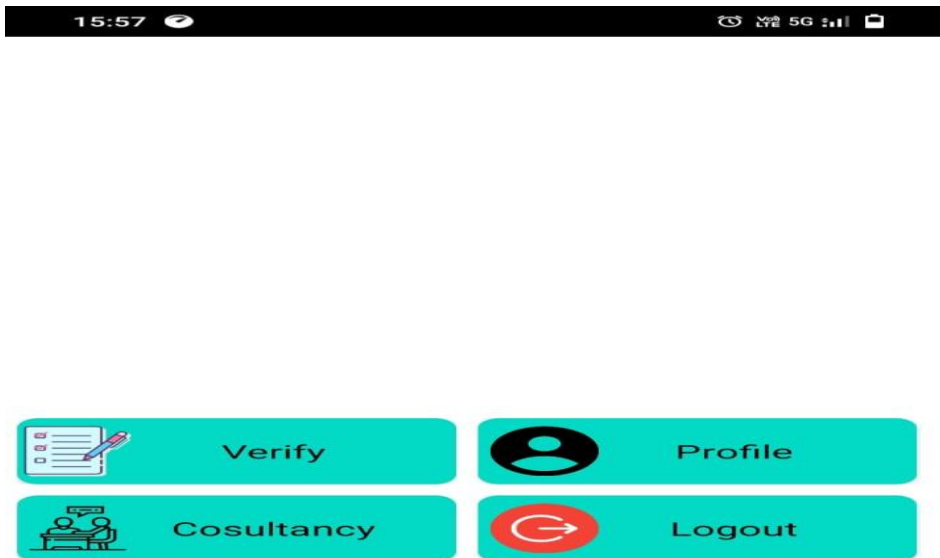


FIG 5.5: DOCTOR INTERFACE

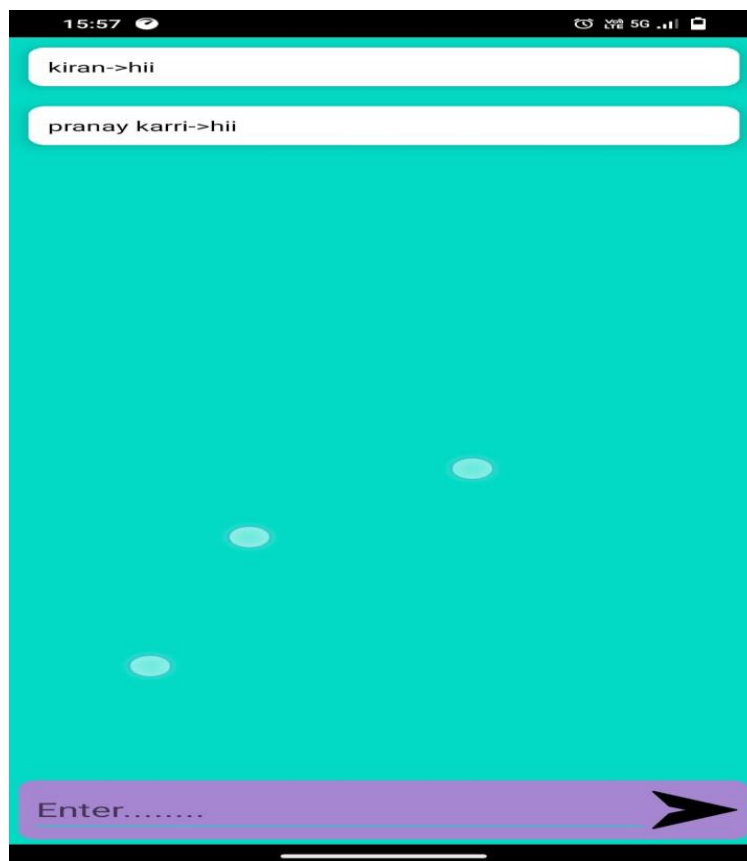


FIG 5.6: DOCTOR CONSULTATION

CHAPTER 6

CONCLUSION

6.1 CONCLUSION

Modern era has started. Individuals both, privileged or unprivileged, proficient or ignorant are now conscious about their health. It is sad that even though needy individuals are in distress they can't give a lot of consideration to their medical services routine in light of the fact because of their low pay. NGOs show incredible drive by giving free treatment to the people who cannot afford expensive medicines. Yet, in most of the cases, they get the treatment but not the costly prescribed medicines. So in the bigger picture, this incredible act of giving free treatment to the people becomes useless as they would need to buy the costly prescribed medicines by themselves. This paper gives a brief outline of the plan and improvement of our proposed online portal, which will be extremely successful and will bear incredible commitment to get the wellbeing administrations for these needy people. Because of this portal we hope that even the wastage of medication will be diminished.

In conclusion, the Android application for medicine donation and doctor consultation stands as a testament to the transformative potential of technology in advancing healthcare accessibility and efficacy. Its robust functionality, coupled with positive user engagement and tangible impact metrics, underscores its significance as a tool for bridging healthcare gaps and improving outcomes, particularly for underserved populations. While the journey thus far has been marked by challenges and iterative improvements, the project's ongoing evolution holds promise for further innovation and expansion. By fostering collaboration, embracing user feedback, and prioritizing sustainability, the application is poised to continue making meaningful contributions to healthcare delivery, ultimately empowering individuals to lead healthier lives and fostering a more equitable healthcare landscape.

6.2 FUTURE ENHANCEMENTS

This program or this online medication donation portal is possibly to end up a entire application in future where all the centres can be furnished in this portal. In the destiny, it can be extended to deal with the consumer through video consultation with doctors imparting medicine upload functionality. It may be very useful for the consumer as it is very important to follow the government guidelines and guidelines as per the contemporary COVID-19 situation. Video consultation facility with health practitioner is the first-class option for the destiny of the portal. For better referrals, we are connecting all scientific centres like counselling, drug donation, and blood donation as functions for you to be part of the portal in future. Hence, in future it becomes a commercial enterprise portal and a unique application in its nature and availability.

Looking ahead, several key enhancements can be envisaged to further elevate the Android application's capabilities and impact in facilitating medicine donation and doctor consultation. Firstly, expanding the scope of services to encompass additional healthcare verticals, such as mental health support, preventive care, and chronic disease management, would cater to a broader range of user needs and contribute to comprehensive healthcare provision. Moreover, integrating advanced features like AI-driven symptom assessment, telemedicine capabilities for remote diagnosis, and medication adherence tracking could enhance the application's efficacy in delivering personalized and proactive healthcare solutions. Additionally, implementing multilingual support and ensuring cultural sensitivity in content delivery would foster inclusivity and accessibility, particularly for diverse user demographics. Strengthening partnerships with healthcare organizations, pharmaceutical companies, and NGOs could facilitate broader reach, resource mobilization, and collaborative initiatives aimed at addressing systemic healthcare challenges. Furthermore, investing in data analytics capabilities to derive actionable insights from user interactions and health outcomes could inform targeted interventions and continuous improvement efforts. Embracing emerging technologies such as blockchain for transparent medicine supply chain management and decentralized health records could bolster security, trust, and interoperability within the ecosystem. Lastly, prioritizing user privacy, data security, and regulatory compliance in all aspects of application development and

deployment remains paramount to fostering trust and sustainability in the long term. By embracing these future enhancements, the Android application can evolve into a comprehensive healthcare platform that not only addresses immediate needs but also catalyses systemic change towards a more equitable and accessible healthcare landscape.

REFERENCES

- [1]Blood donation and life saver-blood donation app; M.R. Anish Hamlin; J. Albert Mayan. 24 July 2017.
- [2] Blood bank information system using Android application Publisher: IEEE Cite This PDF : Neetu Mittal; Karan Snotra; 14 May 2018.
- [3] mHealth: Blood donation application using android smartphone; Muhammad Fahim; Halil Ibrahim Cebe; Jawad Rasheed; Farzad Kiani; 18 August 2016.
- [4]Implement Android Application For Book Donation; Arushi Singh; Shilpi Sharma; 06 August 2020
- [5] Design Mobile Application for Blood Donation System; Muna M. Hummady; 14 February 2023

APPENDIX

A. SOURCE CODE

```
package com.example.medicinedonor.User

import android.annotation.SuppressLint
import android.content.Intent
import android.content.pm.PackageManager
import android.location.Location
import android.location.LocationListener
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.view.ViewGroup
import android.widget.ImageView
import android.widget.Toast
import androidx.cardview.widget.CardView
import androidx.core.app.ActivityCompat
import com.example.medicinedonor.Admin.GetComplaints
import com.example.medicinedonor.LoginPage
import com.example.medicinedonor.R
import com.google.android.gms.location.FusedLocationProviderClient
import com.google.android.gms.location.LocationServices
import java.util.jar.Manifest

class UserActivity : AppCompatActivity() {

    var lat=""

    var lon=""

    lateinit var fused: FusedLocationProviderClient
```

```

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_user)
    fused= LocationServices.getFusedLocationProviderClient(this)
    val image=findViewById<ImageView>(R.id.images)
    val image2=findViewById<ImageView>(R.id.images2)
    val images3=findViewById<ImageView>(R.id.images3)
    val images4=findViewById<ImageView>(R.id.images4)
    val images5=findViewById<ImageView>(R.id.images5)
    val images10=findViewById<ImageView>(R.id.images10)
    val image101=findViewById<ImageView>(R.id.image101)
    val donar=findViewById<CardView>(R.id.donar)
    val donee=findViewById<CardView>(R.id.donee)
    val history=findViewById<CardView>(R.id.history)
    val profile=findViewById<CardView>(R.id.profile)
    val consult=findViewById<CardView>(R.id.conus)
    val historyc=findViewById<CardView>(R.id.historyc)
    val logout=findViewById<CardView>(R.id.logout)

    if(ActivityCompat.checkSelfPermission(this,android.Manifest.permission.ACCE
    SS_COARSE_LOCATION)!=PackageManager.PERMISSION_GRANTED
    &&ActivityCompat.checkSelfPermission(this,android.Manifest.permission.ACC
    ESS_FINE_LOCATION)!=PackageManager.PERMISSION_GRANTED){
        ActivityCompat.requestPermissions(this,
        arrayOf(android.Manifest.permission.ACCESS_FINE_LOCATION,
            android.Manifest.permission.ACCESS_COARSE_LOCATION),100)
    }else{
        charna()
    }
}

```

```

        historyc.setOnClickListener {
            animation(image101,it)
        }
        consult.setOnClickListener {
            animation(images10,it)
        }
        donar.setOnClickListener{
            animation(image,donar)
        }
        donee.setOnClickListener{
            animation(image2,donee)
        }
        history.setOnClickListener{
            animation(images3,history)
        }
        profile.setOnClickListener {
            animation(images4,profile)
        }
        logout.setOnClickListener{
            animation(images5,logout)
        }
    }

    @SuppressWarnings("MissingPermission")
    private fun charna() {
        fused.lastLocation.addOnSuccessListener {

```

```

        lat=it.latitude.toString()

        lon=it.longitude.toString()
    }
}

fun animation(image:ImageView,btn:View) {

    image.alpha=0f
    image.animate().setDuration(500).alpha(1f).withEndAction {
        overridePendingTransition(androidx.appcompat.R.anim.abc_fade_out,
            androidx.appcompat.R.anim.abc_fade_in)
    }
    if(btn.id==R.id.donee){
        val int=Intent(this,DoneeActivity::class.java)
        int.putExtra("lat",lat)
        int.putExtra("lon",lon)
        startActivity(int)
    }else if(btn.id==R.id.logout){
        val share=getSharedPreferences("data", MODE_PRIVATE)
        share.edit().apply {
            clear()
            apply()
        }
        startActivity(Intent(this,LoginPage::class.java ))
        finish()
    }else if(btn.id==R.id.history){
        startActivity(Intent(this,History::class.java))
    }else if(btn.id==R.id.profile){
        startActivity(Intent(this,ProfilePage::class.java))
    }
}

```

```

        }else if(btn.id==R.id.donar){
val k=Intent(this,GetComplaints::class.java)
        k.putExtra("data","data")
            k.putExtra("data2","data2")
            startActivity(k)
        }else if(btn.id==R.id.conus){
val int=Intent(this,RecyclerView::class.java)
            int.putExtra("just","just")
            int.putExtra("data2","data2")
            startActivity(int)
        }else if(btn.id==R.id.historyc){
            val jj=Intent(this,HistoryConsultancy::class.java)
            jj.putExtra("data","user")
            startActivity(jj)
        }
    }
}

    override fun onRequestPermissionsResult(requestCode: Int, permissions:
Array<out String>, grantResults: IntArray) {
        if(requestCode==100){
            charna()
        }

        super.onRequestPermissionsResult(requestCode, permissions,
grantResults)
    }

}

package com.example.medicinedonor.User

```

```

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.EditText
import android.widget.ImageView
import android.widget.Toast
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.example.medicinedonor.Adapter.ChatAdapter
import com.example.medicinedonor.Modle.Dummy
import com.example.medicinedonor.R
import com.example.medicinedonor.Response.CommonResponse
import com.example.medicinedonor.Retrofit.Retrofit
import com.example.medicinedonor.toast
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.Dispatchers.IO
import kotlinx.coroutines.launch
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
import java.lang.StringBuilder

class ChatArea : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_caht_area)
        val int=intent.getStringExtra("data")
        val cycle=findViewById<RecyclerView>(R.id.recyclerView)
        val edite=findViewById<EditText>(R.id.edite)
    }
}

```

```

        val clclick=findViewById<ImageView>(R.id.clclick)

        val array=ArrayList<Dummy>()
val id=intent.getIntExtra("id",0)

        val hh=intent.getStringExtra("doctor")

        cycle.layoutManager=LinearLayoutManager(this)

        if(int!!.isEmpty()){

            val split=int.split("@")

            for (item in split){

                if(item.isEmpty()) {

                    array.add(Dummy(item.trim()))

                }

            }

            val adapter= ChatAdapter(this,array)

            cycle.adapter=adapter

        }

        val f=getSharedPreferences("data", MODE_PRIVATE).getString("name","")

        var my=""

        if (hh!=null){

            my=f!!.split(",")[0]

        }else{

            my=f!!

        }

        clclick.setOnClickListener {

            val edit="@"+my+"->"+edite.text.toString()

            if(edite.text.toString().trim().isEmpty()){

                val split=edit.split("@")

                for (item in split){

                    if(item.isEmpty()) {


```

```

        array.add(Dummy(item))
    }}

    val adapter= ChatAdapter(this,array)

    cycle.adapter=adapter

    val ray=StringBuilder()

    for (item in array){

        ray.append(item.text.trim()+"@")

    }

    CoroutineScope(IO).launch {

        Retrofit.instance.updatingvideo("chat",id,ray.toString()).enqueue(object
:Callback<CommonResponse>{

            override fun onResponse(call: Call<CommonResponse>, response:
Response<CommonResponse>) {

                it.toast(response.body()!!.message)

            }

            override fun onFailure(call: Call<CommonResponse>, t: Throwable) {

                Toast.makeText(this@ChatArea, "${t.message}",
Toast.LENGTH_SHORT).show()

            }

        })

    }

    }else{

        it.toast("Empty")

    }

    edite.setText("")

}

}

package com.example.medicinedonor.Admin

```

```

import android.annotation.SuppressLint
import android.app.ProgressDialog
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import com.example.medicinedonor.R
import com.example.medicinedonor.Response.CommonResponse
import com.example.medicinedonor.Retrofit.Retrofit
import com.example.medicinedonor.logfail
import com.example.medicinedonor.logsucces
import com.example.medicinedonor.toast
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.Dispatchers.IO
import kotlinx.coroutines.launch
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response

class AddingDoctor : AppCompatActivity() {
    @SuppressWarnings("SetTextI18n")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_adding_doctor)
        val name=findViewById<EditText>(R.id.dname)
        val mobile=findViewById<EditText>(R.id.dmobile)
        val graduation=findViewById<EditText>(R.id.graduation)
        val mail=findViewById<EditText>(R.id.mailids)
    }
}

```

```

        val passwordz=findViewById<EditText>(R.id.passwordz)
val data=intent.getStringExtra("data")
        var myview="Adding"
        var id=1
        if(data!=null){
            myview="Updating"
            getSharedPreferences("data", MODE_PRIVATE).apply {
                val split=getString("name","d")!!.split(",")
                name.setText(split[0])
                mobile.setText(getString("mobile",""))
                graduation.setText(split[1])
                mail.setText(getString("mail",""))
                passwordz.setText(getString("password",""))
                id= getString("id","")!!.toInt()
            }
        }
        val btn=findViewById<Button>(R.id.create)
        btn.setOnClickListener {
            val name1=name.text.toString()
            val mobile1=mobile.text.toString()
            val graduation1=graduation.text.toString()
            val mail1=mail.text.toString()
            val password1=passwordz.text.toString()
            if(name1.isEmpty()||
                mobile1.isEmpty()||
                graduation1.isEmpty()||
                mail1.isEmpty()||

```

```

        password1.isEmpty()){

        btn.toast("Empty")
    }else {

        val p = ProgressDialog(this).apply {

            setCancelable(false)

            setTitle("$myview....")

            setMessage("$myview ${name.text} Doctor")

            show()

        }

        CoroutineScope(IO).launch {

            if(data!=null){

                Retrofit.instance.updatedoctor("update",id,name1 + "," +
graduation1, mobile1, mail1, password1, "doctor").enqueue(object
:Callback<CommonResponse>{

                    override fun onResponse(call: Call<CommonResponse>,
response: Response<CommonResponse>) {

                        val k=response.body()!!

                        if(k.message=="updated") {

                            getSharedPreferences("data",
MODE_PRIVATE).edit().apply {

                                putString("name", name1 + "," + graduation1)

                                putString("mobile", mobile1)

                                putString("mail", mail1)

                                putString("type", "doctor")

                                putString("id",id.toString())

                                putString("password", password1)

                                apply()

                                btn.toast(k.message)

                                p.dismiss()

```

```

        }

    }

}

        override fun onFailure(call: Call<CommonResponse>, t:
Throwable) {
    p.dismiss()

        }

    })

} else {

        Retrofit.instance.addDocotor(name1 + "," + graduation1, mobile1,
mail1, password1, "doctor").enqueue(object : Callback<CommonResponse> {

            override fun onResponse(call: Call<CommonResponse>,
response: Response<CommonResponse>) {

                response.body()?.let { logSuccess(it.message) }

                finish()

                p.dismiss()

            }

            override fun onFailure(call: Call<CommonResponse>, t:
Throwable) {

                logFail("${t.message}")

                p.dismiss()

            }

        })

    }

}

}

}

```

```

    }
}
}
package com.example.medicinedonor.Admin
import android.app.AlertDialog
import android.app.Dialog
import android.app.ProgressDialog
import android.content.DialogInterface
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.ImageView
import android.widget.Toast
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.bumptech.glide.Glide
import com.example.medicinedonor.Adapter.AdapterForUserStatus
import com.example.medicinedonor.Adapter.Click
import com.example.medicinedonor.Modle.DoneeM
import com.example.medicinedonor.R
import com.example.medicinedonor.Response.CommonResponse
import com.example.medicinedonor.Response.DoneeStatus
import com.example.medicinedonor.Retrofit.Retrofit
import com.example.medicinedonor.User.DonorActivity
import com.example.medicinedonor.logfail
import com.example.medicinedonor.logsucces
import kotlinx.coroutines.CoroutineScope

```

```

import kotlinx.coroutines.Dispatchers.IO

import kotlinx.coroutines.launch

import retrofit2.Call

import retrofit2.Callback

import retrofit2.Response

class GetComplaints : AppCompatActivity(), Click {

    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)

        setContentView(R.layout.activity_assign_docotor)

        val cycle = findViewById<RecyclerView>(R.id.cycle2)

        cycle.layoutManager = LinearLayoutManager(this)

        val data = intent.getStringExtra("mydata")

        val the = intent.getStringExtra("data")

        val p = ProgressDialog(this).apply {

            setTitle("Loading.....")

            setMessage("Wait.....")

            setCancelable(false)

            show()

        }

        CoroutineScope(IO).launch {

            if (the != null) {

                Retrofit.instance.gettingverified("Verified")

                .enqueue(object : Callback<DoneeStatus> {

                    override fun onResponse(

                        call: Call<DoneeStatus>,

                        response: Response<DoneeStatus>

                    ) {


```

```

        val e = response.body()!!

        val adppter =
            AdapterForUserStatus(this@GetComplaints, e.data,
this@GetComplaints)

        cycle.adapter = adppter

        logsucces(e.message)

        p.dismiss()
    }

    override fun onFailure(call: Call<DoneeStatus>, t: Throwable) {
        p.dismiss()
    }
})
} else {
    if (data != null) {
        Retrofit.instance.verify("Process", data)

        .enqueue(object : Callback<DoneeStatus> {
            override fun onResponse(
                call: Call<DoneeStatus>,
                response: Response<DoneeStatus>
            ) {
                val e = response.body()!!

                val adppter = AdapterForUserStatus(
                    this@GetComplaints,
                    e.data,
                    this@GetComplaints
                )

                cycle.adapter = adppter
            }
        })
    }
}

```

```

        logsucces(e.message)
        p.dismiss()
    }
    override fun onFailure(call: Call<DoneeStatus>, t: Throwable) {
        logfail("${t.message}")
        p.dismiss()
    }
})

package com.example.medicinedonor.Admin
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Toast
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.example.medicinedonor.Adapter.AdapterForDoctors
import com.example.medicinedonor.Adapter.Userclick
import com.example.medicinedonor.Modle.User
import com.example.medicinedonor.R
import com.example.medicinedonor.Response.CommonResponse
import com.example.medicinedonor.Response.UsersResponse
import com.example.medicinedonor.Retrofit.Retrofit
import com.example.medicinedonor.logfail
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.Dispatchers.IO
import kotlinx.coroutines.launch
import retrofit2.Call
import retrofit2.Callback

```

```

import retrofit2.Response

class DoctorsList : AppCompatActivity(),Userclick {

    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)

        setContentView(R.layout.activity_doctors_list)

        val cycle=findViewById<RecyclerView>(R.id.myview)

        cycle.layoutManager=LinearLayoutManager(this)

        CoroutineScope(IO).launch {

            Retrofit.instance.gettinglist("doctor").enqueue(object
:Callback<UsersResponse>{

                override fun onResponse(call: Call<UsersResponse>, response:
Response<UsersResponse>) {

                    val j=response.body()!!

                    cycle.adapter=AdapterForDoctors(this@DoctorsList,j.data,this@DoctorsList)

                }

                override fun onFailure(call: Call<UsersResponse>, t: Throwable) {

                    logfail("${t.message}")

                }

            })

        }

    }

    override fun click(f: User) {

        val int=intent.getIntExtra("data",1)

        CoroutineScope(IO).launch {

            Retrofit.instance.assigndoctor("doctorupdate",int,f.name).enqueue(object
:Callback<CommonResponse>{

                override fun onResponse(call: Call<CommonResponse>, response:
Response<CommonResponse>) {

                    val r=response.body()!!

```

```

        Toast.makeText(this@DoctorsList, r.message,
        Toast.LENGTH_SHORT).show()

    }

    override fun onFailure(call: Call<CommonResponse>, t: Throwable) {
        Toast.makeText(this@DoctorsList, "${t.message}",
        Toast.LENGTH_SHORT).show() } })

    }

}

```

```

package com.example.medicinedonor.Doctor

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.widget.ImageView
import android.widget.TextView
import android.widget.Toast
import androidx.cardview.widget.CardView
import com.example.medicinedonor.Admin.AddingDoctor
import com.example.medicinedonor.Admin.GetComplaints
import com.example.medicinedonor.LoginPage
import com.example.medicinedonor.R
import com.example.medicinedonor.User.HistoryConsultancy
import com.example.medicinedonor.User.RecyclerView
import com.example.medicinedonor.User.UserActivity

class DoctorActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {

```

```

super.onCreate(savedInstanceState)

setContentView(R.layout.activity_doctor)

val verify=findViewById<CardView>(R.id.verify)
val logout=findViewById<CardView>(R.id.logout)
val profile=findViewById<CardView>(R.id.profiled)
val image=findViewById<ImageView>(R.id.image10)
val image2=findViewById<ImageView>(R.id.logout)
val image11=findViewById<ImageView>(R.id.image11)
val image15=findViewById<ImageView>(R.id.image15)

findViewById<CardView>(R.id.card1).setOnClickListener{
    animation(image15,it)
}

profile.setOnClickListener {
    animation(image11,profile)
}

verify.setOnClickListener {
    animation(image,verify)
}

logout.setOnClickListener {
    animation(image2,logout)
}

```

```

package com.example.medicinedonor.Doctor

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.example.medicinedonor.R
import com.example.medicinedonor.Response.ConsultResponse
import com.example.medicinedonor.Retrofit.Retrofit
import com.example.medicinedonor.toast
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.Dispatchers.IO
import kotlinx.coroutines.launch
import retrofit2.Callback
import retrofit2.Response

class ViewUser : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_view_user)
        val cycle=findViewById<RecyclerView>(R.id.cycle3)
        cycle.layoutManager=LinearLayoutManager(this)

    }
}

package com.example.medicinedonor.Ngo

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle

```

```

import android.view.View

import android.widget.ImageView

import androidx.cardview.widget.CardView

import com.example.medicinedonor.LoginPage

import com.example.medicinedonor.R

import com.example.medicinedonor.User.JumpedCycle

class NgoMainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)

        setContentView(R.layout.activity_ngo_main)

        val assigned=findViewById<CardView>(R.id.assigned)
        val profiles=findViewById<CardView>(R.id.profiles)
        val logout3=findViewById<CardView>(R.id.logout3)
        val image12=findViewById<ImageView>(R.id.image13)
        val image14=findViewById<ImageView>(R.id.image14)
        val image15=findViewById<ImageView>(R.id.myloaogut)

        assigned.setOnClickListener {

            animation(image12,it)

        }

        profiles.setOnClickListener {

            animation(image14,it)

        }

        logout3.setOnClickListener {

            animation(image15,it)

        }

    }

    private fun animation(img:ImageView,view: View) {

```

```

img.alpha=0f

img.animate().alpha(1f).setDuration(500).withEndAction {

overridePendingTransition(androidx.appcompat.R.anim.abc_fade_in,androidx.app
compat.R.anim.abc_fade_out)

    if(view.id==R.id.assigned){

startActivity(Intent(this,JumpedCycle::class.java))

    }else if(view.id==R.id.logout3){

        sharedPreferences("data", MODE_PRIVATE).edit().clear().apply()

        startActivity(Intent(this,LoginPage::class.java))

        finishAffinity()

    }

}

}

}

```

B.SCREENSHOTS

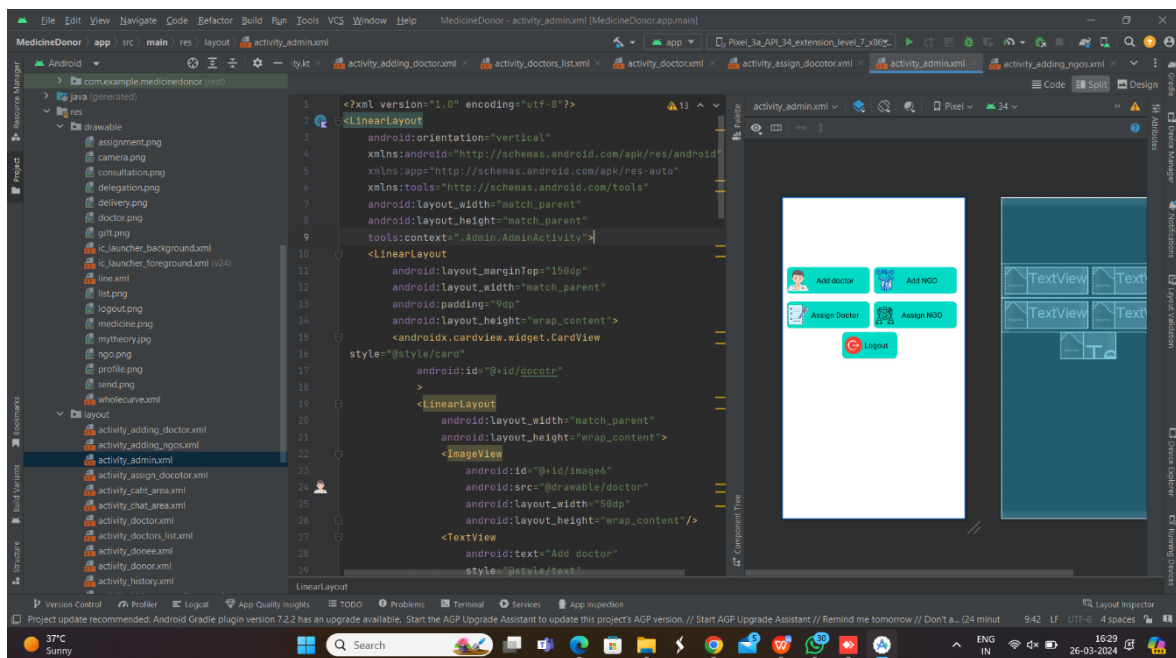


FIG B.1: LAYOUT OF ADMIN ACTIVITY

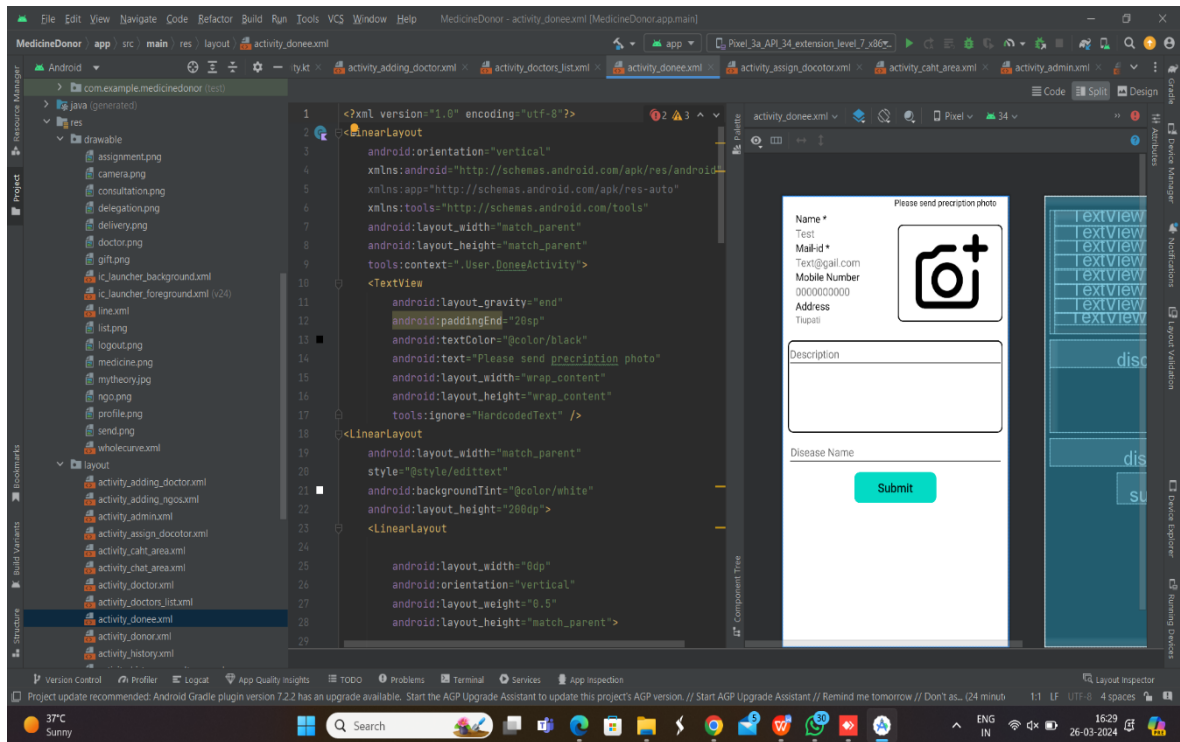


FIG B.2: LAYOUT OF DONEE ACTIVITY

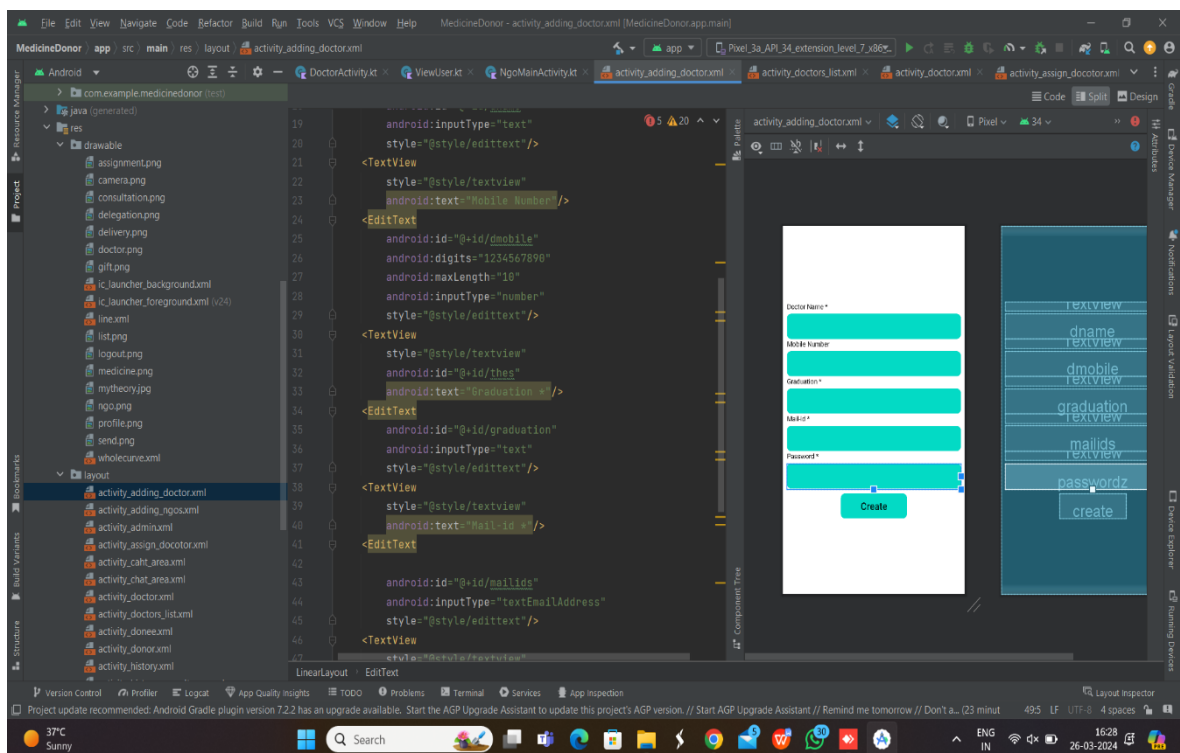


FIG B.3: LAYOUT OF ADDING DOCTORS

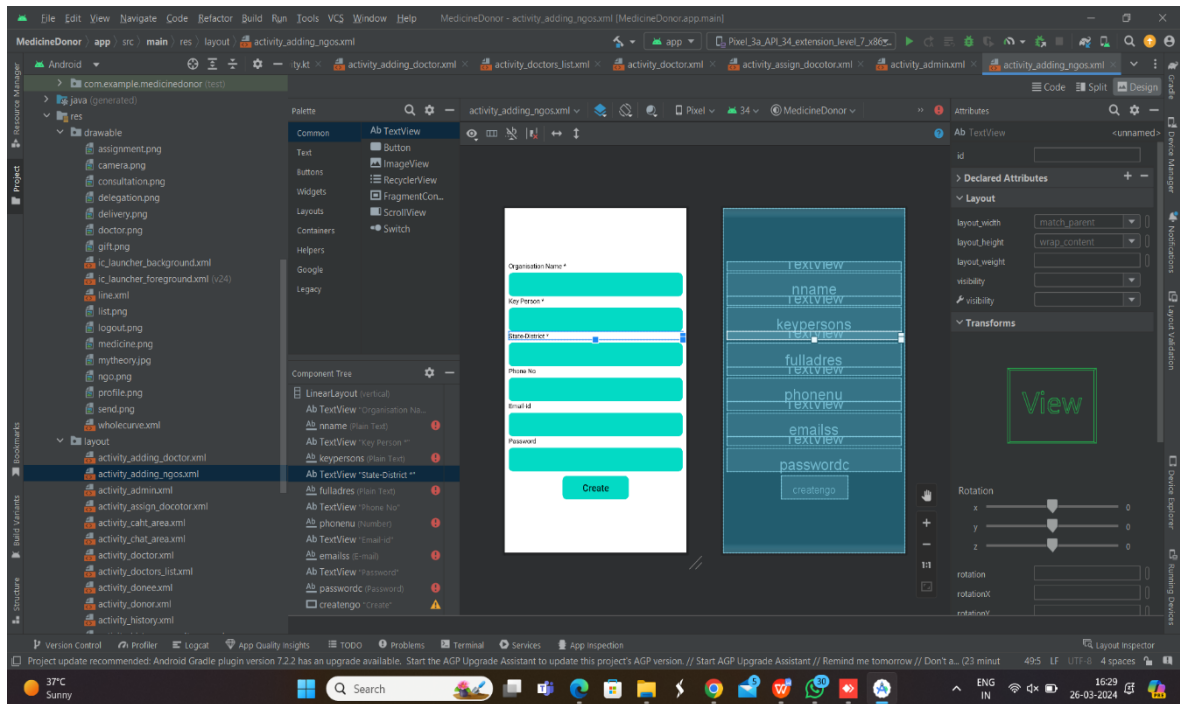


FIG B.4: LAYOUT OF ADDING NGOS

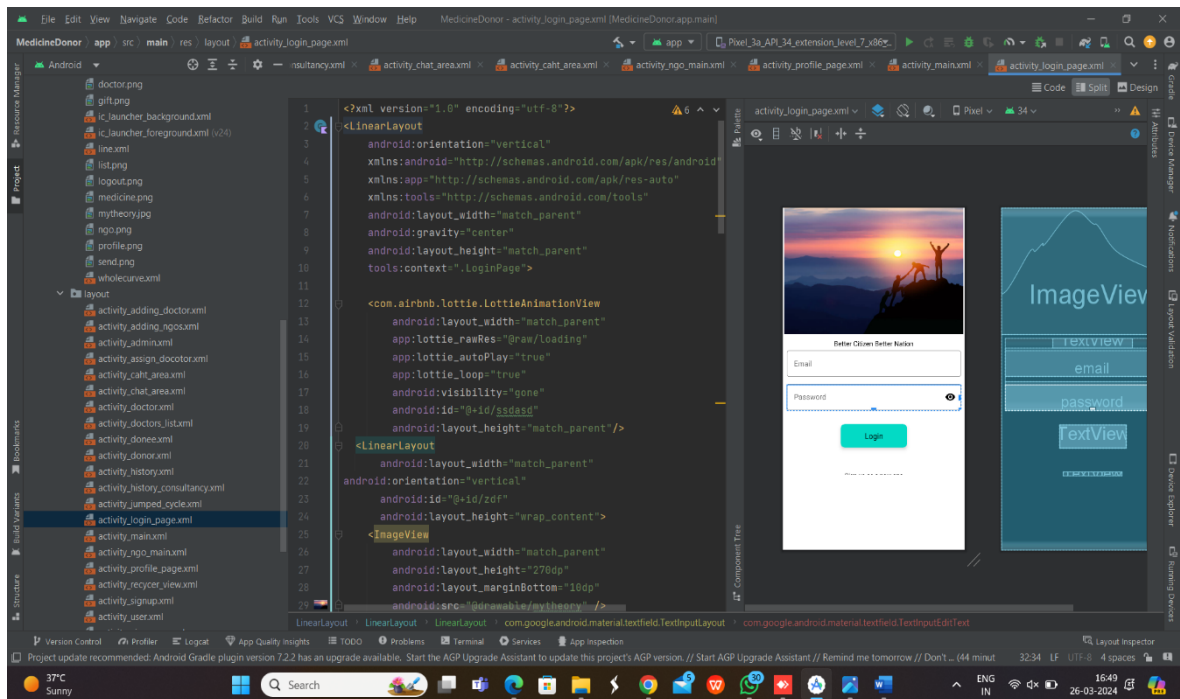


FIG B.5: LOGIN PAGE LAYOUT

C.RESEARCH PAPER

ANDROID APPLICATION FOR MEDICINE DONATION AND DOCTOR CONSULTATION

Gunnam Smaran

Department: IT

Sathyabama Institute of
Science and Technology

Gunnamsmaran@gmail.com

K.D.N.V.S.Pranay

Department: IT

Sathyabama Institute of
Science and Technology

Pranaykarri412@gmail.com

A.Sathiyaraj M . E., Ph. D

Assistant Professor

Sathyabama Institute of
Science and Technology

sathiyaraj.a.it@sathyabama.ac.in

Abstract: To meet the developing call for healthcare facilities and services, customers can harness the energy of the Internet technology and its core networks, via which human beings can assist every different with a single click on their phones. The medicinal drug donation device proposed here pursuits to offer a platform to donate unused drugs or pills to the ones in want. Users sign up in this device via supplying required data. Upon registration, customers can donate drug treatments to an NGO by using offering the precise quantity of scientific treatments acquired. The agency reserves the right to censor users for imparting unlawful or expired pills. A memorial organization will offer donations and medicinal drugs. Users can observe tablets to donate or eat. This drug donation initiative is making an attempt to create an app to get unused capsules so that they may be used by a person in want. Donate unused medicinal drug to an NGO. A complete-fledged non-governmental organization is needed. "User can donate drug treatments. Many those who cannot increase cash to shop for drug treatments, remedies and medicines for each disorder can arrange treatment with used and unused drug

treatments via this app. The app also has health practitioner's advice and prescriptions for patients.

Keywords: Demand, Medicine, accurate, utilization.

INTRODUCTION

Medical Donation System is an online platform designed to meet the growing call for improving healthcare facilities and offerings. By harnessing the strength of Internet technology and its extensive network, the gadget lets in people to help each other with a easy click on from their mobile devices. The principal objective of this company is to offer a handy platform for dispensing drugs which includes unused medicines to the ones in need. Users can easily check in at the web site by means of submitting the specified details.

Once registered, they could deliver drug treatments to NGO's with the aid of imparting correct details of the medicines. In order to make sure the safety and effectiveness of donated capsules, the business enterprise reserves the proper to dam users who try to donate inappropriate or expired capsules. The gadget continues a entire document of donated and to be had drugs, allowing customers to submit

requests for donation or availability of medicine. The goal of this drug donation program is to create an app that enables the collection of unused tablets that can be used by those in need. Through the app, users can donate their unused medicines to NGO's so as to offer help to those who can not have enough money medicines.

This initiative plays a crucial position in ensuring that many terrible humans get the necessary remedy and drugs to cope with their respective illnesses. Additionally, the app has additional features along with doctor guidelines and patient referrals, enhancing the overall healthcare experience. By harnessing the capabilities of this platform, people can actively make contributions to improving get entry to to fitness care and fostering a feel of team spirit amongst groups. Together, we are able to make a significant impact on the lives of these less fortunate, even as underutilized medicines help find a motive and simplicity suffering.

OBJECTIVE

The important objective of this app is to facilitate the exchange of facts among the donor and the donor that allows you to switch the medication from one to some other in a simple way.

RELATED WORK

[1] Application for blood donation and saving lives / MR. Anish Hamlin; Printed via John Albert Mayan.

"Blood" is one of the most critical wishes of our existence. The variety of blood donors could be very less compared to other countries. In our assignment we propose a brand new and efficient way to overcome this hassle. For instance, on the touch of a button, the donor will be requested to go into info inclusive of name, telephone range, age, weight, and date of delivery, blood organization, address and many others. When blood is needed in emergencies,

GPS can be used to check whether a blood donor is nearby or now not. Once the app person enters their required blood institution, it will routinely display close by donors and ship them an alert message. If the first donor isn't to be had, it's going to robotically search for the following donor in line. If the donor accepts the request, a one-time password (OTP) might be despatched to them for verification. List of blood donors on your metropolis/place. Once the donor donates blood, the donor facts is automatically deleted for the subsequent 3 months.

[2] Blood Bank Information System Using Android Editor App; / Nitu Mittal; Karan Snotra.

Availability of blood in emergencies may be very critical for all residing matters. There are many digital blood donation centres for effective communication between them and medical centres. Any on-line blood donation centre does no longer establish an immediately connection among the beneficiary and them. This is the real drawback of the present day framework. Existing structures are tiring; They require more hard work and price more. This paper gives the connection among the present blood banking device and an advanced framework for enhancing efficiency. New thoughts can growth the performance of existing blood banks and help them pass from easy desktop frames to transportable frames. The proposed work analyses the components of the improved framework from numerous angles such as stored facts, information for future packages, and sort of blood company's people donate and receive.

[3] Mobile Health: Blood Donation App on Android Smartphone/Muhammad Fahim; Khalil Ibrahim Sebe; Jawad Rashid; Farsad Giani

There are new frontiers in fitness care shipping the usage of mobile devices and communicate technology. In healthcare, blood donation is a

complicated technique and it takes time to find a donor whose blood institution is well matched with the affected person's blood institution. We are developing an Android primarily based blood donation app as a mHealth option to join the seeker and the donor every time, everywhere. The motive of this application is to offer records about the asked blood and the range of donors available at those locations. It facilitates message the applicant via the network of volunteer blood donors created via our app and on the same time updates the applicant who desires he requested to donate blood. He created us to fee our usefulness a profile of the applicant-donor and analysed what would facilitate well timed get admission to information and speedy reaction in emergency situations.

[4]Implementation of Android App for Book Donation / Arushi Singh; Shilpi Sharma

Books are a supply of know-how and get admission to them is something that is regularly overlooked. However, the fact is that many human beings have books at home that they no longer need, and which are useful to someone who urgently desires them. Android is one of the maximum popular cellular software operating structures inside the international because it offers assist for ease of use and improvement for each clients and builders.. This article proposes an easy mobile app to attach folks People who are inquisitive about presenting their books to individuals who want them. The app is referred to as "Bridge" which connects the needy and philanthropists to help them get the books they want. folks who no longer use them.

[5] Development of mobile application for blood donation organization / Muna M. Hammadi

Computers and mobile phones offer many advantages in our day by day lives and have grown to be common in society. Internet conversation (thru

web sites and apps) has come to be essential in the wake About Corona, pandemic and associated troubles. Difficulties in locating the right blood luggage value many lives. Blood donation is critical for thalassemia patients, most cancers sufferers, coincidence sufferers and surgical control. To donate blood, you need to perform a little studies and visit a transfusion facility. In instances of time constraints, deciding on an amazing donor may be tough. Since peculiar blood types are normally no longer available in all blood banks, recipients regularly battle to locate With the donor's personal blood. This is evident from insufficient regulation of blood banks, exclusion of uncommon blood kinds, lack of knowledge and belief and designation of a particular blood group to clear up the hassle of blood shortage boundaries. The goal of this assignment is to expand and put into effect a cell software. It is proposed to apply a blood donation software linked to a mandatory database that collects and organizes information from all blood donation activities and blood banks. All required blood donation packages are managed and controlled through the app. In this challenge, the front give up of the program is constructed the usage of JavaScript (the use of React Native as the base for JS) and the again end is built. usage of Firefox as the database.

EXISTING SYSTEM

In present systems, maximum concentrates are generated inside the use of blood donation and meals donation, but drug donation is in particular considered on this use.

Disadvantages of Existing System

- Apps are in particular developed for meals and blood donation.

PROPOSED SYSTEM

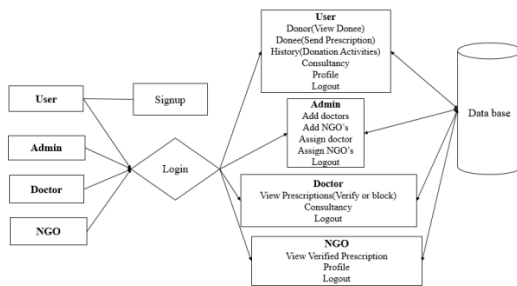
In this proposed system, the user can donate the medicine, then the medication has to be checked via

a doctor, then it has to be surpassed over to an NGO, and then the medication needs to be disbursed.

Advantages of Proposed System

- Easy to use.
- User Friendly application.

BLOCK DIAGRAM



MODULE AND THEIR DESCRIPTION

User: User sign in and login, after login the consumer can see the medicines required by the people and person can get help via importing a prescription and writing the disease associated info, then he can donate their records and see this there too. . Received medicine.

Admin: Admin can upload doctors and NGO's after which view doctors and NGO's and assign duties to them.

NGO: NGO sees that his obligations are established by means of the health practitioner, who can then go to the consumer's vicinity and collect or supply the medicine.

Physician: A medical doctor can view prescriptions assigned with the aid of an administrator, then view the prescription, test and consider the prescription, or block the user.

SYSTEM REQUIREMENTS

Hardware Requirements

Processor: I3/Intel Processor.

RAM: 8GB (min).

Hard Disk: 1 TB.

Key Board: Standard Windows Keyboard.

Mouse: Two or Three Button Mouse.

Software Requirements

Operating System: Windows 7.

Programming: Java

Server-side Script: PHP

IDE: Android Studio.

SDK: Android

Libraries Used: Volley, Material design.

RESULT AND DISCUSSION



Fig. 1.USER INTERFACE

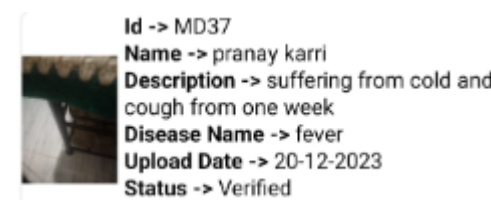


Fig. 2.USER HISTORY



Fig. 3.ADMIN INTERFACE

 The form for adding a doctor has six input fields, each with a label and an asterisk indicating it is required: 'Organisation Name *', 'Key Person *', 'State-District *', 'Phone No', 'Email-id', and 'Password'. Each field is a teal-colored rectangle. Below the fields is a teal 'Create' button. There is also a small circular icon to the left of the 'Create' button.

Fig. 4.ADMIN ADDS DOCTORS

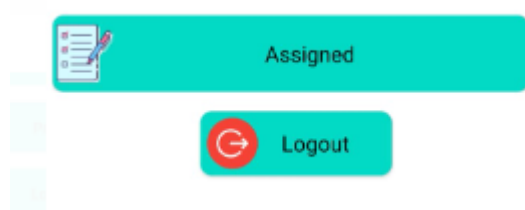


Fig. 5.NGO INTERFACE

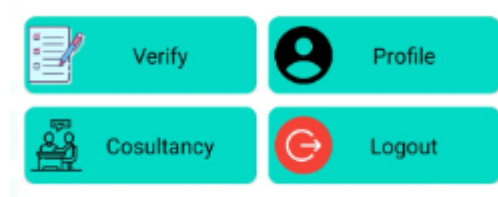


Fig. 6.DOCTOR INTERFACE

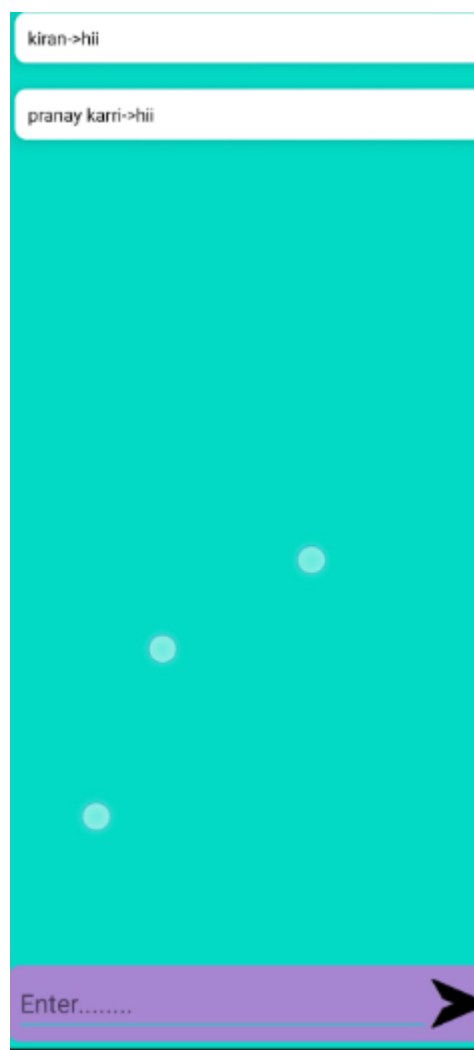


Fig. 7 .DOCTOR CONSULTATION

The application it might be like a beneficiary donating unused medicinal drug to an NGO. That NGO can help needy people. This application also includes the doctor suggestions and prescriptions for the patients.

CONCLUSION

The current age has started. Individuals, privileged and underprivileged, professional and unskilled, are actually conscious about their fitness. It is unlucky that humans dwelling in poverty are unable to pay lots interest to their everyday scientific offerings due to low salaries. NGO's display super motivation in supplying loose treatment to folks who cannot come

up with the money for costly drug treatments. However, in maximum cases they receive remedy but now not steeply-priced capsules. So all in all, this excellent paintings of imparting unfastened remedy to people once they have to buy costly prescribed drugs themselves is wasted. This record presents a brief summary of the making plans and development of our proposed on-line portal, with a view to be tremendously a success and could take excellent dedication to get welfare control to those needy populations. Thanks to this website, we hope that the wastage of drugs may also decrease.

FUTURE ENHANCEMENT

This program or this online medication donation portal is possibly to end up a entire application in future where all the centers can be furnished in this portal. In the destiny, it can be extended to deal with the consumer through video consultation with doctors imparting medicine upload functionality. It may be very useful for the consumer as it is very important to follow the government guidelines and guidelines as per the contemporary COVID-19 situation. Video consultation facility with health practitioner is the first-class option for the destiny of the portal. For better referrals, we are connecting all scientific centers like counselling, drug donation, and blood donation as functions for you to be part of the portal in future. Hence, in future it becomes a commercial enterprise portal and a unique application in its nature and availability.

REFERENCES

- [1] Blood donation and life saver-blood donation app; M.R. Anish Hamlin; J. Albert Mayan. 24 July 2017.
- [2] Blood bank information system using Android application Publisher: IEEE Cite This PDF: Neetu Mittal; Karan Snotra; 14 May 2018.

[3] mHealth: Blood donation application using android smartphone; Muhammad Fahim; Halil Ibrahim Cebe; Jawad Rasheed; Farzad Kiani; 18 August 2016.

[4]Implement Android Application for Book Donation; Arushi Singh; Shilpi Sharma; 06 August 2020

[5] Design Mobile Application for Blood Donation System; Muna M. Hummady; 14 February 2023

ANDROID APPLICATION FOR I DOCTOR CONSULTATION

ORIGINALITY REPORT

6%

SIMILARITY INDEX

3%

INTERNET SOURCES

PRIMARY SOURCES

1

Muna M. Hummady. "D
Application for Blood D
International Conferenc
Applications (ICCA), 202
Publication

2

www.semanticscholar.org
Internet Source

3

Submitted to Sreenidhi
Student Paper

4

ijsrcseit.com
Internet Source

Exclude quotes ☐ On

Exclude bibliography ☐ On