VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI - 590 018, KARNATAKA.

MOBILE APPLICATION DEVELOPMENT

MINI PROJECT REPORT

ON

"ANDROID BLOOD BANK APP"

Submitted in the partial fulfillment of requirements

FOR

MOBILE APPLICATION DEVELOPMENT LABORATORY (18CSMP68)

Submitted by

NIKIL B S	**4BD20CS062**
SYED FARHAN	**4BD20CS104**

PROJECT GUIDES:

Prof. RADHIKA PATIL M. Tech.,
Asst. Professor, Dept. of CS&E

Dr. SANTOSH K C Ph.D.
Associate Professor, Dept. of CS&E

2022-23

Department of Computer Science and Engineering.

Bapuji Institute of Engineering & Technology

Davangere- 577004

Bapuji Institute of Engineering and Technology Davangere-577004

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that **NIKIL B S** AND **SYED FARHAN** bearing USN **4BD20CS062** AND **4BD20CS104** respectively of **Computer Science and Engineering** department have satisfactorily submitted the mini project report entitled **"ANDROID BLOOD BANK APP"**. The report of the project has been approved as it satisfies the academic requirements in respect of project work prescribed for the academic year 2022-23.

Project Guide 1

Dr. SANTOSH K C Ph.D.
Associate Professor
Department of CS&E,
B.I.E.T, Davangere

Project Guide 2

Prof. RADHIKA PATIL M.Tech.,
Asst. Professor
Department of CS&E,
B.I.E.T, Davangere

Head of Department

Dr. Nirmala C R Ph.D.
Prof. & Head,
Department of CS&E,
B.I.E.T, Davangere

Date:

Place: Davangere

Signature of Examiners:

(1) _____

(2) _____

ACKNOWLEDGEMENT

Salutations to our beloved and highly esteemed institute, **"BAPUJI INSTITUTE OF ENGINEERING AND TECHNOLOGY"** for having well qualified staff and labs furnished with necessary equipment's.

We express our sincere thanks to our guide **Dr. SANTOSH K C & Prof. RADHIKA PATIL** for giving us constant encouragement, support and valuable guidance throughout the course of the project without whose stable guidance this project would not have been achieved.

We express whole hearted gratitude to **Dr. NIRMALA C R** who is our respectable HOD of Computer Science & Engineering Department.We wish to acknowledge her help who made our task easy by providing with her valuable help and encouragement.

We also express our whole hearted gratitude to our principal, **Dr. H B ARAVIND,** for his moral support and encouragement.

We would like to extend my gratitude to all staff of **COMPUTER SCIENCE AND ENGINEERING DEPARTMENT** for the help and support rendered to me. We have benefited a lotfrom the feedback, suggestions given by them.

We would like to extend our gratitude to all my family members and friends especially for their advice and moral support.

NIKIL B S (4BD20CS062)
SYED FARHAN (4BD20CS104)

Vision and Mission of the Institute

Vision

"To be a center of excellence recognized nationally internationally, in distinctive areas of engineering education and research, based on a culture of innovation and invention."

Mission

"BIET contributes to the growth and development of its students by imparting a broad based engineering education and empowering them to be successful in their chosen field by inculcating in them positive approach, leadership qualities and ethical values."

Vision and Mission of the Computer Science and Engineering Department

Vision

To be a center-of-excellence by imbibing state-of-the-art technology in the field of Computer Science and Engineering, thereby enabling students to excel professionally and be ethical.

Mission

1. Adapting best teaching and learning techniques that cultivates Questioning and Reasoning culture among the students.
2. Creating collaborative learning environment that ignites the critical thinking in students and leading to the innovation.
3. Establishing Industry Institute relationship to bridge skill gap and make them industry ready and relevant.
4. Mentoring students to be socially responsible by inculcating ethical and moral values.

1. Program Educational Outcomes (PEOs):

PEO1	To apply skills acquired in the discipline Computer Science and Engineering for solving societal and industrial problems with apt technology intervention.
PEO2	To continue their career in industry or to pursue higher studies and research.
PEO3	To become successful entrepreneurs, innovators to design and develop software products on services that meets the societal, technical and business challenges.
PEO4	To work in the diversified environment by acquiring leadership qualities with effective communication skills accompanied by professional and ethical values.

2. Program Specific Outcomes (PSOs):

PSO1	Analyze and develop solutions for problems that are complex in nature by applying the knowledge acquired from the core subject of this program.
PSO2	Ability to develop Secure, Scalable, Resilient and distributed applications for industry and societal requirements.
PSO3	Ability to learn and apply the concepts and construct of emerging technologies like secure Artificial Intelligence, Machine learning, Big Data, Cloud Computing, IoT, Cloud computing, etc. for any real-time problems.

3. Course Outcomes:

CO1	Create, test and debug Android application by setting up Android development environment.
CO2	Implement adaptive, responsive user interfaces that work across a wide range of devices.
CO3	Infer long running tasks and background work in Android applications.
CO4	Demonstrate methods in storing, sharing and retrieving data in Android applications.
CO5	Infer the role of permissions and security for Android applications.

ABSTARCT

The Android Blood Bank App is designed to address the critical issue of blood donation and blood bank management. This mobile application aims to connect blood donors with those in need of blood, streamlining the process of finding and donating blood. The app provides a user-friendly interface for both blood donors and recipients. Donors can create profiles, specifying their blood type, availability, and location, making it easier for recipients to find suitable donors quickly. Recipients can search for donors based on blood type, location, and availability, ensuring timely access to the required blood units. Furthermore, the Android Blood Bank App incorporates additional features to enhance blood bank management. It allows blood banks to register and manage their inventory, ensuring accurate tracking of available blood units. The app also enables blood banks to send alerts and notifications to potential donors in emergency situations or when blood supplies are critically low.

In conclusion, the Android Blood Bank App offers a comprehensive solution to bridge the gap between blood donors and recipients. By leveraging the power of mobile technology, this mini project aims to promote efficient blood donation, save lives, and contribute to the overall improvement of blood bank management.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO ANDROID STUDIO

Android Studio is the official integrated development environment (IDE) for Android application development. It is based on the IntelliJ IDEA, a Java integrated development environment for software, and incorporates its code editing and developer tools. To support application development within the Android operating system, Android Studio uses a Gradle- based build system, emulator, code templates, and GitHub integration. Every project in Android Studio has one or more modalities with source code and resource files. These modalities include Android app modules, Library modules, and Google App Engine modules.

Android Studio uses an Instant Push feature to push code and resource changes to a running application. A code editor assists the developer with writing code and offering code completion, refraction, and analysis. Applications built in Android Studio are then compiled into the APK format for submission to the Google Play Store.

The software was first announced at Google I/O in May 2013, and the first stable build was released in December 2014. Android Studio is available for Mac, Windows, and Linux desktop platforms. It replaced Eclipse Android Development Tools (ADT) as the primary IDE for Android application development. Android Studio and the Software Development Kit can be downloaded directly from Google.

1.1.1 ARCHITECTURE OF ANDROID

Android architecture contains different number of components to support any android device needs. Android software contains an open-source Linux Kernel having collection of number of C/C++ libraries which are exposed through an application framework services. Among all the components Linux Kernel provides main functionality of operating system functions to smartphones and Dalvik Virtual Machine (DVM) provide platform for running an android application.

The main components of android architecture are following:-

- Applications
- Application Framework
- Android Runtime
- Platform Libraries
- Linux Kernel

Pictorial representation of android architecture with several main components and their sub components –

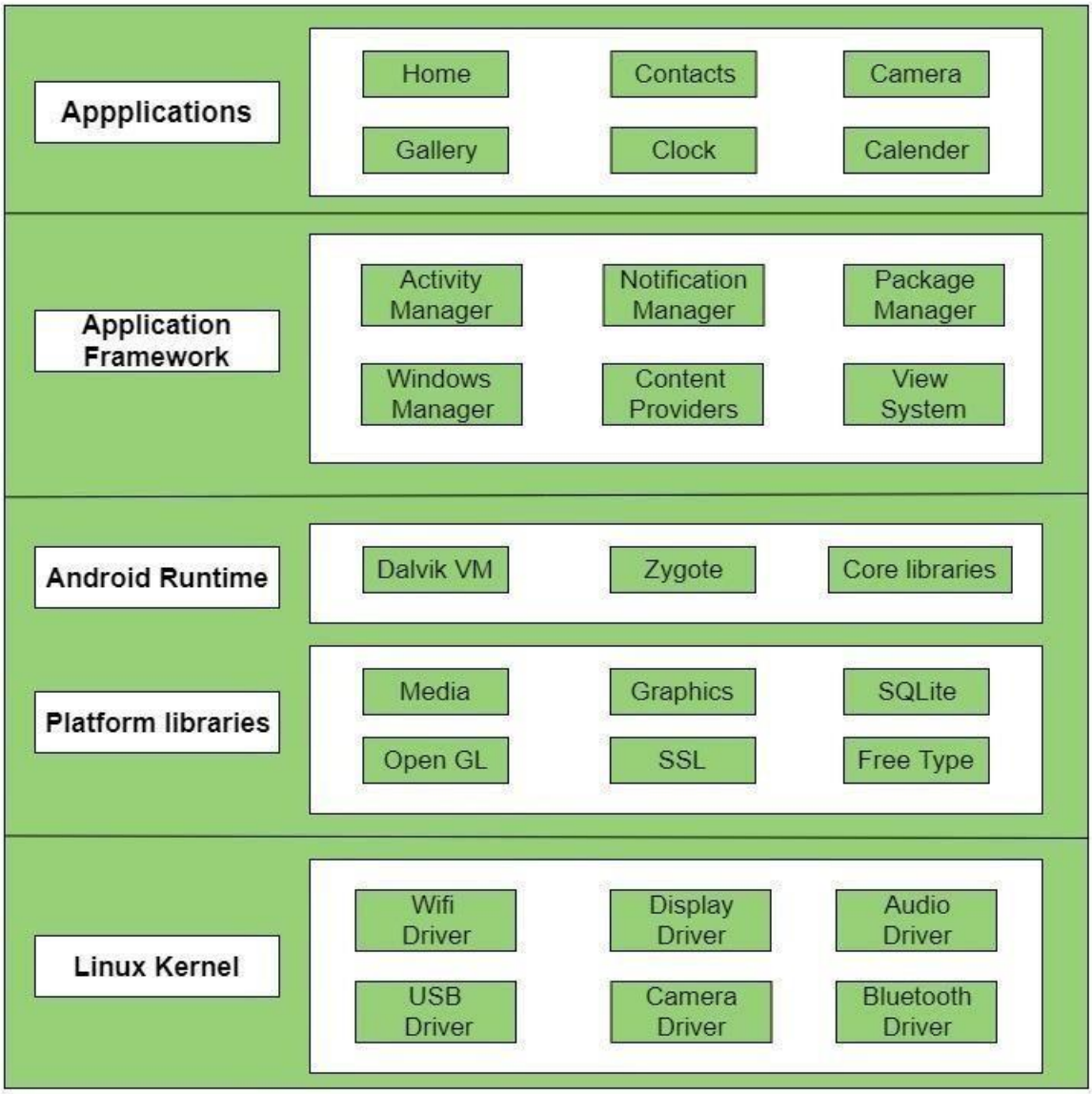

Fig:1.1 : Android Architecture

Applications –

Applications is the top layer of android architecture. The pre-installed applications like home, contacts, camera, gallery etc.. and third party applications downloaded from the play store like chat applications, games etc. will be installed on this layer only.
It runs within the Android run time with the help of the classes and services provided by the application framework.

Application framework –

Application Framework provides several important classes which are used to create an Android application. It provides a generic abstraction for hardware access and also helps in managing the user interface with application resources. Generally, it provides the services with the help of which we can create a particular class and make that class helpful for the Applications creation.
It includes different types of services activity manager, notification manager, view system, package manager etc. which are helpful for the development of our application according to the prerequisite.

Application runtime –

Android Runtime environment is one of the most important part of Android. It contains components like core libraries and the Dalvik virtual machine(DVM). Mainly, it provides the base for the application framework and powers our application with the help of the core libraries.
Like Java Virtual Machine (JVM), Dalvik Virtual Machine (DVM) is a register-based virtual machine and specially designed and optimized for android to ensure that a device can run multiple instances efficiently. It depends on the layer Linux kernel for threading and low-level memory management. The core libraries enable us to implement android applications using the standard JAVA or Kotlin programming languages.

Platform libraries –

The Platform Libraries includes various C/C++ core libraries and Java based libraries such as Media, Graphics, Surface Manager, OpenGL etc. to provide a support for android development.

- Media library provides support to play and record an audio and video formats.
- Surface manager responsible for managing access to the display subsystem.
- SGL and OpenGL both cross-language, cross-platform application program interface (API) are used for 2D and 3D computer graphics.
- SQLite provides database support and FreeType provides font support.
- Web-Kit This open source web browser engine provides all the functionality to display web content and to simplify page loading.
- SSL (Secure Sockets Layer) is security technology to establish an encrypted link between a web server and a web browser.

Linux Kernel –

Linux Kernel is heart of the android architecture. It manages all the available drivers such as display drivers, camera drivers, Bluetooth drivers, audio drivers, memory drivers, etc. which are required during the runtime.

The Linux Kernel will provide an abstraction layer between the device hardware and the other components of android architecture. It is responsible for management of memory, power, devices etc.

The features of Linux kernel are:

- Security: The Linux kernel handles the security between the application and the system.
- Memory Management: It efficiently handles the memory management thereby

providing the freedom to develop our apps.

- Process Management: It manages the process well, allocates resources to processes whenever they need them.
- Network Stack: It effectively handles the network communication.
- Driver Model: It ensures that the application works properly on the device and hardware manufacturers responsible for building their drivers into the Linux build.

1.1.2 INSTALLING AND RUNNING APPLICATIONS ON ANDROID STUDIO

To install and run applications on Android Studio, you can follow these steps:

1. Download and install Android Studio: Go to the official Android Studio website (https://developer.android.com/studio) and download the latest version of Android Studio suitable for your operating system (Windows, macOS, or Linux). Follow the installation instructions provided on the website.

2. Set up Android Virtual Device (AVD): Android Studio comes with an emulator called the Android Virtual Device (AVD) that allows you to test your applications on virtual devices. Launch Android Studio and open the "AVD Manager" by clicking on the AVD Manager icon in the toolbar or going to "Tools" -> "AVD Manager" in the menu bar.

3. Create a new Android project: After setting up the AVD, you can create a new Android project. Go to "File" -> "New" -> "New Project" in the menu bar. Follow the steps in the New Project wizard to configure your project, including selecting the project template, choosing the minimum SDK version, and configuring the activity.

4. Build and run the application: Once your project is set up, you can build and run your application on the virtual device. Click on the "Run" button in the toolbar, or go to "Run" >"Run 'app'" in the menu bar. Select the virtual device you created in the AVD Manager, and Android Studio will compile your code and launch the application on the virtual device.

1.1.2.1 SYSTEM REQUIREMENTS

- OS: Windows 8/8.1/10/11 (64-bit)
- CPU: 2nd generation Intel CPU (Sandy Bridge) or newer, AMD CPU.
- Memory: 8 GB RAM
- Free storage: 8 GB
- Screen resolution: 1920 * 1080

1.1.2.2 SET UP OF ANDROID STUDIO

To set up Android Studio, you can follow these steps:

Download Android Studio: Go to the official Android Studio website(https://developer.android.com/studio) and download the latest version of Android Studio for your operating system (Windows, macOS, or Linux). Install Android Studio: Once the download is complete, run the installer and follow the on-screen instructions to install Android Studio on your computer. The installation process may take a few minutes. Configure Android SDK: After installation, launch Android Studio. On the welcome screen, select "Configure" and then choose "SDK Manager." The SDK Manager allows you to download the necessary Android SDK components for development.

a. In the SDK Platforms tab, select the Android versions you want to target with your app. It's recommended to choose the latest stable version as well as the minimum version you want to support.
b. In the SDK Tools tab, select the components you need, such as the Android Emulator, Android SDK Build-Tools, and others. Again, it's recommended to use the latest stable versions.
c. Click "Apply" to start downloading and installing the selected SDK components.

1.1.2.3 CREATE ANDRIOD VIRTUAL DEVICE

To create an Android Virtual Device (AVD) in Android Studio, you can follow these steps:

1. Open Android Studio: Launch Android Studio on your computer.
2. Open AVD Manager: Once Android Studio is open, click on the "AVD Manager" icon in the toolbar. The icon looks like a smartphone with an Android logo.
3. Create a new virtual device: In the AVD Manager window, click on the "Create Virtual Device" button.
4. Select a device definition: You will be presented with a list of device types and configurations. Choose the device type that you want to emulate by selecting the desired category and then clicking on the "Next" button.
5. Select a system image: On the "System Image" tab, select the desired system image for the Android version you want to emulate. You can choose from various versions and API levels. Click on the "Next" button.
6. Configure the virtual device: Give your virtual device a name, and optionally, choose a skin and specify the device's hardware profile. You can also customize additional settings such as RAM size, internal storage, and camera emulation. Click on the "Finish" button when you're done.
7. Edit virtual device settings (optional): If you need to modify the settings of an existing virtual device, go back to the AVD Manager, select the virtual device, and click on the "Edit" button. From there, you can make changes to the virtual device's configuration.
8. Start the virtual device: Once you have created or configured a virtual device, you can start it by clicking on the green triangle play button next to the virtual device's name in the AVD Manager. The emulator will launch and the virtual device will start booting.
9. Wait for the virtual device to start: The virtual device may take some time to star

up, especially the first time. It will go through the Android boot process and eventually reach the home screen.

10. Use the virtual device for testing: Once the virtual device is running, you can use it for testing and running your Android applications. You can deploy and run your app on the virtual device from within Android Studio.

1.2 INTRODUCTION TO ANDROID BLOOD BANK APP

The Android Blood Bank App is an innovative solution designed to address the challenges faced by blood banks and individuals in need of blood donations. Blood shortage is a recurring issue in healthcare systems worldwide, and this project aims to leverage the power of mobile technology to bridge the gap between blood donors and recipients. The goal of the Android Blood Bank App is to provide a user-friendly platform that connects potential blood donors with individuals in need of blood transfusions. By creating a digital ecosystem, the app streamlines the process of finding and donating blood, making it more efficient and accessible. By combining the convenience of mobile technology with the critical need for blood donations, the Android Blood Bank App mini aims to contribute to the overall improvement of blood bank management and save lives.

1.3 HISTORY

Android is simply known as the official integrated development environment (IDE) for Google's Android operating system. It was built by JetBrains' IntelliJ IDEA and was mainly designed for Android development. You can also choose to download it through Windows, Mac OS, and even Linux-based operating systems that you may be using.It's also important to have in mind that this is simply a newer version of the Eclipse Android Development Tools which was the initial IDE for Android application development.

Android Studio was first released at a Google I/O conference in 2013, on March 16. Later, in May 2013, it was in the preview phase, or version 0.1. From here, it went into the beta phase in 2014 of version 0.8.

The first stable version of Android Studio was created in December 2014. It is also known as version 1.0.Another factor worth knowing is that on May 7, 2019, Java, the most endorsed programming language for Android app development, was replaced by Kotlin. Java is still available and supported in C++.

Android Version Evolution

Since the development of the first cell phone in 2008, Android has always made updates and developed new operating systems that are up to date. There are 12 versions of the operating system installed in more than 3 billion devices around the world.

In the beginning, Android versions were named after some sweet treats and even desserts, with their initial letters following the order of the alphabet. Starting with Android 10, this changed and only simple numbers were used as names.Now, let's take a closer look at the long list of Android versions and explore how they have evolved over time.

1. Android 1.5 – Cupcake

1.5 Cupcake was the first operating system on Android phones, and since it was the first version, it had simple or rather basic applications that people could use. Some of the applications found in this operating system are:

- Gmail
- Calendar
- YouTube
- Maps

All these started all the way back in April 2009.

Other functionalities that came along it include:

- Autorotation
- On-screen keyboard that replaced the physical keyboard

- Video recording
- Search widget
- Customizable home screen
- Third-party app framework

2. Android 1.6 – Donut

Android version 1.5 Cupcake did not last long, in September 2009.Android released a newer version - Donut. Donut was mainly developed as an improvement of the operating system to work perfectly on different screen sizes and even different resolutions. Another new feature that led to Android spreading worldwide and being favored by carriers around the world was the ability to support CDMA networks like Verizon. Finally, the Donut version also included the Power Option widget to control Wi-Fi, GPS, Bluetooth, and other functions.

3. Android 2.0 – 2.1 Éclair

Éclair came earlier than expected, as it was released immediately six weeks after the launch of Android Donut version. Regardless of the quick release, it brought extensive features like live wallpapers and even the drag-and-drop lock screen. Another important feature that was introduced was the voice guidance for Google Maps and also the text-to-speech feature that was supposed to be the alternative to the old keyboard input.

4. Android 2.2 – Froyo

Froyo was released in May 2010. What changes did it bring? Well, Android users whose phones supported this version could turn on the Wi-Fi hotspot and unlock their phones with numeric and alphabetic passwords instead of patterns.

5. Android 2.3 – Gingerbread

Gingerbread brought a newer operating system that featured an improved user interface

that was faster and simpler. It was also equipped with NFC technology and offered the possibility of voice and video chats via Google Talk. It also allowed the use of multiple cameras, which led to the introduction of the selfie camera.

6. Android 3.0 – Honeycomb

Honeycomb boasts of being the very first Android operating system designed mainly for large mobile devices like tablets. This version brought a nice user interface with a new action bar and also supported multitasking. Unfortunately, this is also one of the versions that were not very welcome on the market.

7. Android 4.0 – Ice-cream Sandwich

This version was introduced in October 2011. It came along with updates on the user interface, especially the introduction of the new font "Robot", soft buttons inherited from Honeycomb and a built-in photo editor. This OS version also allowed users to unlock their phones with biometric features, such as the face, since it supported the facial recognition feature. The final improvement that came with this operating system was the ability to set limits on data usage. You could monitor your data usage.

8. Android 4.1 – 4.3 Jelly Bean

The OS version Jelly Bean that was launched in July 2012 not only improved the user interface but also increased the performance. Other improvements that came with this operating system version were the ability to interact with notifications in different ways and even expand the notification buttons. This feature allowed users to turn on and off all types of notifications in an application depending on their preferences.

9. Android 4.4 – KitKat

This version of Android OS was named after the famous KitKat sweets and was released back in September 2013. It introduced many pillar improvements and functions. The first and most important is its capability of being able to run on smartphones having 512 MB of RAM. This made several devices able to update to this version.KitKat also boasted of having white elements instead of blue ones. Lastly, it was also the first-ever version to have the "Ok, Google" voice command service.

10. Android 5.0 – Lollipop

At the end of 2014, Google decided to release the Lollipop OS version. This version brought several improvements, such as Project Volta, which was supposed to improve battery life, and also Material Design, which consists of transitions and animations, shadows and even light effects. Finally , the new feature known as Tap and Go allows its users to transfer all their data to a new device through NFC and Bluetooth technology.

11. Android 6.0 – Marshmallow Announced in May 2015, the Marshmallow version was then officially released in October of the same year. This version came with the native fingerprint reader, USB - C and also Android Pay, which is since then known as Google Pay.

12. Android 7.0 – Nougat

Android 7.0 – Nougat introduced the split-screen feature that allows users to run two apps simultaneously .Besides this amazing feature, this version also introduced a customizable pull-down menu, then the fingerprint gesture to swipe down and also added some other emojis. Another improvement that was not easily noticed was the support for the Vulkan API an approach influencing future updates.

13. Android 8.0 – Oreo

Released in July 2017, Android 8.0 Oreo OS brought major changes to the visual menu.N However, the most important change was the introduction of Project Treble, which is a modular architecture designed to help hardware vendors keep up with evolving Android versions in a simple and fast way.

14. Android 9.0 – Pie

It is still the most popular OS version. A new home button has also been introduced, leading to an extended navigation button at the bottom of the screen. Besides the performance boost, battery optimization, and other security improvements, this version also introduced the Shush function, which puts the phone into Do Not Disturb mode.

15. Android 10

This is the very first Android OS version that has dropped all iconic names related to sweet treats that were seen in previous versions. This might have resulted in the lack of sweet names or a change in marketing strategy.This version in 2019 brought tons of security updates and a large amount of support for APIs and also support for the foldable phones that were supposed to be introduced at that time.

16. Android 11

A year after the release of Android 10, Google introduced Android 11, which offered several options to manage notifications and also controlled upgrades to monitor smart home devices as we embrace the Internet of Things in our lives. Finally, this OS version also introduced the screen recording feature and chat bubbles.

17. Android 12

At the time of this article's publication, the latest Android operating system is available in Android 12, which was released in October 2021.

According to information from the Android Developers Blog, this operating system offers the best user experience, improved energy efficiency, and easier Wi-Fi sharing, among other improvements.

18. Android 13

A few days ago, the first announcement about Android 13 was published. Not much is known yet, but exciting new features are in store. The fact that it will be possible to run Windows on a virtual machine on Android 13 in the new Android version is discussed quite intensely. As Android developer Danny Lin (aka kdrag0n) has shown on Twitter**.**

1.4 OBJECTIVES :

To develop Android Blood Bank App, we have founded following objectives:

- To connect blood donors with recipients in a efficient manner and timely access toblood for patients in need.
- To increase Blood Donation Awareness by educating and raising awareness about theimportance of blood donation is a crucial objective.
- To Establishing partnerships and collaborations with healthcare organizations, hospitals, and existing blood banks.
- To develop an app that can facilitate the quick and efficient delivery of blood units during emergencies.

1.5 IMPORTANT FEATURES :

- Interactive User Interface and efficient user experience:

Android provides exciting widgets and tools to make your creativity and imagination turn into reality. Android developers keep on adding exciting features and widgets to make the user interface quite interactive and smooth. You can use these tools to make your design and then publish that design.

- Larger Community Support:

Google being the holder of Android, regularly updates forums, provides stable releases and supports all budding developers. One can use their developer guides, tools, and discussions to resolve their bugs and developmental issues.As the community is quite extensive and Android is open source, it is easy to develop such exciting apps without any difficulty.

- Easy to share your work and earn money:

Google provides flexibility in distributing the apps developed by anyone. After one develops an app, he or she can quickly go to the Google Play store to publish his/her app and then earn money. It doesn't stick to the Google Play store. One can even share his/her app through other app stores of other manufactures or even directly share through any network.

- Regular Updates and Security Fixes

Google always tries to give its user the best experience and security. To maintain such standards, google updates android to its latest stable version with the least to often no bugs and security issues. Therefore, one can assure safety while working on the stable versions of android.

1.6 ADVANTAGES OF ANDROID STUDIO

1.6.1 **Android is an open-source platform allowing UI customization**:

Licensed under Apache, Android is an open source operating system whose codes developers can change to build customized User Interface. App developers building applications for this platform can get access to the core codes and are at a liberty to change the them to get the customized outcomes. This is not possible when it comes to iOS and app have to strictly adhere to the core code specifications when developing apps for the specific platform.

1.6.2 **Supports cloud storage enabling sync of devices with G-account**:

Since Android is a Google product, users having Gmail account can have access to cloud storage that are supported by the tech company. This means that users can sync devices using Google accounts. Moreover, Google gives 15GB of free cloud storage to every user that is good for an average person using it for personal purposes.

1.6.3 Continual improvement & removal of old features:

Google Android is supported by a huge community of developers and also users who continue to give feedback about the features, their pros and cons. As a result, there is continuous check on the codes and features, making modifications and alterations, bringing in better upgrades all the time. This is one of the reasons why Android is always adding new features while removing older ones that users do not like.

1.6.4 Supports 3rd party widget & information display on screen:

Android gives users the freedom to download 3rd party widgets and also display their content on the home screen. If a user wishes to view time and temperature shown by a specific widget on the home screen, it is possible with devices running on the platform.

1.6.5 Supports running multiple apps simultaneously:

With Android running on a device with good hardware specification, as a user you can have multiple apps running simultaneously. You can continue to listen to music as you check your messages or download files that you've received or even upload them from your device or drive. There are a lot of Android app development companies who build applications based on Android that are very useful in our daily lives.

1.6.6 Expandable memory & runs on affordable large devices:

One of the biggest advantages of using devices running on Android platform is that it supports expandable memory. iOS devices on the other hand do not support external memory expansion by adding memory card to the phone. Users of this platform enjoy the privilege of storing e-books, music, videos and games on their devices.

1.6.7 Wide range of devices to choose from:

Android users are spoilt for choice of smartphone devices of different prices. There is something for people across all budget spectrum when it comes to smartphones running on Android. Almost all companies build devices that support Android platform giving users multiple options when buying. This is something that is very restrictive for Apple users who have to stick to the company's expensive devices only.

CHAPTER 2

REQUIREMENT SPECIFICATION

2.1 FUNCTIONAL REQUIREMENTS:

1. User Registration: The app should allow users to create an account by providing their basic information such as name, contact details, blood type, and any relevant medical history.

2. Blood Donation Requests: Registered users should be able to create blood donation requests specifying the required blood type, quantity, and location.

3. **Donor Search**: Users should be able to search for registered blood donors based on criteria such as blood type, location, and availability. The app should display a list of matching donors with their contact details.

4. **Donor Profile**: The app should provide a profile section for donors to manage their information, including their blood type, contact details, availability for donation, and any changes to their health status.

5. Donation History: The app should maintain a record of the user's blood donation history, including the dates of previous donations, locations, and the blood bank or organization that received the donation.

6. Emergency Services: The app should have an emergency feature that allows users to quickly call for emergency blood transfusion services or notify nearby registered donors in case of critical blood requirement situations.

2.2 NON- FUNCTIONAL REQUIREMENTS:

1. Usability: The app should have an intuitive and user-friendly interface, with clear navigation and well-organized features. It should be easy to learn and use for individuals with varying levels of technological proficiency.

2. Performance: The app should provide a responsive and smooth user experience, with minimal loading times and delays. It should handle user interactions efficiently, even under peak usage scenarios, ensuring a seamless and uninterrupted operation.

3.Reliability: The app should be stable and dependable, with a high level of availability.
It should minimize system crashes, errors, and unexpected downtime, ensuring users can rely on it for critical blood donation needs.

4. Scalability: The app should be designed to accommodate future growth and increasing user demands. It should be scalable to handle a growing user base and additional features without sacrificing performance.

5. Security: The app should employ robust security measures to protect user data and ensure privacy. It should implement encryption protocols, secure authentication mechanisms, and prevent unauthorized access to sensitive user information.

6. Compatibility: The app should be compatible with a wide range of Android devices, including different screen sizes, resolutions, and operating system versions. It should be optimized for various device configurations to ensure consistent performance.

7. Accessibility: The app should be designed with accessibility in mind, adhering to accessibility guidelines and standards. It should support features such as alternative text for images, screen reader compatibility, and adjustable font sizes for users with disabilities.

2.3 SOFTWARE REQUIREMENTS:

The software required for the development of this project is:

- **Application Required** : Android studio Electric Eel
- **Operating System** : Windows 10(and other higher version)
- **Markup Language** : XML
- **Programming Language** : JAVA

2.4 HARDWARE REQUIREMENTS:

The hardware required for the development of this project is:

- **Processor** : Intel Core i5 Processor or higher
- **System Type** : 64-Bit Operating System
- **RAM** : 8 GB RAM Minimum
- **Mobile Device** : Redmi 6 pro or any Android device
- **Screen Resolution** : 1920 * 1080

CHAPTER 3

DESIGN

3.1 INITIAIZATION

- Project Planning: Start by defining the project scope, goals, and requirements of the blood bank app. Identify key stakeholders and establish a project timeline. Determine the target audience and the specific features and functionalities the app should include.

- Team Formation: Assemble a team of developers, designers, and testers with expertise in Android app development. Assign roles and responsibilities to team members based on their skills and experience.

- Technology Stack: Choose the appropriate technology stack for developing an Android app, including programming languages like Java or Kotlin, Android Studio as the development environment, and any necessary frameworks or libraries.

- Data Architecture: Plan the data architecture for the app, including the database structure, data models, and data storage options. Determine whether to use a local database or a server-side database for storing user information, donation requests, and other relevant data.

- User Interface Design: Begin the process of designing the user interface (UI) and user experience (UX) of the app. Create wireframes and mockups to visualize the layout, navigation flow, and screen designs. Ensure the UI is intuitive, visually appealing, and aligned with the app's goals.

3.2. DESIGN

- User Registration and Login: Design screens and workflows for user registration and login processes. Include fields for users to provide their basic information, such as name, contact details, blood type, and any relevant medical history. Ensure the login process is secure, with appropriate authentication mechanisms.

- Blood Donation Requests: Create screens for users to create and manage blood donation requests. Allow users to specify the required blood type, quantity, and location. Implement notifications to alert nearby registered donors who match the requested blood type.

- Donor Search: Design screens that enable users to search for registered blood donors based on criteria such as blood type, location, and availability. Display a list of matching donors along with their contact details.

3.3 FLOW CHART

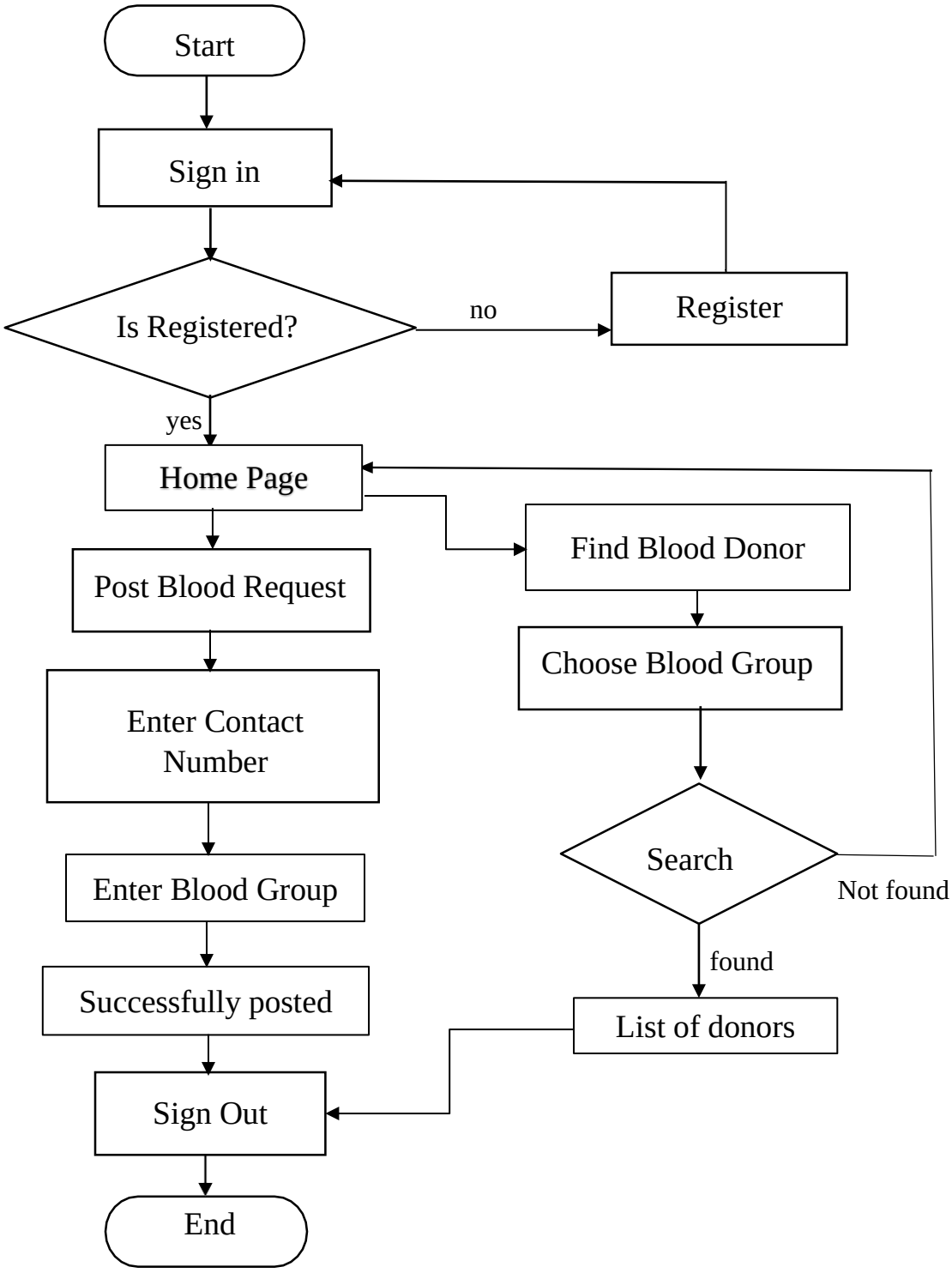

Fig 3.1: Block diagram of Android Blood Bank App

CHAPTER 4

IMPLEMENTATION

4.1 OVERVIEW

In this project we exhibit overview of Android Blood Bank App. We have taken the help of Android Studio (Electric Eel) which brought all the codes together to create app. It will also handle AI and physicsroutines.

4.2 CODE:

```
package com.mad.bloodbank;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.PopupMenu;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.Gravity;
import android.view.View;
import android.widget.EditText;
import android.widget.FrameLayout;
import android.widget.Toast;
import com.google.android.material.snackbar.Snackbar;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.vinayak09.bloodbankbyvinayak.adapters.UserAdapter;
import com.vinayak09.bloodbankbyvinayak.model.User;
import java.util.ArrayList;
import java.util.HashMap;
public class DisplayDonorsActivity extends AppCompatActivity {
    RecyclerView list;
    UserAdapter adapter;
    ArrayList<User> users,temp;
    EditText districtFilter;
    User self;
    String uid = FirebaseAuth.getInstance().getUid();
  PopupMenu popupMenu;
  @Override
```

```
  protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_display_donors);
     initializeComponents();
     getDonors();
  }
  void initializeComponents() {
     popupMenu = new PopupMenu(this, findViewById(R.id.more));
     popupMenu.getMenuInflater().inflate(R.menu.donors_menu,popupMenu.getMenu());
     popupMenu.setOnMenuItemClickListener(item -> {
        if(item.getItemId() == R.id.changePass){
           FirebaseAuth.getInstance().sendPasswordResetEmail(self.getEmail());
           Snackbar snack = Snackbar.make(findViewById(android.R.id.content),"Password
Reset Link Sent On Registered Email.", Snackbar.LENGTH_LONG);
           View view1 = snack.getView();
           FrameLayout.LayoutParams params
=(FrameLayout.LayoutParams)view1.getLayoutParams();
           params.gravity = Gravity.CENTER_VERTICAL;
           view1.setLayoutParams(params);
           if(self!=null) {
              FirebaseAuth.getInstance().sendPasswordResetEmail(self.getEmail());
           }else {
              snack.setText("Error Occurred!");
           }
           snack.show();
        }else if(item.getItemId() == R.id.logout){
           FirebaseAuth.getInstance().signOut();
           startActivity(new Intent(this,SplashScreen.class));
           DisplayDonorsActivity.this.finish();
        }else if(item.getItemId() == R.id.visibleDonors){
           if(item.isChecked()){
              item.setChecked(false);
              updateVisible(false);
           }else {
              item.setChecked(true);
              updateVisible(true);
           }
        }
        return true;
     });
     self = new User();
     districtFilter = findViewById(R.id.districtFilter);
     list = findViewById(R.id.donorsList);
     users = new ArrayList<>();
     temp = new ArrayList<>();
     adapter = new UserAdapter(this, users, position -> {
        //Handle call button event
        Intent intent = new Intent(Intent.ACTION_DIAL);
        intent.setData(Uri.parse("tel:"+temp.get(position).getMobile()));
        startActivity(intent);
     }, position -> {
```

```
        User sent = temp.get(position);
        Intent sendIntent = new Intent();
        sendIntent.setAction(Intent.ACTION_SEND);
        sendIntent.putExtra(Intent.EXTRA_TITLE,"Be Hero, Donate Blood.");
          sendIntent.putExtra(Intent.EXTRA_TEXT, "Hello.\n"+"Here is the Information about
blood Donor:\n"+sent.getFName()+" "+sent.getLName()+"\nBlood Group :
"+sent.getBloodGroup()+"\nAddress:"+sent.getState()+" "+sent.getDistrict()+"
"+sent.getTehsil()+"\nMobile Number :"+sent.getMobile());
        sendIntent.setType("text/plain");
        Intent shareIntent = Intent.createChooser(sendIntent, "Be Hero, Donate Blood.");
        startActivity(shareIntent);
    });
    list.setLayoutManager(new LinearLayoutManager(this));
    list.setAdapter(adapter);
    districtFilter.addTextChangedListener(new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int after) {
        }
        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count) {
        }
        @Override
        public void afterTextChanged(Editable s) {
            updateList(s.toString());
        }
    });
 }
 private void updateVisible(boolean b) {
    HashMap<String, Object> updateValues = new HashMap<>();

    if(b){
        updateValues.put("Visible","True");
    }else {
        updateValues.put("Visible","False");
    }

FirebaseDatabase.getInstance().getReference("Donors").child(uid).updateChildren(updateValu
es);
 }

 private void updateList(String s) {
    System.out.println(s);
    temp.clear();
    for( User v : users){
        if(v.getDistrict().toUpperCase().contains(s)||s.equalsIgnoreCase("ALL")) {
            System.out.println(v.getDistrict());
            temp.add(v);
        }
    }
    adapter.updateList(temp);
 }
```

```
  private void getDonors() {
        FirebaseDatabase.getInstance().getReference("Donors").addValueEventListener(new
 ValueEventListener() {
          @Override
          public void onDataChange(@NonNull DataSnapshot snapshot) {
             users.clear();
             temp.clear();
             for(DataSnapshot ds:snapshot.getChildren()){
                User user = ds.getValue(User.class);
                if(user.getStep().equals("Done")) {
                   if (user.getVisible().equals("True")) {
                      users.add(user);
                      temp.add(user);
                   }
                   if (user.getUID().equals(uid)) {
                      self = user;

 popupMenu.getMenu().findItem(R.id.visibleDonors).setChecked(self.getVisible().equals("Tru
 e"));
                   }
                }
             }
             updateList(districtFilter.getText().toString());
             filterList();
          }

          @Override
          public void onCancelled(@NonNull DatabaseError error) {
          }
       });
  }
  private void filterList() {
     adapter.notifyDataSetChanged();
  }

  public void viewRequestList(View view) {
     startActivity(new Intent(DisplayDonorsActivity.this,DispalyRequestsActivity.class));
     this.finish();
  }

  public void popUp(View view) {

     popupMenu.show();
  }
}
```

CHAPTER 5

TESTING

The following testing approaches were employed during the testing phase:

1. **Unit Testing**: Unit testing was performed to validate individual components and modules of the app. This involved testing each unit in isolation to ensure they functioned correctly. Key functionalities, such as user registration, login, and blood donation requests, were thoroughly tested to ensure they operated as expected.

2. **Integration Testing**: Integration testing was conducted to assess the interaction between different components of the app. The primary focus was on verifying the seamless integration of modules and functionalities.

3. **System Testing**: System testing was performed to evaluate the app as a whole, ensuring that all components work together harmoniously. The entire flow of the app, from user registration to blood donation requests and notifications, was tested to validate its functionality and usability. Scenarios such as handling network interruptions, error handling, and edge cases were considered during this phase.

4. **Usability Testing**: Usability testing was conducted to gauge the app's user-friendliness and intuitiveness. A group of representative users were invited to perform specific tasks and provide feedback. The feedback was used to improve the app's user interface, navigation, and overall user experience.

5. **Compatibility Testing**: Compatibility testing was carried out to ensure the app functions correctly across different Android devices, screen sizes, and operating system versions. The app was tested on various devices, including smartphones and tablets, running different versions of Android to identify any compatibility issues.

6. **Performance Testing**: Performance testing aimed to assess the app's responsiveness, stability, and resource usage under different scenarios. Load testing was performed to determine the app's behavior when multiple users simultaneously access the app and perform various activities. Performance benchmarks were established to measure response times and resource consumption.

CHAPTER 6

SNAPSHOTS

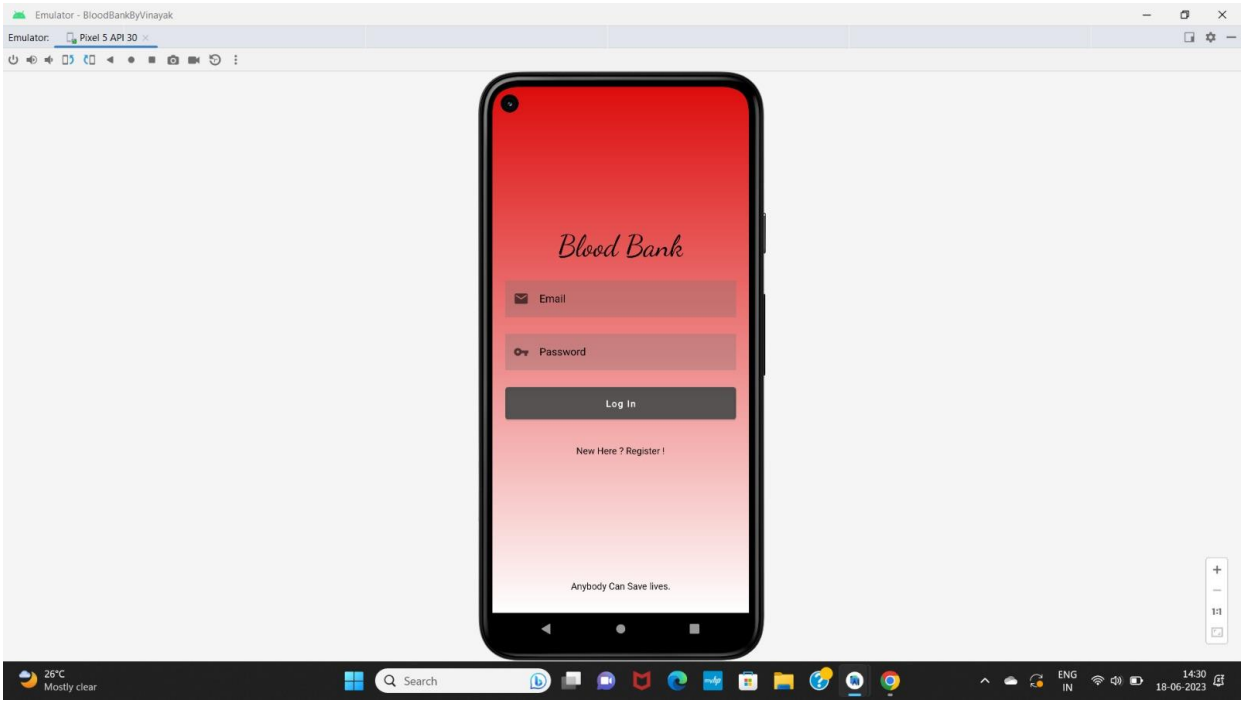

Fig 6.1: Login Page

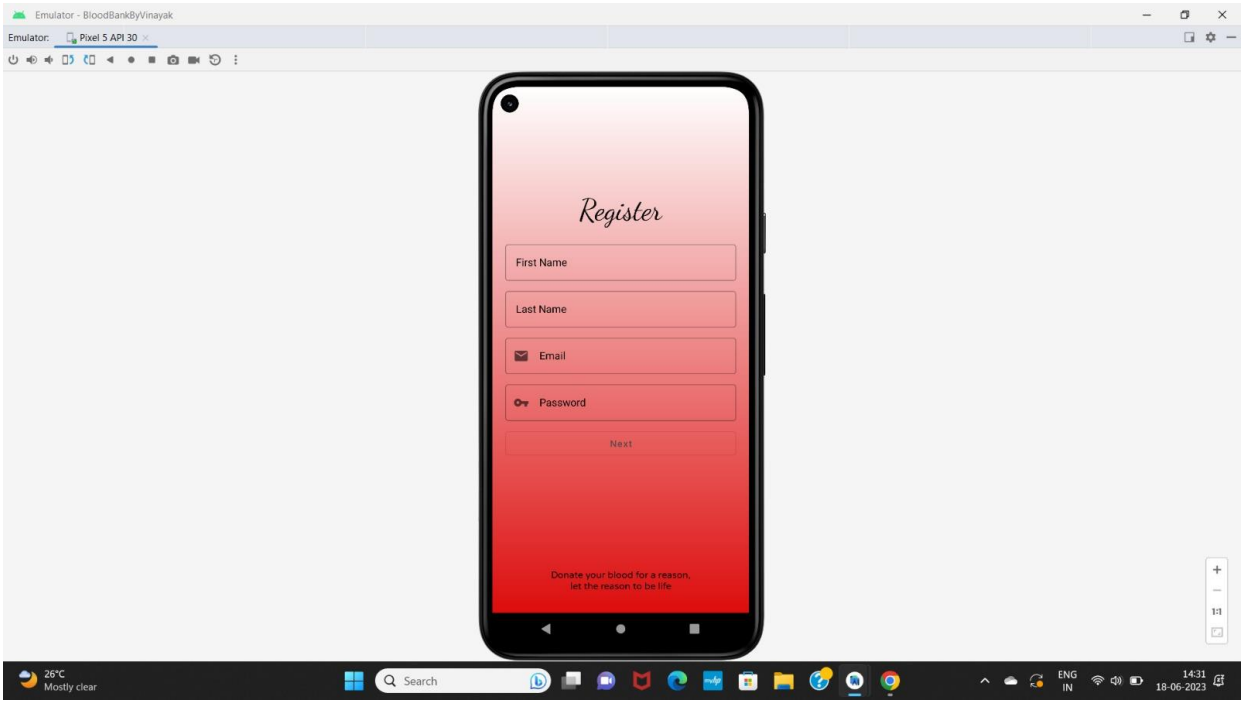

Fig 6.2: Registration Page

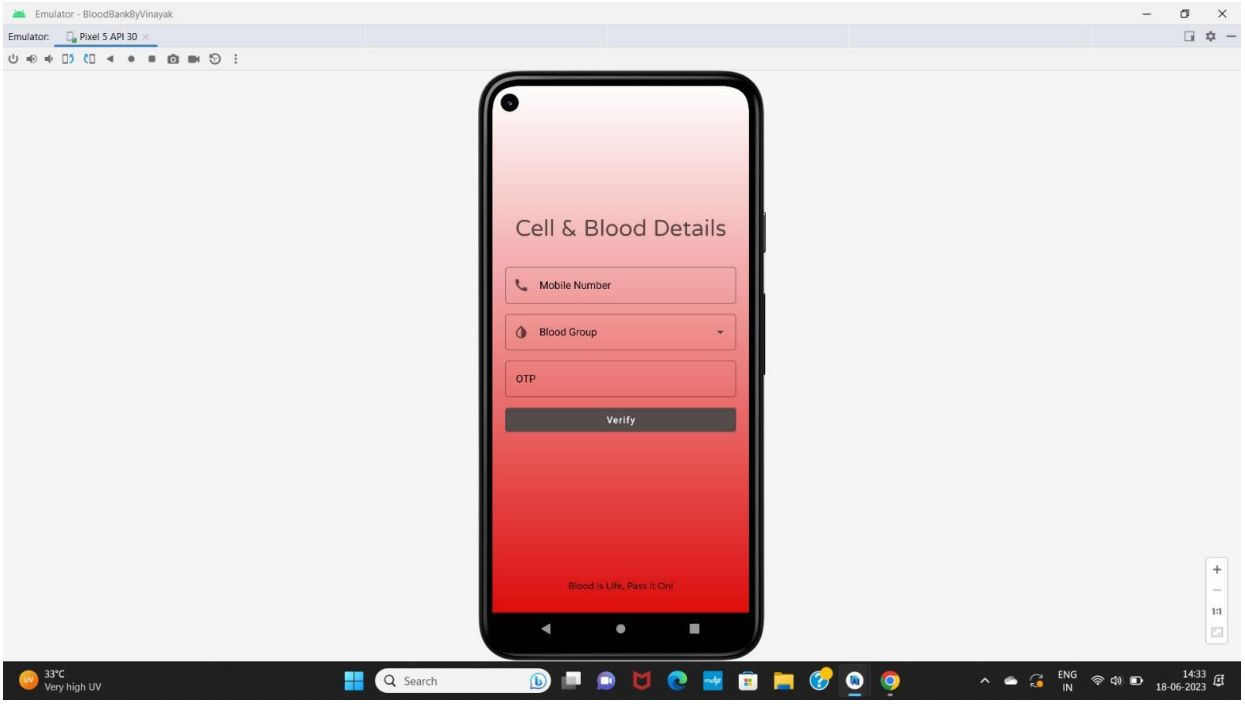

Fig 6.3: Verification and Blood Details

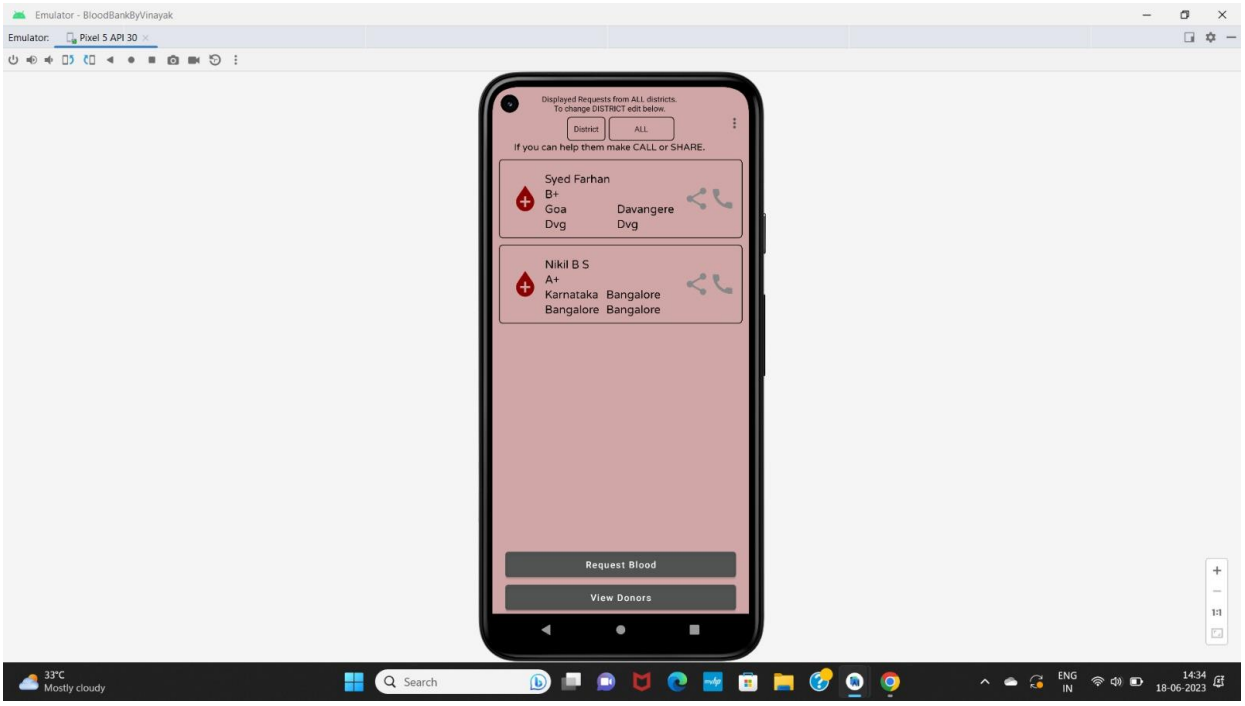

Fig 6.4: Blood Requests

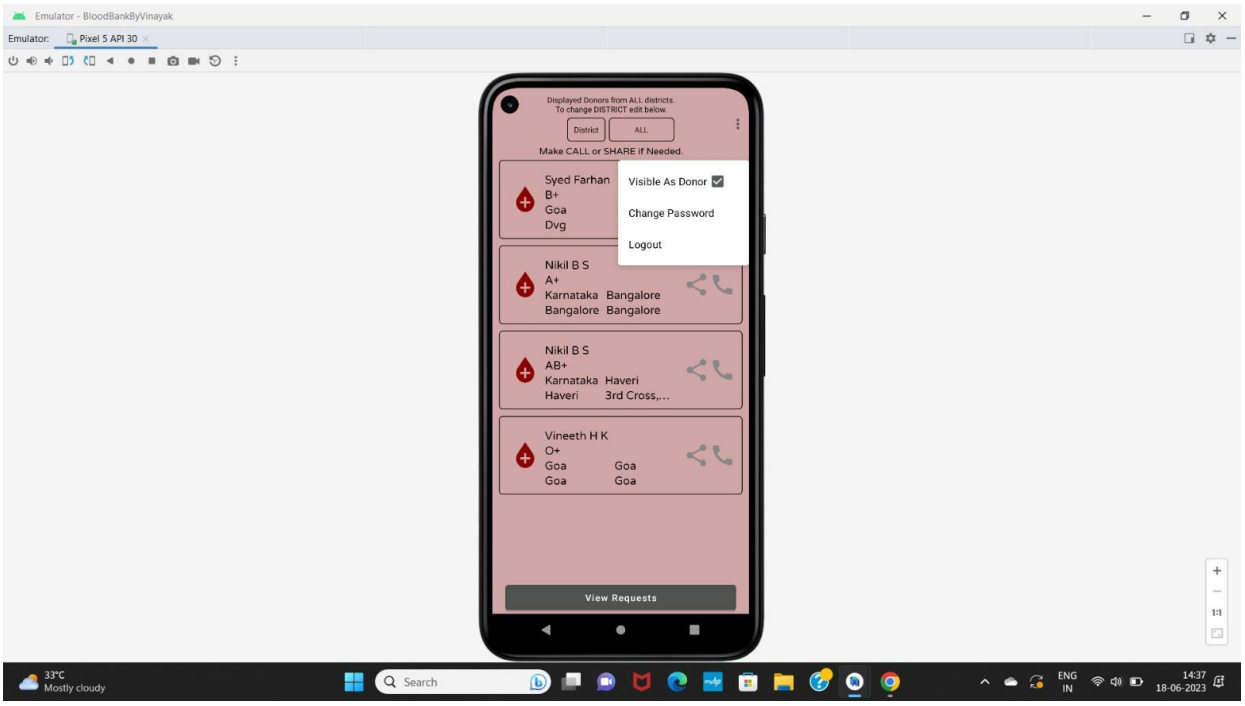

Fig 6.5: Blood Donors List

CONCLUSION

In conclusion, the development of an Android Blood Bank App provides a valuable solution for connecting blood donors and recipients efficiently. The app's functional and non-functional requirements, such as user registration, blood donation requests, donor search, and notification system, contribute to creating a user-friendly and reliable platform. The app's usability and performance are crucial in ensuring a seamless user experience. By implementing an intuitive interface, responsive interactions, and scalability, users can easily navigate through the app, search for donors, and make blood donation requests without encountering significant delays or system crashes.

Security and privacy are of utmost importance in a blood bank app. With robust authentication mechanisms, encryption protocols, and compliance with data protection regulations, users can trust that their personal information and medical history are kept confidential and protected.

REFERENCES

BOOKS:

1. Google Developer Training, "Android Developer Fundamentals Course – Concept Reference", Google Developer Training Team, 2017.
2. Erik Hellman, "Android Programming – Pushing the Limits", 1st Edition, Wiley India Pvt Ltd, 2014. ISBN-13: 978-8126547197
3. Dawn Griffiths and David Griffiths, "Head First Android Development", 1st Edition, O'Reilly SPD Publishers, 2015. ISBN-13: 978-9352131341
4. Bill Phillips, Chris Stewart and Kristin Marsicano, "Android Programming: The Big Nerd Ranch Guide", 3rd Edition, Big Nerd Ranch Guides, 2017. ISBN-13: 978-0134706054

REFERENCES:

- https://www.gitbook.com/book/google-developer-training/android-developer-fundamentals-
- https://cloud.smartdraw.com/editor.aspx?templateId=490dad73-de30-42bf-9a58-1789d56c1afd&flags=128#depoId=36058085&credID=-39639645