



**PRESIDENCY COLLEGE**  
(AUTONOMOUS)  
AFFILIATED TO BENGALURU CITY UNIVERSITY, APPROVED BY AICTE, DELHI & RECOGNISED BY THE GOVT. OF KARNATAKA  
RE-ACCREDITED BY NAAC WITH 'A+' GRADE

**BHOJAN SAAJHA**



**PRESIDENCY COLLEGE**  
(AUTONOMOUS)

AFFILIATED TO BENGALURU CITY UNIVERSITY, APPROVED BY AICTE, DELHI & RECOGNISED BY THE GOVT. OF KARNATAKA  
RE-ACCREDITED BY NAAC WITH 'A+' GRADE

A Project Report on

**“BHOJAN SAAJHA”**

Submitted in Partial Fulfillment of the Requirements For the award of the degree

**Master of Computer Applications**

**SUBMITTED BY**

**PAURUSH KUMAR BANSAL**

**22P01056**

**PRESIDENCY COLLEGE**

Kempapura, Hebbal, Bengaluru – 24

**Re-accredited by NAAC with 'A+' Grade**

**DEPARTMENT OF COMPUTER APPLICATIONS**



# PRESIDENCY COLLEGE

## (AUTONOMOUS)

AFFILIATED TO BENGALURU CITY UNIVERSITY, APPROVED BY AICTE, DELHI & RECOGNISED BY THE GOVT. OF KARNATAKA  
RE-ACCREDITED BY NAAC WITH 'A+' GRADE

### CERTIFICATE

*This is to certify that **PAURUSH KUMAR BANSAL** with Register No. **22P01056** has satisfactorily completed the fourth semester MCA Project titled “**BHOJAN SAAJHA**”, as a partial fulfillment of the requirements for the award of the Degree in ***Master of Computer Applications***, awarded by ***Bengaluru City University***, during the Academic Year **2023-2024**.*

**Project Guide:**

**Head of Department**

(Department of computer Application)

**Examiners**

**Reg No:** -----

1. -----

**Examination Center:** -----

2. -----

**Date of the exam:** -----

# Declaration

The project titled “**BHOJAN SAAJHA**” developed by me in the partial fulfillment for the award of Master of Computer Application. It is a systematic work carried by us under the guidance of **Mr. Dhanasingh B. Rathod**, Assistant professor, Department of Computer Applications.

I, declare that this same project has not been submitted to any degree or diploma to the Bengaluru City University or any other Universities.

Name of the student: -

Date: -

Signature: -

-----

# Acknowledgement

The development of software is generally bit complex and time-consuming task. The goal of developing the project “**BHOJAN SAAJHA**” could not be archived without the encouragements of kindly helpful and supportive people. Here by we convey our sincere thanks for all of them.

I take this opportunity to express my gratitude to people who had been instrumental in the successful completion of this project.

I am thankful to our management trustee for providing us an opportunity to work and complete the project successfully.

I wish to express my thanks to our **Principal Dr. Suchithra R Nair** for her support to the project work. I would like to acknowledge my gratitude to our HOD of Master of Computer Applications, **Dr. Alli A** for her encouragement and support. Without their encouragement and guidance this project would not have materialized.

The guidance and support received from our Internal Guide **Mr. Dhanasingh B. Rathod** who contributed to this project, was vital for the success of the project. We are grateful for his constant support and help.

## INDEX

SL.NO	DESCRIPTION	PAGE NO.
01.	INTRODUCTION	6.
02.	REQUIREMENT ANALYSIS	9.
03.	SYSTEM SPECIFICATIONS	12.
04.	H/W and S/W CONFIGURATIONS	14.
05.	SOFTWARE PROFILE	16.
06.	SYSTEM DESIGN	40.
07.	SYSTEM TESTING	47.
08.	UI DESIGN	52.
09.	SYSTEM IMPLEMENTATION - CODING	59.
10.	CONCLUSION and FUTURE ENHANCEMENTS	117.
11.	BIBLIOGRAPHY	119.



# ***Introduction***

---

## **1. INTRODUCTION**

---

### **1.1 OBJECTIVE:**

The primary objective of the " BHOJAN SAAJHA " app is to bridge the gap between surplus food in restaurants and individuals in need. The app will facilitate the donation of excess food to poor people, ensuring that no food goes to waste and helping to alleviate hunger in the community.

" BHOJAN SAAJHA" is an innovative Android application designed to help locate free food for the needy and provide a platform for restaurants and individuals to donate excess food. The app will serve as a real-time solution to hunger by connecting donors with recipients, thereby promoting food security and reducing food wastage.

### **MAJOR MODULES:**

- User Module
- Donor Module
- Search and Map Module
- Notification Module
- Feedback Module

### **Functionalities of Modules**

#### **User Module can perform the following operations**

- Registration and login
- Profile management (update user details)
- Location services (set and update location)

#### **Donor Module can perform the following operations:**

- Restaurant registration

- Food listing (add, update food items)
- Status updates (update availability of food)

**Search and Map Module can perform the following operations**

- Search for free food locations
- Map integration (view food locations on the map)

**Notification Module can perform the following operations**

- Send real-time notifications to users about available food

**Feedback Module can perform the following operations**

- Collect user feedback
- Manage enquiries and respond to feedback





# ***Software Requirement Analysis***

---

## **2. SOFTWARE REQUIREMENT ANALYSIS**

---

System Analysis is a detailed study of the various operations performed by the "BHOJAN SAAJHA" application and its relationships with external entities. Analysis begins when developers conduct a thorough examination of the existing food donation processes and systems. During the analysis phase, data is collected regarding how food donations are managed, from restaurants' surplus food to how individuals in need access it. Common tools like Data Flow Diagrams (DFDs), interviews, and questionnaires are used to gather relevant information.

The analysis requires training, experience, and a logical approach to collect accurate data crucial for developing the app. The success of the "BHOJAN SAAJHA" system hinges on clearly defining the food donation challenges, conducting a thorough investigation, and choosing the appropriate solutions. A well-structured analysis model not only aids in understanding the problem of food waste and hunger but also serves as the framework for crafting an efficient solution.

By analysing the current food donation system and identifying gaps, the proposed "BHOJAN SAAJHA" app will be carefully designed to meet the needs of both donors and recipients, ensuring a practical solution that helps reduce food wastage and provide for the hungry

### **2.1 PRESENT SYSTEM**

Currently, there is no centralized platform to manage and distribute excess food from restaurants to those in need, leading to significant food wastage. Many restaurants end up discarding edible food due to the absence of an organized donation process. While some volunteer groups and organizations attempt to coordinate food donations, these efforts are often fragmented and inefficient, lacking the scale needed to make a significant impact. The process is largely manual, making it difficult for restaurants to identify where to donate, and for recipients to access available food.

Additionally, without a real-time mechanism for communication, many opportunities for food distribution are missed. The lack of a streamlined approach makes it challenging to track donations, ensure food safety, and effectively match surplus food with those who need it most. This disorganization results in a substantial amount of food being wasted daily, highlighting

the need for a solution like "BHOJAN SAAJHA" to simplify and improve the food donation process.

## **2.2 PROPOSED SYSTEM**

The "BHOJAN SAAJHA" app will effectively address food wastage and hunger by providing a centralized, real-time solution connecting food donors and recipients. Key features of the system include:

- **User-Friendly Interface:** The app will offer an intuitive interface, allowing easy navigation for users to find free food locations. Both donors (restaurants and individuals) and recipients will have streamlined access to the platform.
- **Surplus Food Listings:** Restaurants can register and list surplus food with details like type, quantity, and expiry, making food donations fast and efficient.
- **Real-Time Updates:** Users will receive real-time notifications when new food is available, ensuring quick access to fresh food before it is wasted.
- **Map Integration:** The app's map feature will help users locate nearby donation points, with directions provided for convenience.
- **Location-Based Matching:** Geolocation services will match users with nearby food donation points, promoting quick and efficient distribution.
- **Donation Tracking:** Donors and recipients can track their activity, maintaining transparency and accountability within the platform.
- **Feedback System:** Recipients can provide feedback on donated food, ensuring safety and quality standards are maintained.
- **Multi-Language Support:** The app will offer support for multiple languages to cater to diverse users.

By simplifying the donation process and offering real-time solutions, "BHOJAN SAAJHA" will reduce food wastage while efficiently distributing surplus food to those in need.



# ***Software Requirement Specifications***

---

### **3. SOFTWARE REQUIREMENT SPECIFICATION**

---

The Software Requirements Specification (SRS) for the "BHOJAN SAAJHA" app outlines its functionalities and performance expectations, serving as a foundation for development. This document minimizes the time and effort required for developers to achieve project goals while reducing overall costs. By detailing the app's operations and interactions with users and system components, the SRS ensures that all stakeholders understand the project's scope.

For "BHOJAN SAAJHA," core requirements were analysed at the project's outset, aiding in the selection of appropriate hardware and software. The app will be developed using Android Studio for compatibility with various Android devices and will utilize Firebase for efficient real-time data management, including food listings and user notifications.

Designed for high operating speed, the app will provide real-time notifications and quick access to food availability. Security is prioritized; user data, including personal information and location, will be safeguarded through encryption and secure transmission. The app will be available 24/7, allowing users to access features anytime.

"BHOJAN SAAJHA" will also be portable across devices, promoting user engagement. Its maintainability will enable efficient updates and enhancements, while recovery mechanisms will ensure minimal disruption during technical issues. Comprehensive testing will validate performance across different devices and browsers. By adhering to the SRS guidelines, "BHOJAN SAAJHA" aims to effectively connect food donors with those in need, fostering food security and reducing waste in communities.



# ***H/W and S/W Configurations***

## 4. SOFTWARE AND HARDWARE CONFIGURATION

<b>HARDWARE</b>	
Processor	Intel i5 or higher
RAM	8 GB or higher
HDD	256 GB SSD or higher
<b>SOFTWARE</b>	
CLIENT-SIDE TECHNOLOGIES	Android SDK, Java, XML for UI design
SERVER-SIDE TECHNOLOGIES	Java (Spring Boot framework), RESTful APIs
DATABASE	Firebase Realtime Database
WEB SERVER	Firebase
IDE	Android Studio
TESTING & BUILD TOOLS	JUnit, Espresso for testing, Gradle for build automation



# ***Software Profile***



## **5. SOFTWARE PROFILE**

---

### **5.1 ABOUT ANDROID**

#### **Android:**

Android is an open-source operating system designed for mobile devices, developed by Google. It provides a rich application framework that allows developers to build innovative apps and games for mobile devices. Android supports a wide range of devices, including smartphones, tablets, and wearable devices. The Android SDK (Software Development Kit) includes the tools and APIs necessary to develop applications on the Android platform. The most important components of any Android application include user interface (UI) elements, user interactions, and the ability to connect to web services.

#### **Hardware Interface:**

##### *Client side:*

- ⦿ Processor: Quad-core processor (1.5 GHz or higher).
- ⦿ RAM: 4GB or more.
- ⦿ Storage: Minimum 64GB internal storage
- ⦿ Network Interface: Wi-Fi and mobile data support.

##### *Server Side:*

- ⦿ Processor: Quad-core processor.
- ⦿ RAM: 8GB or more.
- ⦿ Disk space: 2GB.

## **Software Interface**

### **Client side:**

- ⦿ Operating System: Android 6.0 (Marshmallow) or above
- ⦿ Development Environment: Android Studio
- ⦿ APIs: Google Play Services, Firebase SDK
- ⦿ Network Interface: Internet access for API calls and image loading

### **Server Side:**

- ⦿ Web Server: Firebase Hosting or Apache Tomcat (if applicable)
- ⦿ Database: Firebase Realtime Database or Firestore
- ⦿ Frameworks: RESTful API for server-side communication

## **Android Versions Timeline**

### **1. Android 1.0 (September 2008)**

Key Features: The first commercial version of Android included basic features like a web browser, access to the Android Market, and support for Google services. It laid the foundation for future Android developments with its open-source nature.

### **2. Android 1.5 Cupcake (April 2009)**

Key Features: Introduced a virtual keyboard, support for widgets, and the ability to upload videos to YouTube. It marked the beginning of a more user-friendly interface and customization options.

### **3. Android 1.6 Donut (September 2009)**

Key Features: Enhanced Android Market search capabilities and introduced support for different screen sizes and resolutions. It improved the user experience across a wider range of devices.

### **4. Android 2.0 Éclair (October 2009)**

Key Features: Added support for multiple accounts, improved Google Maps, and introduced a new user interface design. The inclusion of Bluetooth 2.1 allowed for better connectivity options.

### **5. Android 2.2 Froyo (May 2010)**

Key Features: Featured speed improvements, support for Flash Player, and the ability to install apps on external storage. Froyo enhanced performance and made the platform more versatile for users.

### **6. Android 2.3 Gingerbread (December 2010)**

Key Features: Improved UI, support for larger screens, and enhanced performance and battery life. Introduced native support for VoIP and NFC technology, paving the way for mobile payments.

### **7. Android 3.0 Honeycomb (February 2011)**

**Key Features:** Specifically designed for tablets, Honeycomb introduced a new UI and improved multitasking capabilities. It brought support for hardware acceleration, making apps more responsive.

### **8. Android 4.0 Ice Cream Sandwich (October 2011)**

**Key Features:** Unified the smartphone and tablet user experience, introducing a more refined UI and features like Face Unlock. Enhanced notifications and improved multitasking made it a significant update.

### **9. Android 4.1 Jelly Bean (July 2012)**

**Key Features:** Introduced Google Now, an intelligent personal assistant, and Project Butter for smoother performance. Enhanced voice search capabilities and notifications provided a more intuitive user experience.

### **10. Android 4.4 KitKat (October 2013)**

**Key Features:** Improved performance on lower-end devices and introduced a more immersive full-screen mode. It added support for NFC and introduced the "OK Google" voice command functionality.

### **11. Android 5.0 Lollipop (November 2014)**

**Key Features:** Introduced Material Design, enhancing the visual appeal of the OS. Improved notifications and battery life management features provided a more efficient user experience.

### **12. Android 6.0 Marshmallow (October 2015)**

**Key Features:** Focused on improving app permissions and introduced Doze mode for better battery management. Enhanced fingerprint recognition support increased device security.

### **13. Android 7.0 Nougat (August 2016)**

**Key Features:** Introduced split-screen multitasking and enhanced notification features. The update also brought improvements to performance and security, making the OS more robust.

### **14. Android 8.0 Oreo (August 2017)**

Key Features: Introduced Picture-in-Picture mode and notification dots for better app management. Enhanced security features and battery optimizations improved overall device performance.

### **15. Android 9.0 Pie (August 2018)**

Key Features: Emphasized user experience through adaptive battery and brightness features. Introduced gesture navigation and digital wellbeing tools to promote healthier device usage.

### **16. Android 10 (September 2019)**

Key Features: Brought a system-wide dark mode and improved privacy controls. Enhanced gesture navigation and better app permissions increased user control over their devices.

### **17. Android 11 (September 2020)**

Key Features: Enhanced messaging features with chat bubbles and improved device control options. Introduced one-time permissions for better security and privacy.

### **18. Android 12 (October 2021)**

Key Features: Introduced Material You, allowing users to personalize their UI with dynamic color themes. Enhanced privacy dashboard gave users more control over their data.

### **19. Android 13 (August 2022)**

Key Features: Focused on personalization, enhancing app language settings, and improving security and privacy options. Introduced new features for better media and photo sharing experiences.

### **20. Android 14 (October 2023)**

Key Features: Enhanced customization options with new themes and widgets. Improved performance and battery optimization, along with further refinements to privacy controls.

## **Characteristics: -**

Android is a versatile and widely-used operating system developed by Google, primarily for mobile devices. It is based on the Linux kernel and designed for touchscreen devices such as smartphones and tablets. The following characteristics highlight the unique features and advantages of the Android platform:

### **1. Open Source**

Android is built on an open-source model, which allows developers and manufacturers to access its source code. This openness encourages innovation, customization, and collaboration within the developer community, leading to a wide variety of applications and device configurations.

### **2. User-Friendly Interface**

Android provides a customizable and user-friendly interface that allows users to personalize their devices with widgets, themes, and various launcher options. This flexibility enhances user experience and accessibility, catering to a diverse range of preferences and needs.

### **3. Rich Application Ecosystem**

The Google Play Store hosts millions of applications, offering users a rich ecosystem of apps for various purposes, from productivity to entertainment. The platform supports third-party app stores, enabling developers to distribute their applications beyond the Google Play Store.

### **4. Multitasking Capabilities**

Android supports multitasking, allowing users to switch between applications seamlessly. Features such as split-screen mode and picture-in-picture enhance productivity by enabling users to view multiple apps simultaneously.

### **5. Wide Hardware Compatibility**

Android is designed to work across a wide range of hardware devices, from low-end smartphones to high-performance tablets. This compatibility allows manufacturers to create a diverse array of devices catering to different market segments and user preferences.

### **6. Customizable User Experience**

Users can modify their Android devices extensively, including changing the home screen layout, installing custom ROMs, and using various launchers. This customization fosters a unique experience tailored to individual user preferences.

## **7. Regular Updates and Improvements**

Android receives regular updates, introducing new features, security enhancements, and performance improvements. Google releases major Android versions annually, while security patches are provided monthly to ensure device safety.

## **8. Google Services Integration**

Android devices come pre-loaded with Google services, including Google Assistant, Google Maps, and Google Drive, providing users with powerful tools for productivity and navigation. This integration enhances the overall functionality of the platform.

## **9. Support for Various File Formats**

Android supports a wide range of file formats, including multimedia, documents, and archives. This flexibility allows users to manage and share content easily across different applications and platforms.

## **10. Security Features**

Android incorporates multiple security features, including biometric authentication, encrypted storage, and Google Play Protect, which scans apps for malware. These features help safeguard user data and maintain privacy.

## **11. Community Support**

The large and active Android community offers extensive resources, tutorials, and forums for users and developers. This support network fosters learning and troubleshooting, making it easier for individuals to navigate the platform.

## **5.2 CLIENT-SIDE TECHNOLOGIES**

Client-side technologies refer to the tools and languages that operate within the user's device, managing the interaction between users and applications. In Android development, client-side technologies ensure seamless user interface design, functionality, and performance. Below are key client-side technologies and their roles in Android applications:

### **1. Java/Kotlin**

Java was the official language for Android development until Kotlin was introduced as a preferred language in 2017. These programming languages are used for building the logic and structure of Android apps.

Java: A general-purpose, object-oriented programming language that has been the foundation of Android app development for years. Java provides rich libraries, tools, and frameworks that allow developers to create powerful, robust apps.

Kotlin: Kotlin is a modern, concise, and safe programming language that reduces boilerplate code and enhances readability. It seamlessly integrates with Java, offering developers flexibility in app development.

### **2. XML (Extensible Markup Language)**

XML is used for designing the user interface and defining layouts, views, and resources in Android apps. Android uses XML to declare UI components, and it serves as the primary language for defining layouts and visual elements.

Role of XML: XML allows developers to create structured layouts using elements like buttons, text views, and images. These XML layouts are converted into code, which helps manage the user interface seamlessly across different devices and screen sizes.

### **3. Android UI Frameworks (Jetpack Compose)**

Android UI frameworks allow developers to design interactive and responsive user interfaces. The most recent framework, Jetpack Compose, is a modern toolkit for building native UIs in Android applications.



**Jetpack Compose:** This UI framework simplifies and accelerates UI development with Kotlin. Developers can create UIs declaratively, meaning they describe how the UI should look in response to state changes, reducing code complexity and enhancing app performance.

#### **4. HTML5 and WebView**

Android applications sometimes require web content to be displayed within the app, which is achieved using WebView. WebView enables apps to load HTML content or even a full website inside an application.

**HTML5:** HTML5 is the latest version of HTML and is widely used to create web-based content that can be embedded in Android apps through WebView. It supports multimedia elements like video, audio, and canvas, making it a versatile option for integrating web content in apps.

#### **5. JavaScript**

JavaScript is a lightweight, dynamic scripting language used in conjunction with HTML and WebView to add interactivity to web content in Android applications. JavaScript can control the behavior of embedded web content.

**Use in Android:** JavaScript is executed within WebView to manipulate dynamic content and handle user interactions. It enables rich features such as form validation, real-time updates, and client-side computations in web-based parts of Android applications.

#### **6. CSS (Cascading Style Sheets)**

CSS is used to style and layout web content embedded in Android apps using WebView. It enhances the visual presentation of HTML content by defining how elements like text, buttons, and images are displayed on the screen.

**Role in Android Apps:** CSS allows developers to create responsive and adaptive designs for web-based content embedded in Android apps. It provides control over the layout, color schemes, fonts, and positioning of elements.

#### **7. JSON (JavaScript Object Notation)**

JSON is a lightweight data-interchange format that is easy for humans to read and write and easy for machines to parse and generate. It is widely used in client-server communication in Android apps, especially when fetching or sending data from APIs.

Use in Android: JSON is commonly used in Android apps to structure data exchanged between the app and a backend server. It simplifies the process of data handling, making it easier to process and display content in Android apps.

## **5.3 DATABASE**

Databases are essential components of modern Android applications, enabling efficient data storage, retrieval, and manipulation. Android provides several built-in and external solutions for database management, with the most widely used being SQLite and Room Persistence Library. These database technologies are critical for managing app data, ensuring persistence across sessions, and optimizing the user experience by providing quick access to large amounts of structured data.

### **1. SQLite**

SQLite is a self-contained, high-reliability, embedded, and lightweight database engine used in Android for storing and retrieving structured data. Since it is embedded within the Android operating system, SQLite requires no external configuration or installation, making it the default choice for local database storage in Android apps. Here's an overview of SQLite's key features and usage in Android:

- **Lightweight and Fast:** SQLite is designed to be lightweight with minimal setup, using only a small footprint of the device's resources. It efficiently handles smaller databases, making it ideal for mobile applications where processing power and storage may be limited.
- **Relational Database Model:** Like other relational database systems, SQLite uses SQL (Structured Query Language) for interacting with data. This means developers can create, read, update, and delete (CRUD) records using familiar SQL queries.

- **Local Storage Solution:** SQLite stores data locally on the device, allowing apps to function even without network access. This is beneficial for offline data storage, such as caching user preferences, saving user-generated content, or logging app usage statistics.
- **No Setup Required:** One of SQLite's major advantages is that it requires no setup or server configuration. It runs as part of the app and stores the data in files within the app's directory.
- **Security:** Android apps can encrypt SQLite databases using external libraries or the Android Keystore system, enhancing data security for sensitive information like user credentials or financial records.

### **Working with SQLite in Android**

Developers directly interact with SQLite databases using the SQLiteOpenHelper class, which provides methods to create, manage, and update the database. The following are typical steps in working with SQLite:

1. **Creating a Database:** Developers extend the SQLiteOpenHelper class and override the onCreate() method to define the database schema, including tables, columns, and data types.
2. **Inserting Data:** Using the insert() method, apps can add new records to the database, with data represented in a ContentValues object.
3. **Querying Data:** Queries are made using SQL commands, either directly through the.rawQuery() method or using the query builder for easier filtering and sorting.
4. **Updating and Deleting Data:** The update() and delete() methods allow apps to modify or remove records from the database based on specific conditions.

Despite being a powerful tool, SQLite can become cumbersome for large or complex databases, especially when managing relationships between data, such as foreign keys or performing complex joins between multiple tables. To address this, Android introduced the Room Persistence Library as a more user-friendly and efficient way of managing databases.

## **2. Room Persistence Library**

The Room Persistence Library is an abstraction layer over SQLite that simplifies database management in Android applications. It is part of Android's Jetpack architecture components, introduced to help developers handle SQLite databases with fewer errors and more structured code. Room is designed to work seamlessly with the Android architecture components, such as ViewModel and LiveData, ensuring data persistence across app lifecycles.

- **Simplified SQL Operations:** While SQLite requires developers to write raw SQL queries, Room abstracts much of this complexity. Instead of manually writing queries, developers define database interactions using annotated data access objects (DAOs). This significantly reduces the likelihood of errors in query syntax and makes the code cleaner and more readable.
- **Entity-Relationship Mapping:** Room allows developers to define entities as simple POJO (Plain Old Java Object) classes. These entities map directly to the SQLite tables, where each field in the POJO represents a column in the table. Room automatically handles relationships between tables, such as one-to-many or many-to-many relationships, through annotations, eliminating the need for complex SQL joins.
- **Compile-Time Validation:** One of Room's standout features is its compile-time verification of SQL queries. Any error in query syntax, schema mismatches, or incorrect field names will be caught during compilation, helping developers identify issues early in the development process.
- **Lifecycle-Aware Data Access:** Room integrates with Android's lifecycle components, enabling reactive data handling. For example, developers can return data wrapped in LiveData or Flow, allowing the UI to automatically update when data changes. This

helps keep the app responsive and ensures that it adheres to Android's best practices for managing app state and memory.

### Room Components

1. Entities: Entities represent the tables in the SQLite database. Each entity is a class annotated with `@Entity`, where fields represent the columns, and the primary key is defined using `@PrimaryKey`. Example:

*@Entity*

```
public class User {  
    @PrimaryKey  
    public int id;  
    public String name;  
    public int age;  
}
```

2. Data Access Objects (DAOs): DAOs are interfaces that define the methods for interacting with the database. They are annotated with `@Dao` and contain methods like `@Query`, `@Insert`, `@Update`, and `@Delete`. Example:

*@Dao*

```
public interface UserDao {  
    @Insert  
    void insertUser(User user);  
  
    @Query("SELECT * FROM User WHERE id = :userId")  
    User getUser(int userId);  
}
```

3. Database Class: The database class holds the database configuration and serves as the main access point for the connection to the database. It is annotated with `@Database`, and lists all entities and DAOs associated with the database. Example:

```
@Database(entities = {User.class}, version = 1)
public abstract class AppDatabase extends RoomDatabase {
    public abstract UserDao userDao();
}
```

### **Advantages of Room**

- **Structured Code:** Room encourages developers to follow best practices for data management by enforcing the separation of concerns between data storage (DAO) and business logic.
- **Better Performance:** Room optimizes database transactions and queries, often outperforming raw SQLite implementations in larger or more complex apps.
- **Easy Testing:** Room provides support for both unit testing and integration testing. The Room database can be easily tested using in-memory databases, ensuring that no actual app data is modified during tests.

### **Challenges with Room**

- **Learning Curve:** For developers accustomed to working directly with SQLite, the Room framework might require some adjustment, especially in terms of managing annotations and relationships between entities.
- **Database Migration:** When the schema of a Room database changes, developers need to define migration strategies to preserve data integrity. This can be challenging when dealing with large or complex databases.

## **5.4 NETWORKING WITH ANDROID**

Networking is a crucial aspect of Android development, as many modern apps interact with remote servers to fetch or store data. Whether it's retrieving user data from a backend service, downloading images, or interacting with third-party APIs, Android provides several libraries to make network operations more efficient and reliable. Two widely-used libraries for networking in Android are Retrofit and Volley. Each of these libraries simplifies different aspects of making HTTP requests and handling responses, with distinct features tailored to various use cases.

### **1. Retrofit**

Retrofit is a type-safe HTTP client developed by Square for Android, which simplifies the process of making network requests and consuming RESTful web services. It is designed to handle complex networking operations, providing a robust and flexible framework for interacting with APIs. Retrofit abstracts much of the boilerplate code associated with networking and offers an intuitive way to connect Android apps to external APIs.

#### **Key Features of Retrofit**

- **Type-Safe HTTP Requests:** Retrofit uses Java annotations to define API endpoints, making HTTP requests type-safe. This means that the structure of requests and responses is strictly defined, reducing the chances of runtime errors. Developers can define API requests using standard Java interfaces and Retrofit will automatically generate the necessary HTTP calls.
- **Flexible Data Parsing:** Retrofit supports multiple data formats for responses, including JSON, XML, and protocol buffers. One of its most popular features is its seamless integration with Gson, a library that automatically converts JSON responses into Java objects. This eliminates the need for manual parsing and significantly speeds up the development process.

#### **How Retrofit Works**

Retrofit works by creating a **REST** client that defines the API endpoints and how they should be accessed. The core components of Retrofit are:

- I. **Service Interface:** This is a Java interface that contains methods annotated with HTTP verbs (such as GET, POST, PUT, DELETE, etc.) and the corresponding API endpoints. Parameters for queries or request bodies are specified using annotations such as @Query, @Body, or @Path.

```
public interface ApiService {  
  
    @GET("users/{id}")  
    Call<User> getUser(@Path("id") int userId);  
  
    @POST("users/new")  
    Call<User> createUser(@Body User user);  
  
}
```

- II. **Retrofit Builder:** This is where you configure Retrofit by setting the base URL and optionally adding a converter factory (such as Gson) to handle responses.

```
Retrofit retrofit = new Retrofit.Builder()  
    .baseUrl("https://api.example.com/")  
    .addConverterFactory(GsonConverterFactory.create())  
    .build();
```

- III. **Making API Calls:** Once the interface and builder are set up, you can create an instance of the API service and invoke the defined methods to make network requests.

```
ApiService service = retrofit.create(ApiService.class);  
  
Call<User> call = service.getUser(1);  
  
call.enqueue(new Callback<User>() {  
  
    @Override  
    public void onResponse(Call<User> call, Response<User> response) {  
        // Handle the response  
    }  
})
```



```
@Override  
  
public void onFailure(Call<User> call, Throwable t) {  
  
    // Handle failure  
  
}  
  
});
```

## Benefits of Using Retrofit

- **Asynchronous Requests:** Retrofit handles both synchronous and asynchronous HTTP requests. Using callbacks, developers can execute asynchronous requests to prevent blocking the main thread, improving app performance.
- **Modular and Extensible:** Retrofit is highly extensible and can be easily integrated with other libraries like OkHttp (for advanced networking) and RxJava (for reactive programming). This modularity makes it ideal for complex applications that require sophisticated network handling.
- **Built-in Error Handling:** Retrofit includes built-in error handling mechanisms, such as status code checking and handling network timeouts, which can simplify the process of dealing with API failures.

## Use Cases

- **REST API Integration:** Retrofit is particularly suited for apps that need to interact with RESTful APIs, which typically return JSON or XML responses.
- **Complex Queries and Form Submissions:** Retrofit simplifies handling of complex API requests, including multi-part file uploads, form submissions, and URL query parameter mapping.

## 2. Volley

Volley is another networking library for Android, developed by Google, that simplifies the process of making network requests, particularly when it comes to image loading, caching, and

handling multiple concurrent requests. While Retrofit is more focused on API consumption, Volley provides a more general-purpose solution for network operations and is particularly effective in handling real-time data and image requests.

- **Optimized for Real-Time Requests:** Volley is particularly good for real-time network requests and responses, such as updating user data or displaying dynamic content. It queues and manages multiple network requests, prioritizing those that need faster responses.
- **Inbuilt Caching and Retry Mechanism:** Volley has a built-in caching mechanism that stores network responses locally to minimize redundant network calls. It also supports automatic retry strategies for failed requests, including exponential backoff.

### **Key Features of Volley**

1. **Simplified HTTP Requests:** Volley supports various HTTP request types, such as GET, POST, PUT, and DELETE, with simple API calls. Developers can use `JsonObjectRequest` or `StringRequest` to easily manage JSON and string data.
2. **Image Loading:** Volley provides optimized tools for loading images from a URL and displaying them in `ImageView` widgets. The `ImageLoader` class simplifies image caching and memory management, ensuring that images are loaded efficiently without overloading the UI thread.
3. **Request Queues:** Volley automatically manages a queue for handling multiple network requests simultaneously. This makes it easy to manage priority levels for requests, cancel requests, or batch requests.
4. **Built-In Caching:** One of Volley's standout features is its inbuilt response caching system, which caches network responses based on headers. This is especially useful for apps that fetch frequently updated data, like social media feeds or news apps.

5. **Retry and Timeout Handling:** Volley supports customizable retry policies. It automatically retries failed requests with exponential backoff and timeout periods. This helps in managing unreliable network conditions and ensuring that important requests are not dropped prematurely.

### **Advantages of Volley**

- **Real-Time Data:** Volley is well-suited for apps that require real-time updates, such as chat apps, live feeds, or dynamic content updates.
- **Efficient Image Loading:** Its built-in image loading and caching features make it a great choice for media-rich applications, such as apps that rely heavily on image galleries, product catalogs, or social media feeds.
- **Automatic Caching and Queueing:** Volley's automatic response caching and request queuing make it a reliable solution for managing multiple concurrent network requests, improving the performance of data-driven applications.

### **Limitations of Volley**

- **Less Suitable for Large File Uploads:** While Volley is optimized for real-time and small-sized data requests, it is not the best choice for handling large payloads, such as file uploads or streaming content. In such cases, Retrofit or OkHttp would be more efficient.
- **Verbose Syntax:** While powerful, Volley can be more verbose and less flexible than Retrofit when it comes to complex API requests, such as handling pagination, custom converters, or advanced response handling.

## **5.5 DEPENDENCY INJECTION**

Dependency injection (DI) is a design pattern used to manage and provide dependencies in an application, which makes code more modular, reusable, and testable. In Android development, dependency injection is often implemented using frameworks like Dagger or Hilt.

### **Dagger**

Dagger is a fully static, compile-time dependency injection library for Java and Android. It helps manage the complex web of dependencies by generating code that facilitates the creation of objects. Dagger can handle both singleton objects that persist throughout the app's lifecycle and scoped objects that live only as long as the activity or fragment.

#### **Key Features of Dagger:**

1. **Compile-Time Injection:** Dagger generates code at compile time, which is more efficient than runtime injection and helps detect errors early.
2. **Modular and Scalable:** With Dagger, you can manage complex object graphs, making the code base more modular and maintainable as the app grows.

### **Hilt**

Hilt is a simplified version of Dagger tailored for Android. It integrates seamlessly with Android components like Activities, Fragments, and ViewModels, reducing boilerplate code and simplifying the process of injecting dependencies across the app.

#### **Key Features of Hilt:**

1. **Ease of Use:** Hilt automates much of Dagger's setup and integrates with Android lifecycle components.
2. **Scoped Containers:** Hilt manages the scope of dependencies, such as limiting the lifetime of an object to a specific activity or fragment, ensuring memory efficiency.

Both Dagger and Hilt improve code maintainability, making it easier to manage dependencies, especially in large-scale applications where testing, scaling, and reducing boilerplate code are essential.

## **5.6 USER INTERFACE**

### **RecyclerView**

RecyclerView is a more advanced and flexible view for displaying large sets of data compared to its predecessor, ListView. It is an essential part of Android's UI toolkit and is optimized for displaying long lists or grids of data efficiently by reusing views that are no longer visible.

Key Features of RecyclerView:

1. **ViewHolder Pattern:** RecyclerView uses the ViewHolder pattern to improve performance by reusing views that scroll off the screen, reducing the overhead of inflating new views unnecessarily.
2. **LayoutManager:** RecyclerView supports different layouts (e.g., linear, grid, or custom layouts) through its LayoutManager, making it adaptable to various use cases.

With its flexible architecture, RecyclerView is a go-to UI component for displaying dynamic data in a scrollable list or grid, offering smooth performance even for large data sets.

### **Material Design**

Material Design is a comprehensive design language developed by Google that aims to create consistent, visually appealing, and intuitive user interfaces across platforms and devices. It emphasizes clean, bold, and tactile visual elements that provide meaningful feedback to users.

Key Features of Material Design:

1. **Visual Hierarchy:** Material Design uses shadows, layers, and bold colors to create a clear visual hierarchy, helping users understand the relationships between UI elements.
2. **Cross-Platform Consistency:** Material Design ensures consistency across Android, iOS, and the web, ensuring a unified user experience across different devices.

Material Design components are widely integrated into Android's UI framework, offering a modern, consistent look and feel, with tools like Material Components for easier implementation.

## **5.7 SECURITY**

### **Firestore Authentication**

Firestore Authentication is a service provided by Google's Firestore platform that allows developers to implement user authentication in Android applications easily and securely. It supports multiple authentication methods, including email/password, phone number, and third-party providers like Google, Facebook, and GitHub.

#### **Key Features of Firestore Authentication:**

1. Multiple Authentication Methods: Firestore Authentication supports various login mechanisms, including OAuth-based social logins (e.g., Google and Facebook), phone authentication, and email/password, making it versatile for different app requirements.
2. Secure Authentication: Firestore handles secure token generation and session management, ensuring that authentication processes are robust and secure against common security threats like session hijacking.

Firestore Authentication simplifies the process of adding secure user authentication to Android apps, offering built-in security features and reducing the burden on developers to manage user credentials.

## **5.8 ANALYTICS**

### **Google Analytics/Firebase Analytics**

Google Analytics and Firebase Analytics are tools that help track and analyze user behavior within Android applications, offering insights into app performance and user engagement. While Google Analytics is widely used for web tracking, Firebase Analytics is specifically designed for mobile applications.

#### **Key Features of Firebase Analytics:**

1. Event Tracking: Firebase Analytics automatically tracks user events (e.g., screen views, button clicks) and custom events, helping developers understand how users interact with the app and identify areas for improvement.

2. Real-Time Data: Firebase provides real-time analytics, giving developers up-to-the-minute insights into user behavior, session duration, and app engagement.

By integrating Firebase Analytics into an Android app, developers can make data-driven decisions to enhance user experience, optimize app performance, and increase user retention through targeted insights.



# ***System Design***



## 6. SYSTEM DESIGN

---

### 6.1 DATABASE SCHEMA DESIGN FOR FIREBASE:

The database schema for "BHOJAN SAAJHA" includes tables for **Login**, **Users**, **Admin**, **Receiver**, **Donate**, **Contact**, and **Food Map**. The **Login** table contains user ID, username, password, and role. The **Users** table stores user ID, name, email, and contact info, while the **Admin** table manages admin users. The **Receiver** table holds recipient information, and the **Donate** table logs food donations, including donation ID, user ID (foreign key), food description, quantity, and location ID (foreign key). The **Contact** table records inquiries, and the **Food Map** table captures location details. This schema ensures efficient data management for the application.

#### Collection: Users

- **userId**: Unique identifier for the user (auto-generated by Firebase).
- **userName**: Name of the user (either donor or recipient).
- **userEmail**: Email address of the user.
- **userPhone**: Contact number for the user.
- **userRole**: Role of the user (Donor, Receiver, or Admin).
- **userLocation**: Location of the user (city, state, or address).

#### Collection: Food Donations

- **donationId**: Unique identifier for each food donation (auto-generated).
- **donorId**: References the user ID of the donor (foreign key reference to Users collection).
- **foodDescription**: Detailed description of the donated food items.
- **foodQuantity**: Quantity of the food donated.
- **donationLocation**: Location of the donation (linked to the Food Map).
- **donationTimestamp**: Timestamp indicating when the food was donated.

#### Collection: Food Map

- **locationId**: Unique identifier for each location (auto-generated).
- **locationName**: Name of the location (e.g., restaurant, community center).

- **locationAddress:** Full address of the location.
- **latitude:** Latitude coordinate for map integration.
- **longitude:** Longitude coordinate for map integration.

### **Collection: Contacts**

- **contactId:** Unique identifier for each inquiry (auto-generated).
- **userId:** References the user ID of the person making the inquiry (foreign key reference to Users collection).
- **contactMessage:** Message or inquiry submitted by the user.
- **contactTimestamp:** Timestamp indicating when the inquiry was made.

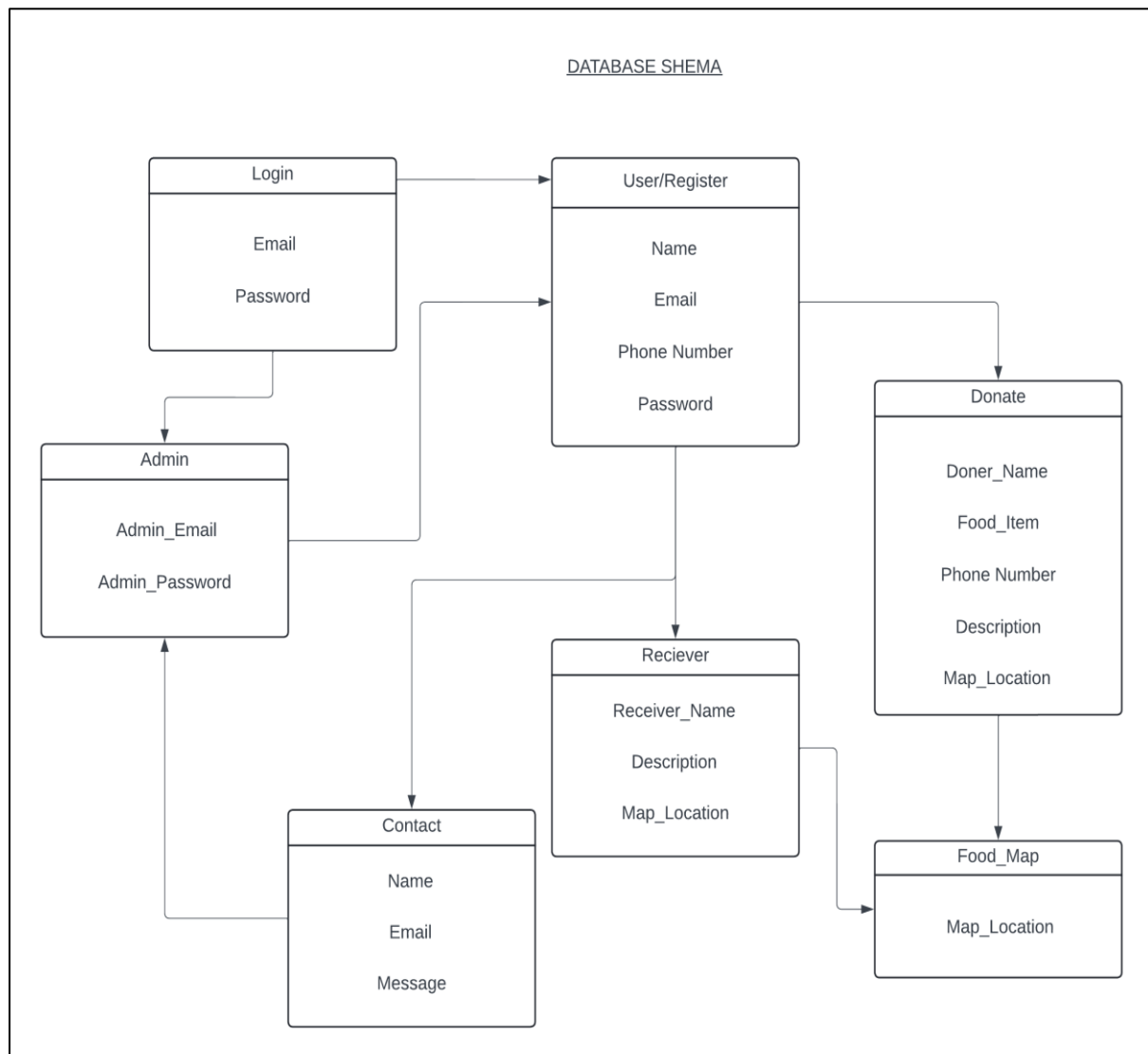
### **Collection: Notifications**

- **notificationId:** Unique identifier for each notification (auto-generated).
- **userId:** References the user ID of the recipient of the notification.
- **notificationMessage:** Content of the notification (e.g., updates on food availability).
- **notificationTimestamp:** Timestamp indicating when the notification was sent.

### **Key Points for Firebase Design:**

- **NoSQL Structure:** The design leverages Firebase's NoSQL capabilities, allowing for flexible data modeling without strict relationships like foreign keys.
- **Real-time Updates:** The app can utilize Firebase's real-time capabilities to push notifications and updates to users instantly when new food donations are available or inquiries are responded to.
- **User-centric Data:** Data is organized around users (donors and receivers), ensuring easy access and management of information related to food donations and inquiries.

**Scalability:** The structure can easily scale as the number of users and donations grows, allowing for efficient handling of data without performance degradation.



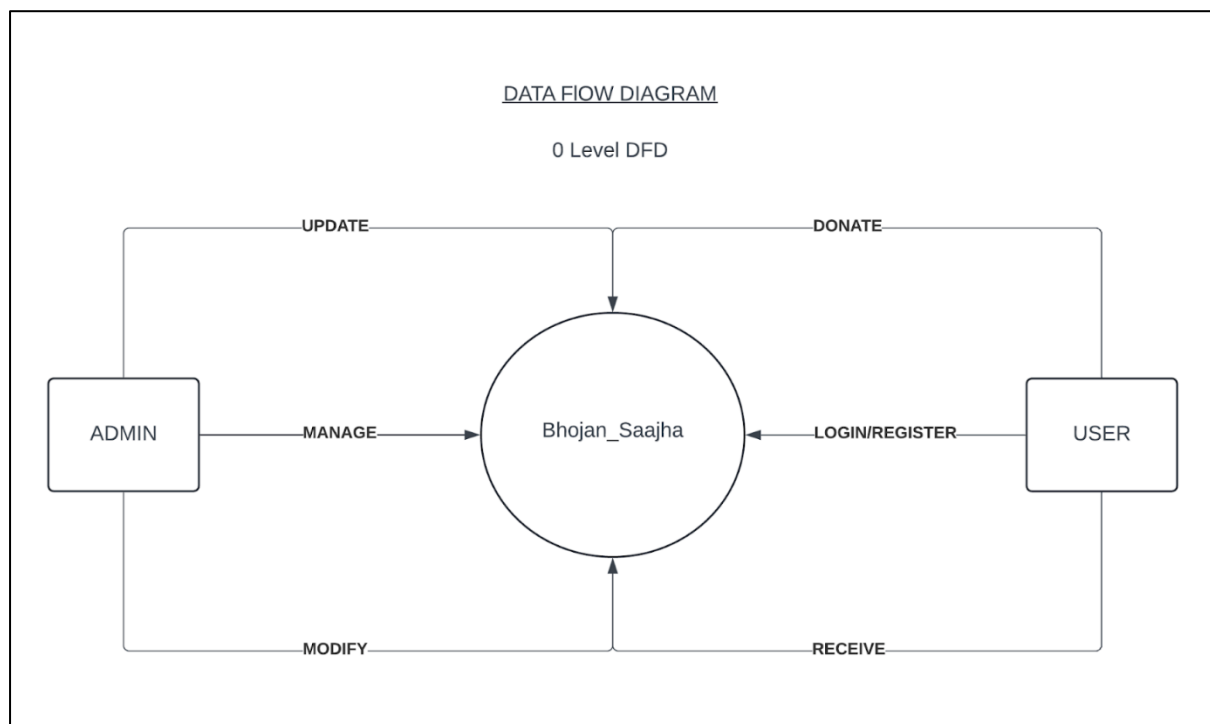
## 6.2 DATA FLOW DIAGRAM

### Data Flow diagram:

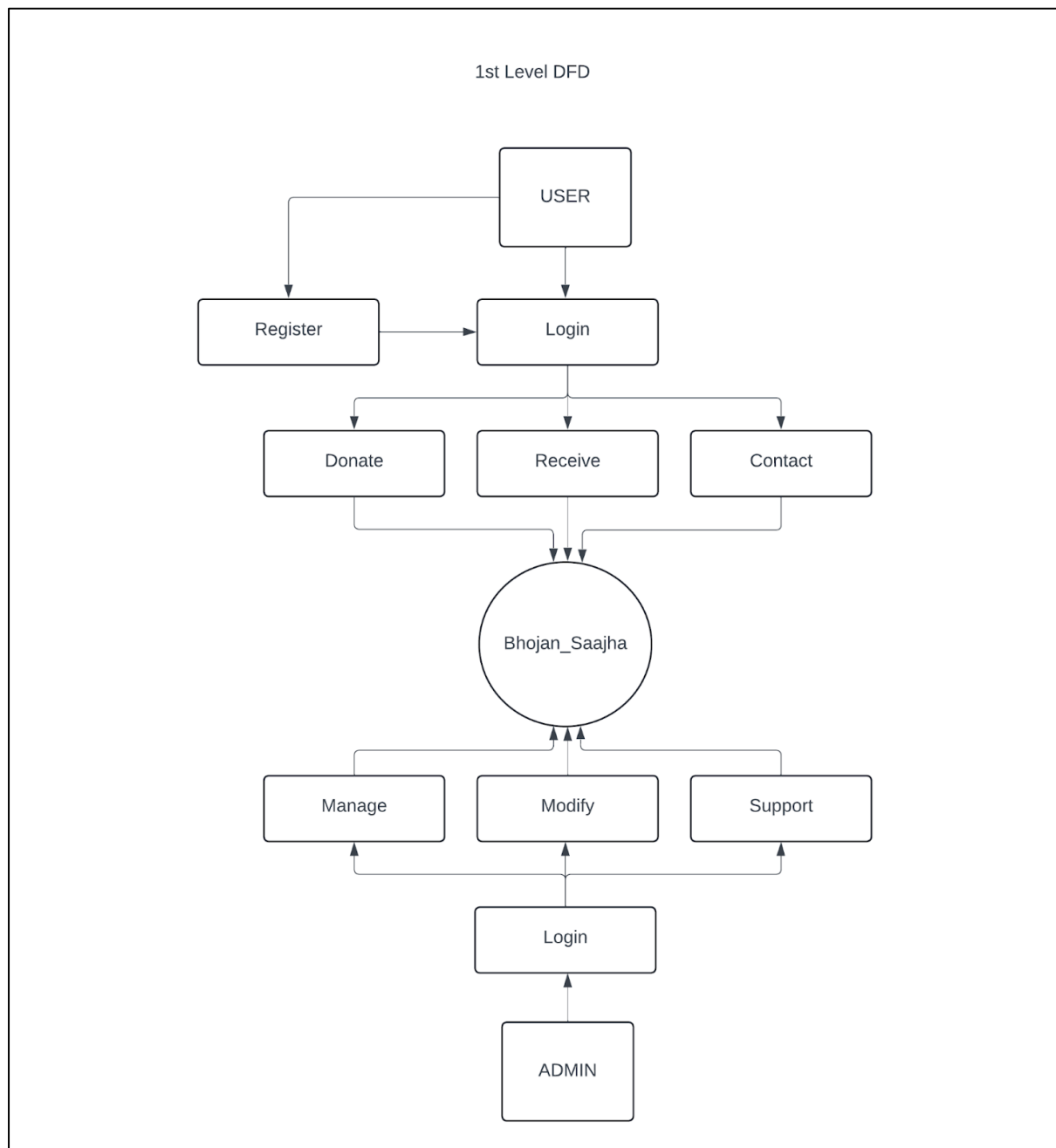
The Data Flow Diagram (DFD) for "BHOJAN SAAJHA" is crucial for illustrating how data flows within the application. It provides clarity in data management by identifying relationships between components like user profiles, food listings, and notifications. This visualization enables stakeholders, including developers and project managers, to understand the system's processes better, fostering effective collaboration during development.

Additionally, the DFD outlines inputs, processes, and outputs, laying the groundwork for further analysis and design refinement. By mapping data flows, developers can optimize data handling, enhance user experience, and ensure compliance with relevant regulations. Ultimately, this comprehensive understanding of data interactions supports the development of a user-friendly platform that connects food donors with those in need, promoting food security and reducing waste in the community.

### LEVEL 0:

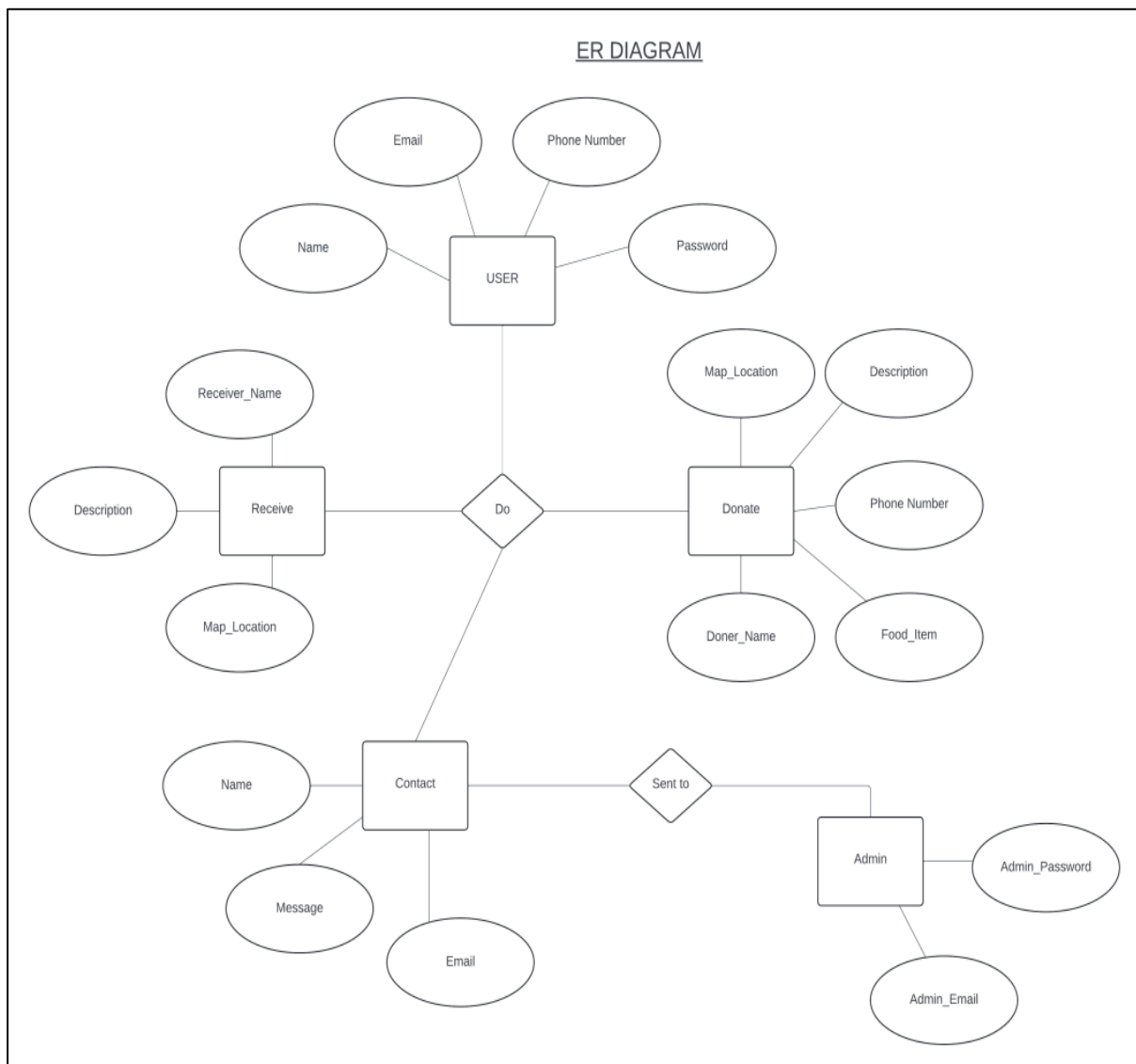


**LEVEL 1:**



### 6.3 ER DIAGRAM

The Entity-Relationship (ER) Diagram for "BHOJAN SAAJHA" outlines key entities and their relationships, including Users (donors), Receivers (individuals in need), Donations, Contacts, and Admins. Each user can create multiple donations and establish contact with receivers, while admins oversee the entire process to ensure smooth operations. This diagram clarifies data relationships and guides the database design.





# ***System Testing***

## **7. TESTING**

### **7.1 CODING , TESTING AND IMPLEMENTATION**

The main interfaces are:

- Application server and Database server interface.

Check if all the interactions between these servers are executed properly. Errors are handled properly. If a database server returns any error message for any query by application server then application server should catch and display these error messages appropriately to users. Check what happens if a user interrupts any transaction in-between? Check what happens if the connection to the web server is reset in between?

#### **Implementation Steps:**

- Ensure Firebase is correctly initialized in your app's Application class or onCreate() method of the first activity.
- Create a mock Firebase database for testing environments to prevent altering production data.
- Test fetching and updating data through the repository classes (DealsRepository and CheckUpdate).

#### **Testing:**

Unit Testing:

- Mock Firebase API responses using libraries like Mockito.
- Check if the correct callbacks (onSuccess/onFailure) are invoked for fetching deals.
- Simulate database errors to validate the handling of failures (e.g., no internet, Firebase fetch error).

Integration Testing:

- Test the flow from the database to the view (RecyclerView) using Espresso to ensure the UI displays data correctly.
- Verify that the RealtimeDeals fragment displays a loading spinner and handles empty responses correctly.



**Errors:**

- No Internet Connection: Ensure Snackbar for retry appears when there's no network connectivity.
- Firebase Fetch Failure: Ensure the app logs the error using Log.e(TAG, errorMessage) and displays an appropriate message via Toast to the user.
- Null or Empty Data: Handle cases where the data may not exist in Firebase (empty lists or null references).

## **7.2 TESTING AND ERRORS**

Compatibility of our Application is a very important testing aspect. See which compatibility test to be executed:

- Android Device compatibility
- Operating system compatibility

**Android Device compatibility:**

Android device compatibility is crucial for the "BHOJAN SAAJHA" application to ensure it functions effectively across a diverse range of smartphones and tablets. Testing will focus on various screen sizes, hardware specifications, and resolutions to guarantee a consistent user experience. This includes verifying that the app's layout, features, and performance are optimized for different devices, enabling users to access food donation services seamlessly.

**OS compatibility:**

Operating system compatibility is vital for the "BHOJAN SAAJHA" application to function correctly on different versions of the Android platform. The application will be tested on multiple Android versions to ensure that all features and functionalities perform as intended, regardless of the user's device. This testing will help identify any version-specific issues and ensure that users have a reliable and consistent experience, promoting accessibility for all potential users of the app.

### **Implementation Steps:**

- Use a GridLayoutManager with item decorations for spacing.
- Ensure that the DealsAdapter populates each view holder correctly, especially for images and text views.
- Lazy-load images using Picasso or any other image loader library.

### **Testing:**

#### **Unit Testing:**

- Mock the DealsModel and verify that the correct data is set in each view holder (ImageView, TextView, etc.).
- Validate that image loading uses fallback URLs when the provided image URL is invalid or empty.

#### **UI Testing:**

- Use Espresso to scroll through the RecyclerView and verify that all items are loaded and displayed.
- Test dynamic data addition by simulating API calls and verifying that RecyclerView updates when new data is appended using addDeals().

### **Errors:**

- Null Images or Invalid URLs: Ensure placeholder images are shown when URLs are broken or missing.
- RecyclerView Overlapping: Check if spacing between grid items is maintained properly using GridSpacingItemDecoration.

### 7.3 SAMPLE TEST CASES DONE

Sl. No.	Test Case Done	Action taken
1	User tries to log in with invalid credentials	A pop-up message appears in the app, notifying the user of invalid credentials and prompting them to try again.
2	User submits a food donation form without filling required fields	A pop-up message is displayed on the specific incomplete fields, prompting users to complete them before proceeding.
3	User enters invalid data in the quantity field (e.g., negative values)	A pop-up message appears on the specific field, indicating that the quantity must be a positive number.
4	User attempts to access a food listing with an invalid ID	The application captures the error and prompts the user to enter a valid food listing ID.
5	User selects an incorrect location for food donation	An error message is displayed, notifying the user to select a valid location from the provided options.



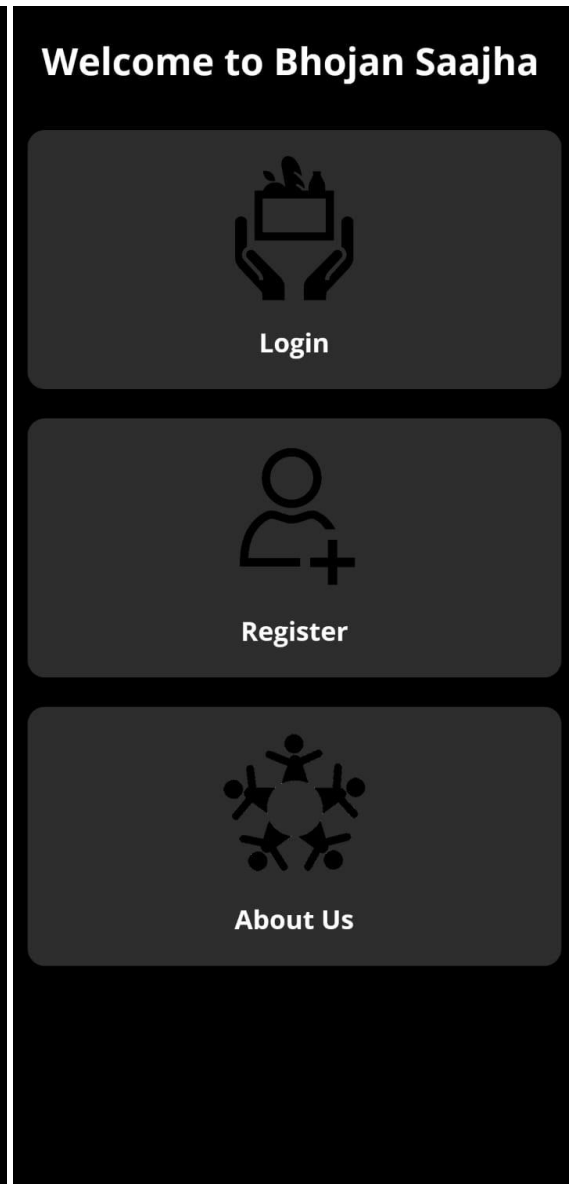
# ***Screenshots***

## 8. SCREENS

Splash Screen:




Home Screen:



Register Screen:

7:18 0.5 KB/s 94%



Name


---

Email

---

Phone Number

---

Password 


---

**REGISTER**

Already Registered?


Login Screen:

7:19 0.2 KB/s 94%



Email

---

Password 









---

**LOGIN**

Not Registered! Click here to Register

Dashboard Screen:

### My Dashboard

 <b>Donate</b>	 <b>Receive</b>
 <b>Food Map</b>	 <b>My Pins</b>
 <b>History</b>	 <b>About Us</b>
 <b>Contact Us</b>	 <b>Log Out</b>

Donate Screen:

### Donate

**Donor Name**

---

**Food Items**

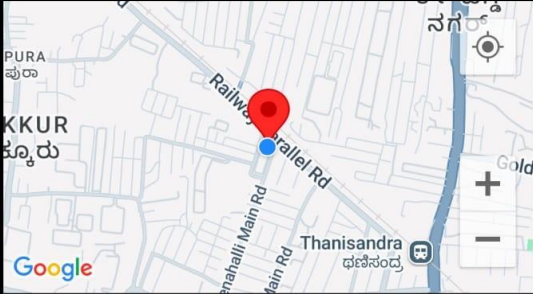
---

**Phone Number**

---

**Description**

---



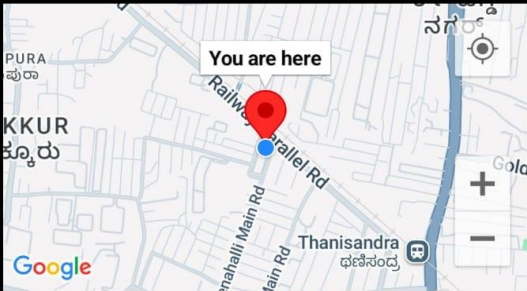
**SUBMIT**

**Receive Screen:**

Receive

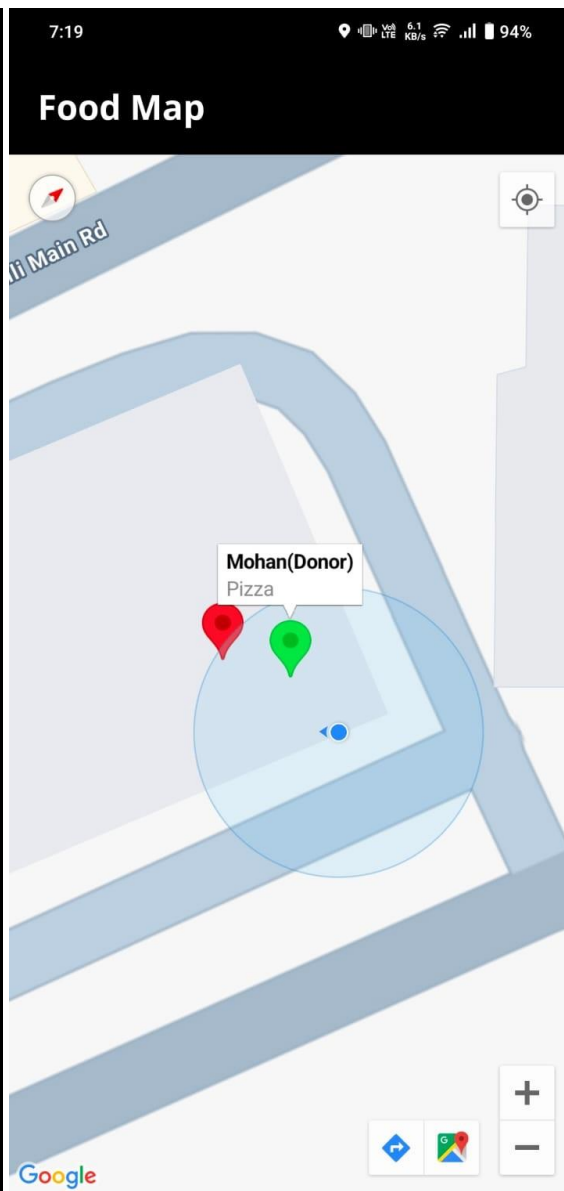
Receiver Name

Description



SUBMIT

**Food Map Screen:**





## History Screen:

### History

☐ Select All

Name: Mohan

Description: Pizza

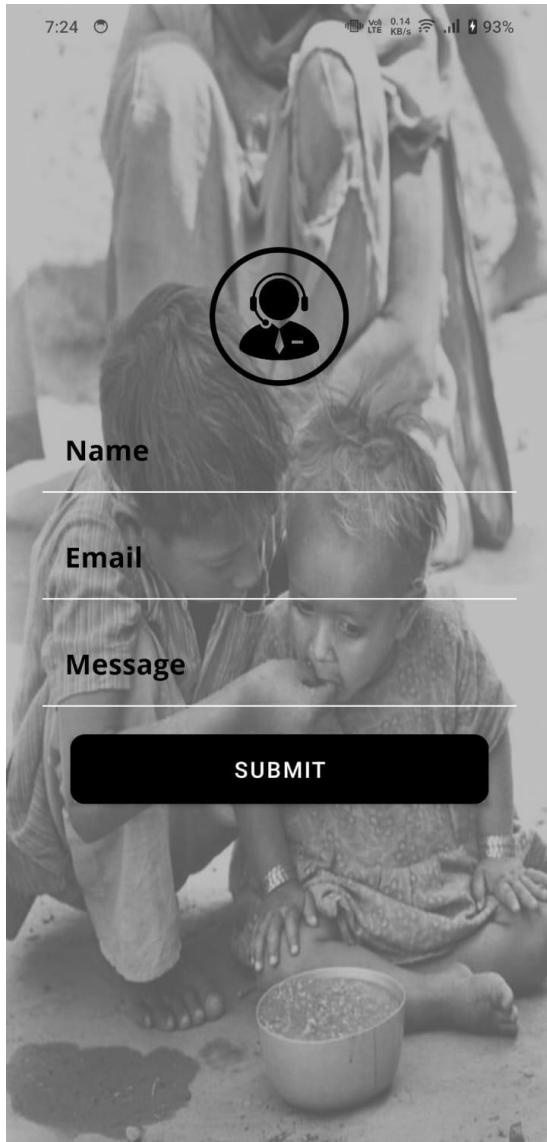
Date & Time: 2024-10-03 22:39:46

Delete


## My Pins Screen:

### My Pins

**Contact Screen:**



7:24 0.14 KB/s 93%



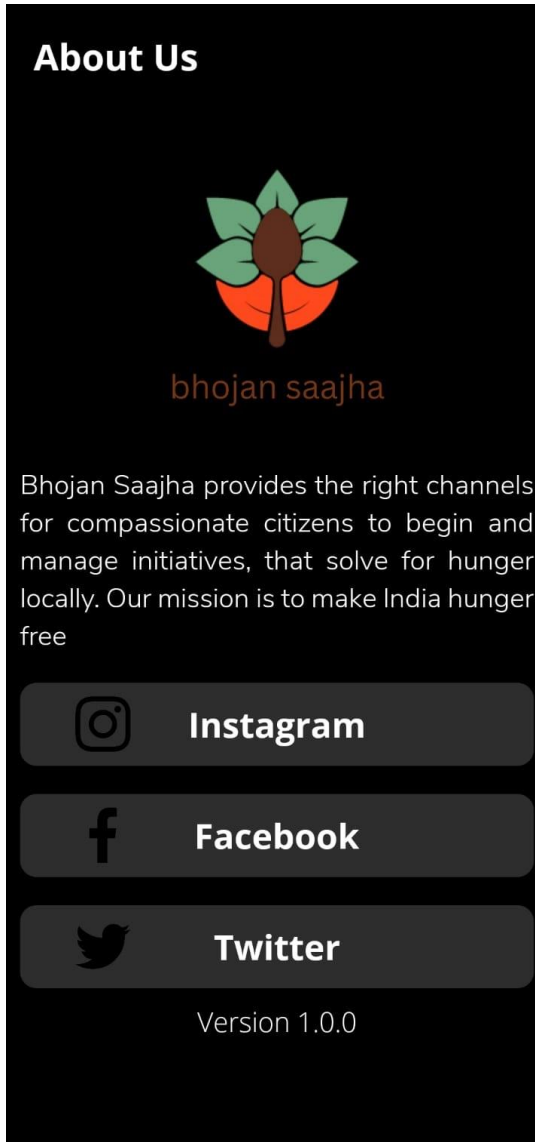
**Name**

**Email**


**Message**

**SUBMIT**

**About Us Screen:**





**About Us**




**bhojan saajha**

Bhojan Saajha provides the right channels for compassionate citizens to begin and manage initiatives, that solve for hunger locally. Our mission is to make India hunger free

 **Instagram**

 **Facebook**

 **Twitter**

Version 1.0.0



# ***Coding***

## Splash Screen:

### Java

```
package com.paurush.bhojan_saajha;

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

public class SplashScreen extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash_screen); // Set your layout

        // Delay for a few seconds
        new Handler().postDelayed(() -> {
            startActivity(new Intent(SplashScreen.this, LandingPage.class));
            finish();
        }, 3000); // 2000 milliseconds (2 seconds)
    }
}
```

### XML

#### background\_splashscreen

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@color/black"/>
    <item>
        <bitmap
            android:src="@drawable/splash"
            android:gravity="center"/>
    </item>
</layer-list>
```

#### activity\_splash\_screen

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/black"
    tools:context=".About">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">
```

```
tools:ignore="ScrollViewSize">
```

```
<ImageView
    android:id="@+id/splashImage"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/splash"
    android:contentDescription="@string/app_name" />
</LinearLayout>
</ScrollView >
```

### Home Screen:

#### JAVA

```
package com.paurush.bhojan_saajha;
```

```
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
```

```
import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.cardview.widget.CardView;
```

```
public class LandingPage extends AppCompatActivity {
```

```
    private CardView login,register,about;
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_landingpage);
```

```
        login = findViewById(R.id.cardLogin);
        register = findViewById(R.id.cardRegister);
        about = findViewById(R.id.cardAboutus);
```

```
        login.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(LandingPage.this, LogUp.class));
            }
        });
```

```
        register.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(LandingPage.this, SignUp.class));
            }
        });
```

```
        about.setOnClickListener(new View.OnClickListener() {
            @Override
```

```

        public void onClick(View v) {
            startActivity(new Intent(LandingPage.this, About.class));
        }
    };
}
}

```

## XML

```

<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/black"
    tools:context=".LandingPage">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        tools:ignore="ScrollViewSize">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/welcome"
            android:fontFamily="@font/opensans"
            android:textSize="26sp"
            android:textColor="@color/white"
            android:layout_margin="20dp"/>

        <androidx.cardview.widget.CardView
            android:id="@+id/cardLogin"
            android:clickable="true"
            android:layout_width="match_parent"
            android:layout_height="180dp"
            app:cardCornerRadius="12dp"
            app:cardElevation="5dp"
            app:cardBackgroundColor="@color/gray"
            android:layout_margin="10dp"
            tools:ignore="KeyboardInaccessibleWidget">

            <ImageView
                android:layout_width="100dp"
                android:layout_height="100dp"
                android:src="@drawable/donate"
                android:layout_marginTop="18dp"
                android:layout_gravity="center_horizontal"
                tools:ignore="ContentDescription" />

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/login"

```

```
android:fontFamily="@font/opensans"  
android:textSize="18sp"  
android:textColor="@color/white"  
android:layout_marginTop="135dp"  
android:layout_gravity="center_horizontal"/>
```

```
</androidx.cardview.widget.CardView>
```

```
<androidx.cardview.widget.CardView  
    android:id="@+id/cardRegister"  
    android:clickable="true"  
    android:layout_width="match_parent"  
    android:layout_height="180dp"  
    app:cardCornerRadius="12dp"  
    app:cardElevation="5dp"  
    app:cardBackgroundColor="@color/gray"  
    android:layout_margin="10dp"  
    tools:ignore="KeyboardInaccessibleWidget">
```

```
<ImageView  
    android:layout_width="100dp"  
    android:layout_height="100dp"  
    android:src="@drawable/add"  
    android:layout_marginTop="18dp"  
    tools:ignore="ContentDescription"  
    android:layout_gravity="center_horizontal"/>
```

```
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/register"  
    android:fontFamily="@font/opensans"  
    android:textSize="18sp"  
    android:textColor="@color/white"  
    android:layout_marginTop="135dp"  
    android:layout_gravity="center_horizontal" />
```

```
</androidx.cardview.widget.CardView>
```

```
<androidx.cardview.widget.CardView  
    android:id="@+id/cardAboutus"  
    android:clickable="true"  
    android:layout_width="match_parent"  
    android:layout_height="180dp"  
    app:cardCornerRadius="12dp"  
    app:cardElevation="5dp"  
    app:cardBackgroundColor="@color/gray"  
    android:layout_margin="10dp"  
    tools:ignore="KeyboardInaccessibleWidget">
```

```
<ImageView  
    android:layout_width="100dp"  
    android:layout_height="100dp"  
    android:src="@drawable/about"  
    android:layout_marginTop="18dp"
```

```
tools:ignore="ContentDescription"
android:layout_gravity="center_horizontal"/>
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/about"
    android:fontFamily="@font/opensans"
    android:textSize="18sp"
    android:textColor="@color/white"
    android:layout_marginTop="135dp"
    android:layout_gravity="center_horizontal" />
</androidx.cardview.widget.CardView>
</LinearLayout>
</ScrollView>
```

### **Register Screen:**

#### **JAVA**

```
package com.paurush.bhojan_saajha;

import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FirebaseFirestore;

import java.util.HashMap;
import java.util.Map;
import java.util.Objects;

public class SignUp extends AppCompatActivity {

    public static final String TAG = "TAG";
    private EditText mFullName,mEmail,mPassword,mPhone;
    private FirebaseAuth fAuth;
    private FirebaseFirestore fStore;
    private String userID;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_signup);
```



```

mFullName = findViewById(R.id.name);
mEmail = findViewById(R.id.email);
mPassword = findViewById(R.id.password);
mPhone = findViewById(R.id.phone);
Button mRegisterBtn = findViewById(R.id.register);
TextView mLoginBtn = findViewById(R.id.login);

fAuth=FirebaseAuth.getInstance();
fStore=FirebaseFirestore.getInstance();

if(fAuth.getCurrentUser() !=null){
    Intent intent = new Intent(SignUp.this, MainActivity.class);
    intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_NEW_TASK);
    startActivity(intent);
}

mRegisterBtn.setOnClickListener(v -> {
    //startActivity(new Intent(SignUp.this, MainActivity.class));

    String email = mEmail.getText().toString().trim();
    String password= mPassword.getText().toString().trim();
    String name= mFullName.getText().toString().trim();
    String phone= mPhone.getText().toString().trim();

    if(TextUtils.isEmpty(email))
    {
        mEmail.setError("Email is Required.");
        return;
    }

    if(TextUtils.isEmpty(password))
    {
        mPassword.setError("Password is Required.");
        return;
    }

    if(password.length() < 6)
    {
        mPassword.setError("Password Must be >=6 Characters");
        return;
    }

    fAuth.createUserWithEmailAndPassword(email,password).addOnCompleteListener(task -> {
        if(task.isSuccessful()){
            Toast.makeText(SignUp.this, "User Created.", Toast.LENGTH_SHORT) .show();
            userID = Objects.requireNonNull(fAuth.getCurrentUser()).getUid();
            DocumentReference documentReference = fStore.collection("users").document(userID);
            Map<String,Object> user = new HashMap<>();
            user.put("name",name);
            user.put("email",email);
            user.put("phone",phone);
            documentReference.set(user).addOnSuccessListener(aVoid -> {
                Log.d(TAG,"onSuccess: user Profile is created for "+ userID);
            }
        }
    }
    }
    
```

```
        Toast.makeText(SignUp.this, "Registered Successfully.", Toast.LENGTH_SHORT).show();
    });
    //startActivity(new Intent(getApplicationContext(),MainActivity.class));
    Intent intent = new Intent(SignUp.this, MainActivity.class);
    intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_NEW_TASK);
    startActivity(intent);
}
else{
    Toast.makeText(SignUp.this, "Error!" +
Objects.requireNonNull(task.getException()).getMessage(),Toast.LENGTH_SHORT).show();
}
});
});
});

mLoginBtn.setOnClickListener(v -> startActivity(new Intent(getApplicationContext(), LogUp.class)));

}
}
```

### **XML**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/bthre"
    android:padding="16dp"
    android:gravity="center"
    tools:context=".SignUp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:gravity="center"
        android:layout_marginTop="-50dp"
        android:layout_marginRight="10dp"
        android:layout_marginLeft="10dp"
        tools:ignore="UselessParent">

        <ImageView
            android:id="@+id/profile"
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:background="@drawable/round_background"
            android:padding="15dp"
            android:src="@drawable/add"
            tools:ignore="ContentDescription" />

        <com.google.android.material.textfield.TextInputLayout
            android:id="@+id/nameError"
```

```
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="5dp"
app:hintAnimationEnabled="true"
android:textColorHint="@color/black"
app:hintTextColor="@color/black"
app:boxStrokeColor="@color/black">
```

```
<EditText
    android:id="@+id/name"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:hint="@string/name"
    android:textSize="20sp"
    android:fontFamily="@font/opensans"
    android:inputType="textNoSuggestions"
    android:background="@android:color/transparent"
    android:textColor="@color/black"
    android:textColorHint="@color/black"
    android:maxLines="1"
    android:singleLine="true"
    android:importantForAutofill="no"/>
```

```
</com.google.android.material.textfield.TextInputLayout>
```

```
<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/emailError"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="5dp"
    app:hintAnimationEnabled="true"
    android:textColorHint="@color/black"
    app:hintTextColor="@color/black"
    app:boxStrokeColor="@color/black">
```

```
<EditText
    android:id="@+id/email"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:hint="@string/prompt_email"
    android:textSize="20sp"
    android:fontFamily="@font/opensans"
    android:inputType="textEmailAddress"
    android:background="@android:color/transparent"
    android:maxLines="1"
    android:singleLine="true"
    android:textColor="@color/black"
    android:textColorHint="@color/black"
    android:importantForAutofill="no"/>
```

```
</com.google.android.material.textfield.TextInputLayout>
```

```

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/phoneError"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="5dp"
    app:hintAnimationEnabled="true"
    android:textColorHint="@color/black"
    app:hintTextColor="@color/black"
    app:boxStrokeColor="@color/black">

    <EditText
        android:id="@+id/phone"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:hint="@string/phone_number"
        android:textSize="20sp"
        android:fontFamily="@font/opensans"
        android:maxLength="13"
        android:background="@android:color/transparent"
        android:inputType="phone"
        android:maxLines="1"
        android:singleLine="true"
        android:textColor="@color/black"
        android:textColorHint="@color/black"
        android:importantForAutofill="no"/>

</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/passError"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:passwordToggleEnabled="true"
    android:layout_marginTop="5dp"
    app:hintAnimationEnabled="true"
    android:textColorHint="@color/black"
    app:hintTextColor="@color/black"
    app:boxStrokeColor="@color/black">

    <EditText
        android:id="@+id/password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:hint="@string/prompt_password"
        android:textSize="20sp"
        android:fontFamily="@font/opensans"
        android:inputType="textPassword"
        android:background="@android:color/transparent"
        android:maxLines="1"
        android:singleLine="true"
        android:textColor="@color/black"
        android:textColorHint="@color/black"

```

```
android:importantForAutofill="no"/>
```

```
</com.google.android.material.textfield.TextInputLayout>
```

```
<Button
```

```
    android:id="@+id/register"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:layout_margin="20dp"
    android:background="@drawable/button_rounded"
    android:text="@string/register"
    android:textColor="@android:color/white"
    android:textSize="16sp"/>
```

```
<TextView
```

```
    android:id="@+id/login"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="5dp"
    android:layout_marginBottom="5dp"
    android:gravity="center"
    android:text="@string/not_register"
    android:textColor="@color/white"
    android:fontFamily="@font/opensans"
    android:textSize="17sp"/>
```

```
</LinearLayout>
```

```
</LinearLayout>
```

### **Login Screen:**

#### **JAVA**

```
package com.paurush.bhojan_saajha;

import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;

import com.google.firebase.auth.FirebaseAuth;

import java.util.Objects;

public class LogUp extends AppCompatActivity {

    private EditText mEmail,mPassword;
    private FirebaseAuth fAuth;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable(this);
    setContentView(R.layout.activity_logup);

    mEmail = findViewById(R.id.email);
    mPassword = findViewById(R.id.password);
    TextView mRegisterBtn = findViewById(R.id.register);
    Button mLoginBtn = findViewById(R.id.login);

    FirebaseAuth fAuth = FirebaseAuth.getInstance();

    if(fAuth.getCurrentUser() !=null){
        Intent intent = new Intent(LogUp.this, MainActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_NEW_TASK);
        startActivity(intent);
    }

    mLoginBtn.setOnClickListener(v -> {

        String email = mEmail.getText().toString().trim();
        String password= mPassword.getText().toString().trim();

        if(TextUtils.isEmpty(email))
        {
            mEmail.setError("Email is Required.");
            return;
        }

        if(TextUtils.isEmpty(password))
        {
            mPassword.setError("Password is Required.");
            return;
        }

        if(password.length() < 6)
        {
            mPassword.setError("Password Must be >=6 Characters");
            return;
        }

        //authenticate the user
        fAuth.signInWithEmailAndPassword(email,password).addOnCompleteListener(task -> {
            if(task.isSuccessful()){
                Toast.makeText(LogUp.this, "Logged in Successfully.", Toast.LENGTH_SHORT) .show();
                Intent intent = new Intent(LogUp.this, MainActivity.class);
                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_NEW_TASK);
                startActivity(intent);
            }else{
                Toast.makeText(LogUp.this, "Error! " +
Objects.requireNonNull(task.getException()).getMessage(),Toast.LENGTH_SHORT).show();
            }
        });
    });
}
```

```

    }
    });
    //startActivity(new Intent(LogUp.this, MainActivity.class));
    });

    mRegisterBtn.setOnClickListener(v -> {
        // redirect to RegisterActivity
        startActivity(new Intent(LogUp.this, MainActivity.class));
    });

}
}

```

## XML

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/bthre"
    android:padding="16dp"
    android:gravity="center"
    tools:context=".LogUp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:gravity="center"
        android:layout_marginTop="-50dp"
        android:layout_marginRight="10dp"
        android:layout_marginLeft="10dp"
        tools:ignore="UselessParent">
        <ImageView
            android:id="@+id/profile"
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:background="@drawable/round_background"
            android:padding="15dp"
            android:src="@drawable/add"
            tools:ignore="ContentDescription" />

        <com.google.android.material.textfield.TextInputLayout
            android:id="@+id/emailError"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="5dp"
            app:hintAnimationEnabled="true"
            android:textColorHint="@color/black"

```

```
app:hintTextColor="@color/black"  
app:boxStrokeColor="@color/black">
```

```
<EditText  
    android:id="@+id/email"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="10dp"  
    android:hint="@string/prompt_email"  
    android:textSize="20sp"  
    android:fontFamily="@font/opensans"  
    android:importantForAutofill="no"  
    android:inputType="textEmailAddress"  
    android:maxLines="1"  
    android:background="@android:color/transparent"  
    android:textColor="@color/black"  
    android:textColorHint="@color/black"  
    android:singleLine="true"/>
```

```
</com.google.android.material.textfield.TextInputLayout>
```

```
<com.google.android.material.textfield.TextInputLayout  
    android:id="@+id/passError"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="5dp"  
    app:passwordToggleEnabled="true"  
    app:hintAnimationEnabled="true"  
    android:textColorHint="@color/black"  
    app:hintTextColor="@color/black"  
    app:boxStrokeColor="@color/black">
```

```
<EditText  
    android:id="@+id/password"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="10dp"  
    android:hint="@string/prompt_password"  
    android:fontFamily="@font/opensans"  
    android:textSize="20sp"  
    android:importantForAutofill="no"  
    android:inputType="textPassword"  
    android:background="@android:color/transparent"  
    android:textColor="@color/black"  
    android:textColorHint="@color/black"  
    android:maxLines="1"  
    android:singleLine="true" />
```

```
</com.google.android.material.textfield.TextInputLayout>
```

```
<Button  
    android:id="@+id/login"  
    android:layout_width="match_parent"  
    android:layout_height="50dp"
```



```
android:layout_margin="20dp"
android:background="@drawable/button_rounded"
android:text="@string/login"
android:textColor="@android:color/white"
android:textSize="16sp" />
```

```
<TextView
    android:id="@+id/register"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="5dp"
    android:gravity="center"
    android:text="@string/create_new"
    android:textColor="@color/white"
    android:fontFamily="@font/opensans"
    android:textSize="17sp" />
```

```
</LinearLayout>
</LinearLayout>
```

### **Dashboard Screen:**

#### **JAVA**

```
package com.paurush.bhojan_saajha;

import android.content.Intent;
import android.os.Bundle;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.cardview.widget.CardView;

import com.google.firebase.auth.FirebaseAuth;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);

        CardView donate = findViewById(R.id.cardDonate);
        CardView receive = findViewById(R.id.cardReceive);
        CardView logout = findViewById(R.id.cardLogout);
        CardView foodmap = findViewById(R.id.cardFoodmap);
        CardView mypin = findViewById(R.id.cardMyPin);
        CardView history = findViewById(R.id.cardHistory);
        CardView about = findViewById(R.id.cardAboutus);
        CardView contact = findViewById(R.id.cardContact);

        FirebaseAuth fAuth = FirebaseAuth.getInstance();
        if(fAuth.getCurrentUser() == null){
            Intent intent = new Intent(MainActivity.this, LandingPage.class);
```

```

        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_NEW_TASK);
        startActivity(intent);
    }

    donate.setOnClickListener(v -> startActivity(new Intent(MainActivity.this, Donate.class)));

    receive.setOnClickListener(v -> startActivity(new Intent(MainActivity.this, Receive.class)));

    logout.setOnClickListener(v -> {
        FirebaseAuth.getInstance().signOut();
        Intent intent = new Intent(MainActivity.this, LandingPage.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_NEW_TASK);
        startActivity(intent);
    });

    foodmap.setOnClickListener(v -> startActivity(new Intent(MainActivity.this, FoodMap.class)));

    mypin.setOnClickListener(v -> startActivity(new Intent(MainActivity.this, MyPin.class)));

    history.setOnClickListener(v -> startActivity(new Intent(MainActivity.this, History.class)));

    about.setOnClickListener(v -> startActivity(new Intent(MainActivity.this, About.class)));

    contact.setOnClickListener(v -> startActivity(new Intent(MainActivity.this, Contact.class)));

    }
}

```

## XML

```

<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/black"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="My Dashboard"
            android:fontFamily="@font/opensans"
            android:textSize="24sp"
            android:textColor="@color/white"
            android:layout_margin="20dp"/>
        <GridLayout

```

```
android:layout_width="wrap_content"
android:layout_height="match_parent"
android:rowCount="4"
android:columnCount="2"
android:layout_margin="5dp"
android:alignmentMode="alignMargins"
android:layout_gravity="center"
android:orientation="horizontal">

<androidx.cardview.widget.CardView
    android:id="@+id/cardDonate"
    android:clickable="true"
    android:layout_width="165dp"
    android:layout_height="165dp"
    app:cardCornerRadius="12dp"
    app:cardElevation="5dp"
    app:cardBackgroundColor="@color/gray"
    android:layout_margin="10dp">
    <ImageView
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:src="@drawable/donate"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="15dp" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Donate"
        android:fontFamily="@font/opensans"
        android:textSize="18dp"
        android:textColor="@color/white"
        android:layout_marginTop="130dp"
        android:layout_gravity="center_horizontal" />
</androidx.cardview.widget.CardView>

<androidx.cardview.widget.CardView
    android:id="@+id/cardReceive"
    android:clickable="true"
    android:layout_width="165dp"
    android:layout_height="165dp"
    app:cardCornerRadius="12dp"
    app:cardElevation="5dp"
    app:cardBackgroundColor="@color/gray"
    android:layout_margin="10dp">
    <ImageView
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:src="@drawable/rcv"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="15dp"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Receive"
```

```
        android:fontFamily="@font/opensans"
        android:textSize="18dp"
        android:textColor="@color/white"
        android:layout_marginTop="130dp"
        android:layout_gravity="center_horizontal"/>
</androidx.cardview.widget.CardView>
```

```
<androidx.cardview.widget.CardView
    android:id="@+id/cardFoodmap"
    android:clickable="true"
    android:layout_width="165dp"
    android:layout_height="165dp"
    app:cardCornerRadius="12dp"
    app:cardElevation="5dp"
    app:cardBackgroundColor="@color/gray"
    android:layout_margin="10dp">
```

```
<ImageView
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:src="@drawable/map"
    android:layout_gravity="center_horizontal"
    android:layout_marginTop="15dp"/>
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Food Map"
    android:fontFamily="@font/opensans"
    android:textSize="18dp"
    android:textColor="@color/white"
    android:layout_marginTop="130dp"
    android:layout_gravity="center_horizontal"/>
</androidx.cardview.widget.CardView>
```

```
<androidx.cardview.widget.CardView
    android:id="@+id/cardMyPin"
    android:clickable="true"
    android:layout_width="165dp"
    android:layout_height="165dp"
    app:cardCornerRadius="12dp"
    app:cardElevation="5dp"
    app:cardBackgroundColor="@color/gray"
    android:layout_margin="10dp">
```

```
<ImageView
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:src="@drawable/mypins"
    android:layout_gravity="center_horizontal"
    android:layout_marginTop="15dp" />
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="My Pins"
    android:fontFamily="@font/opensans"
```

```
        android:textSize="18dp"
        android:textColor="@color/white"
        android:layout_marginTop="130dp"
        android:layout_gravity="center_horizontal" />
</androidx.cardview.widget.CardView>
```

```
<androidx.cardview.widget.CardView
    android:id="@+id/cardHistory"
    android:clickable="true"
    android:layout_width="165dp"
    android:layout_height="165dp"
    app:cardCornerRadius="12dp"
    app:cardElevation="5dp"
    app:cardBackgroundColor="@color/gray"
    android:layout_margin="10dp">
    <ImageView
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:src="@drawable/history"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="15dp" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="History"
        android:fontFamily="@font/opensans"
        android:textSize="18dp"
        android:textColor="@color/white"
        android:layout_marginTop="130dp"
        android:layout_gravity="center_horizontal" />
</androidx.cardview.widget.CardView>
```

```
<androidx.cardview.widget.CardView
    android:id="@+id/cardAboutus"
    android:clickable="true"
    android:layout_width="165dp"
    android:layout_height="165dp"
    app:cardCornerRadius="12dp"
    app:cardElevation="5dp"
    app:cardBackgroundColor="@color/gray"
    android:layout_margin="10dp">
```

```
    <ImageView
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:src="@drawable/about"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="15dp"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="About Us"
        android:fontFamily="@font/opensans"
        android:textSize="18dp"
```

```
        android:textColor="@color/white"
        android:layout_marginTop="130dp"
        android:layout_gravity="center_horizontal"/>
</androidx.cardview.widget.CardView>
```

```
<androidx.cardview.widget.CardView
    android:id="@+id/cardContact"
    android:clickable="true"
    android:layout_width="165dp"
    android:layout_height="165dp"
    app:cardCornerRadius="12dp"
    app:cardElevation="5dp"
    app:cardBackgroundColor="@color/gray"
    android:layout_margin="10dp">
```

```
<ImageView
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:src="@drawable/contact"
    android:layout_gravity="center_horizontal"
    android:layout_marginTop="15dp"/>
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Contact Us"
    android:fontFamily="@font/opensans"
    android:textSize="18dp"
    android:textColor="@color/white"
    android:layout_marginTop="130dp"
    android:layout_gravity="center_horizontal"/>
```

```
</androidx.cardview.widget.CardView>
```

```
<androidx.cardview.widget.CardView
    android:id="@+id/cardLogout"
    android:clickable="true"
    android:layout_width="165dp"
    android:layout_height="165dp"
    app:cardCornerRadius="12dp"
    app:cardElevation="5dp"
    app:cardBackgroundColor="@color/gray"
    android:layout_margin="10dp">
```

```
<ImageView
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:src="@drawable/logout"
    android:layout_gravity="center_horizontal"
    android:layout_marginTop="15dp"/>
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Log Out"
    android:fontFamily="@font/opensans"
    android:textSize="18dp"
```

```
        android:textColor="@color/white"
        android:layout_marginTop="130dp"
        android:layout_gravity="center_horizontal"/>
    </androidx.cardview.widget.CardView>
```

```
    </GridLayout>
</LinearLayout>
</ScrollView>
```

### **Donate Screen:**

#### **JAVA**

```
package com.paurush.bhojan_saajha;
```

```
import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.location.Location;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
```

```
import androidx.activity.EdgeToEdge;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
```

```
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.CollectionReference;
import com.google.firebase.firestore.FieldValue;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.GeoPoint;
```

```
import java.util.HashMap;
import java.util.Map;
import java.util.Objects;
```

```
public class Donate extends AppCompatActivity implements OnMapReadyCallback,
GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener,
com.google.android.gms.location.LocationListener {
```

```
private GoogleMap mMap;
private GoogleApiClient mGoogleApiClient;
private Location mLastLocation;
private final int REQUEST_CODE = 11;
private SupportMapFragment mapFragment;
private EditText mFullName, mFoodItem, mDescription, mPhone;
private FirebaseAuth fAuth;
private FirebaseFirestore fStore;
private String userID;
public static final String TAG = "TAG";

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable(this);
    setContentView(R.layout.activity_donate);

    // Initialize views
    mFullName = findViewById(R.id.donorname);
    mFoodItem = findViewById(R.id.fooditem);
    mPhone = findViewById(R.id.phone);
    mDescription = findViewById(R.id.description);
    Button mSubmitBtn = findViewById(R.id.submit);

    // Initialize Firebase Auth and Firestore
    fAuth = FirebaseAuth.getInstance();
    fStore = FirebaseFirestore.getInstance();

    // Initialize Google Map
    mapFragment = (SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.google_map);
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {
        mapFragment.getMapAsync(this);
    } else {
        ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION}, REQUEST_CODE);
    }

    // Set onClickListener for submit button
    mSubmitBtn.setOnClickListener(v -> validateAndSubmitData());
}

@Override
public void onMapReady(@NonNull GoogleMap googleMap) {
    mMap = googleMap;
    buildGoogleApiClient();
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        return;
    }
    mMap.setMyLocationEnabled(true);
}
```



}

```
protected synchronized void buildGoogleApiClient() {
    mGoogleApiClient = new GoogleApiClient.Builder(this)
        .addConnectionCallbacks(this)
        .addOnConnectionFailedListener(this)
        .addApi(LocationServices.API)
        .build();
    mGoogleApiClient.connect();
}
```

@Override

```
public void onLocationChanged(@NonNull Location location) {
    mLastLocation = location;
    LatLng latLng = new LatLng(location.getLatitude(), location.getLongitude());

    MarkerOptions markerOptions = new MarkerOptions().position(latLng).title("You are here");
    mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(latLng, 15));
    Objects.requireNonNull(mMap.addMarker(markerOptions)).showInfoWindow();
}
```

```
private void validateAndSubmitData() {
    String fullname = mFullName.getText().toString().trim();
    String fooditem = mFoodItem.getText().toString().trim();
    String description = mDescription.getText().toString().trim();
    String phone = mPhone.getText().toString().trim();

    // Validate user inputs
    if (validateInputs(fullname, fooditem, phone)) {
        userID = Objects.requireNonNull(fAuth.getCurrentUser()).getUid();

        // Check if mLastLocation is not null before creating GeoPoint
        if (mLastLocation != null) {
            GeoPoint geoPoint = new GeoPoint(mLastLocation.getLatitude(), mLastLocation.getLongitude());
            saveDonationData(fullname, fooditem, description, phone, geoPoint);
        } else {
            Toast.makeText(getApplicationContext(), "Location not available!", Toast.LENGTH_SHORT).show();
        }
    }
}
```

```
private boolean validateInputs(String fullname, String fooditem, String phone) {
    if (TextUtils.isEmpty(fullname)) {
        mFullName.setError("Name is Required.");
        return false;
    }
    if (TextUtils.isEmpty(fooditem)) {
        mFoodItem.setError("Food item is required.");
        return false;
    }
    if (TextUtils.isEmpty(phone) || phone.length() != 10 || !phone.matches("\\d{10}")) {
        mPhone.setError("Phone Number must be exactly 10 digits and contain only numbers.");
        return false;
    }
}
```

```

        return true;
    }

    private void saveDonationData(String fullname, String fooditem, String description, String phone, GeoPoint
    geoPoint) {
        Map<String, Object> user = new HashMap<>();
        user.put("timestamp", FieldValue.serverTimestamp());
        user.put("name", fullname);
        user.put("food item", fooditem);
        user.put("phone", phone);
        user.put("description", description);
        user.put("location", geoPoint);
        user.put("userid", userID);
        user.put("type", "Donor");

        CollectionReference collectionReference = fStore.collection("user data");
        collectionReference.add(user)
            .addOnSuccessListener(documentReference -> {
                Toast.makeText(getApplicationContext(), "Submit Successful!", Toast.LENGTH_SHORT).show();
                Log.d(TAG, "Donation data saved successfully!");

                // Clear all fields after successful submission
                clearFields();

                navigateToMainActivity();
            })
            .addOnFailureListener(e -> {
                Toast.makeText(getApplicationContext(), "Error saving donation data!",
                Toast.LENGTH_SHORT).show();
                Log.w(TAG, "Error saving donation data", e);
            });
    }

    private void clearFields() {
        mFullName.setText("");
        mFoodItem.setText("");
        mDescription.setText("");
        mPhone.setText("");
    }

    private void navigateToMainActivity() {
        Intent intent = new Intent(Donate.this, MainActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.FLAG_ACTIVITY_CLEAR_TASK |
        Intent.FLAG_ACTIVITY_NEW_TASK);
        startActivity(intent);
    }

    @Override
    public void onConnected(@Nullable Bundle bundle) {
        LocationRequest mLocationRequest = new LocationRequest();
        mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
        if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) !=
        PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(this,
        Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {

```

```

        return;
    }
    LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient, mLocationRequest,
this);
}

@Override
public void onConnectionSuspended(int i) {

}

@Override
public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {

}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[]
grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    if (requestCode == REQUEST_CODE) {
        if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            mapFragment.getMapAsync(this);
        } else {
            Toast.makeText(this, "Permission Denied", Toast.LENGTH_SHORT).show();
        }
    }
}
}
}

```

## XML

```

<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/black"
    tools:context=".Donate">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/donate"
            android:fontFamily="@font/opensans"
            android:textSize="24sp"
            android:textColor="@color/white"
            android:layout_margin="20dp"/>

```

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center">
    <androidx.cardview.widget.CardView
        xmlns:Card_View="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_margin="5dp"
        android:padding="10dp"
        app:cardBackgroundColor="@android:color/transparent"
        Card_View:cardCornerRadius="5dp"
        Card_View:cardElevation="0dp">
```

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="10dp"
    android:layout_marginRight="10dp"
    android:layout_marginLeft="10dp"
    android:orientation="vertical">
```

```
<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/nameError"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textColorHint="@color/white"
    app:hintTextColor="@color/white"
    app:boxStrokeColor="@color/white">
```

```
<EditText
    android:id="@+id/donorname"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:hint="@string/donor_name"
    android:textSize="20sp"
    android:fontFamily="@font/opensans"
    android:inputType="text"
    android:textColor="@color/white"
    android:textColorHint="@color/white"
    android:backgroundTint="@android:color/transparent"
    android:maxLines="1"
    android:singleLine="true"
    android:importantForAutofill="no"
    tools:ignore="UnusedAttribute" />
```

```
</com.google.android.material.textfield.TextInputLayout>
```

```
<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/itemError"
    android:layout_width="match_parent"
```

```
android:layout_height="wrap_content"
android:layout_marginTop="5dp"
android:textColorHint="@color/white"
app:hintTextColor="@color/white"
app:boxStrokeColor="@color/white">

<EditText
    android:id="@+id/fooditem"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:hint="@string/food_name"
    android:textSize="20sp"
    android:fontFamily="@font/opensans"
    android:textColor="@color/white"
    android:textColorHint="@color/white"
    android:inputType="text"
    android:backgroundTint="@android:color/transparent"
    android:maxLines="2"
    android:singleLine="true"
    android:importantForAutofill="no"
    tools:ignore="UnusedAttribute" />

</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/phoneError"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="5dp"
    android:textColorHint="@color/white"
    app:hintTextColor="@color/white"
    app:boxStrokeColor="@color/white">

    <EditText
        android:id="@+id/phone"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:hint="@string/phone_number"
        android:textSize="20sp"
        android:fontFamily="@font/opensans"
        android:backgroundTint="@android:color/transparent"
        android:maxLength="12"
        android:inputType="phone"
        android:textColor="@color/white"
        android:textColorHint="@color/white"
        android:maxLines="1"
        android:singleLine="true"
        android:importantForAutofill="no"
        tools:ignore="UnusedAttribute" />

</com.google.android.material.textfield.TextInputLayout>
```

```
<com.google.android.material.textfield.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="5dp"
    android:textColorHint="@color/white"
    app:hintTextColor="@color/white"
    android:textSize="20sp"
    android:fontFamily="@font/opensans"
    app:boxStrokeColor="@color/white">

    <EditText
        android:id="@+id/description"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:hint="@string/description"
        android:textSize="20sp"
        android:fontFamily="@font/opensans"
        android:inputType="textMultiLine"
        android:maxLines="10"
        android:textColor="@color/white"
        android:textColorHint="@color/white"
        android:backgroundTint="@android:color/transparent"
        android:importantForAutofill="no"
        tools:ignore="UnusedAttribute" />

</com.google.android.material.textfield.TextInputLayout>

<fragment
    android:id="@+id/google_map"
    xmlns:map="http://schemas.android.com/apk/res-auto"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="200dp"
    android:layout_marginTop="20dp"
    map:uiZoomControls="true"
    tools:ignore="FragmentTagUsage" />

<Button
    android:id="@+id/submit"
    android:layout_width="match_parent"
    android:layout_height="60dp"
    android:layout_margin="20dp"
    android:backgroundTint="@color/button_background_color"
    android:text="@string/submit"
    android:textColor="@color/button_text_color"
    android:textSize="20sp"
    tools:ignore="UnusedAttribute" />

</LinearLayout>
</androidx.cardview.widget.CardView>
</RelativeLayout>
</LinearLayout>
</ScrollView>
```

## Receive Screen:

### JAVA

```
package com.paurush.bhojan_saajha;

import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.location.Location;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;

import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.CollectionReference;
import com.google.firebase.firestore.FieldValue;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.GeoPoint;

import java.util.HashMap;
import java.util.Map;
import java.util.Objects;

public class Receive extends AppCompatActivity implements OnMapReadyCallback,
GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener,
com.google.android.gms.location.LocationListener {

    private GoogleMap mMap;
    GoogleApiClient mGoogleApiClient;
    Location mLastLocation;
    LocationRequest mLocationRequest;
    private final int REQUEST_CODE = 11;
    SupportMapFragment mapFragment;
    EditText mFullName,mDescription;
    Button mSubmitBtn;
```

```

FirebaseAuth fAuth;
FirebaseFirestore fStore;
String userID;
public static final String TAG = "TAG";

```

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable(this);
    setContentView(R.layout.activity_receive);

    mFullName = findViewById(R.id.receivername);
    mDescription = findViewById(R.id.description);
    mSubmitBtn=findViewById(R.id.submit);

    fAuth=FirebaseAuth.getInstance();
    fStore= FirebaseFirestore.getInstance();

    mapFragment = (SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.google_map);
    if (ActivityCompat.checkSelfPermission(this, android.Manifest.permission.ACCESS_FINE_LOCATION)
== PackageManager.PERMISSION_GRANTED) {
        mapFragment.getMapAsync(this);
    } else {
        ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION}, REQUEST_CODE);
    }

}

```

```

@Override
public void onMapReady(@NonNull GoogleMap googleMap) {
    mMap = googleMap;
    buildGoogleApiClient();
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        return;
    }
    mMap.setMyLocationEnabled(true);
}

```

```

protected synchronized void buildGoogleApiClient(){
    mGoogleApiClient = new GoogleApiClient.Builder(this)
        .addConnectionCallbacks(this)
        .addOnConnectionFailedListener(this)
        .addApi(LocationServices.API)
        .build();
    mGoogleApiClient.connect();
}

```

```

@Override
public void onLocationChanged(@NonNull Location location) {
    mLastLocation = location;
}

```



```

LatLng latLng = new LatLng(location.getLatitude(),location.getLongitude());

//MarkerOptions markerOptions1 = new MarkerOptions().position(latLng).title("You are here");
//mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
//mMap.animateCamera(CameraUpdateFactory.zoomTo(15));
//mMap.addMarker(markerOptions1).showInfoWindow();

MarkerOptions markerOptions = new MarkerOptions().position(latLng).title("You are here");
mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(latLng,15));
Objects.requireNonNull(mMap.addMarker(markerOptions)).showInfoWindow();

mSubmitBtn.setOnClickListener(v -> {
    String fullname = mFullName.getText().toString().trim();
    String description= mDescription.getText().toString().trim();
    String type= "Receiver";

    if(TextUtils.isEmpty(fullname))
    {
        mFullName.setError("Name is Required.");
        return;
    }
    if(TextUtils.isEmpty(description))
    {
        mFullName.setError("Description is Required.");
        return;
    }
    }

    userID = Objects.requireNonNull(fAuth.getCurrentUser()).getUid();
    //DocumentReference documentReference = fStore.collection("receiver").document(userID);
    CollectionReference collectionReference = fStore.collection("user data");

    GeoPoint geoPoint = new GeoPoint(location.getLatitude(),location.getLongitude());
    Map<String,Object> user = new HashMap<>();
    user.put("timestamp", FieldValue.serverTimestamp());
    user.put("name",fullname);
    user.put("description",description);
    user.put("location",geoPoint);
    user.put("userid",userID);
    user.put("type",type);

    collectionReference.add(user)
        .addOnSuccessListener(documentReference -> {
            Toast.makeText(getApplicationContext(),"Success!",Toast.LENGTH_SHORT).show();
            Log.d(TAG,"Success!");
            //startActivity(new Intent(getApplicationContext(),MainActivity.class));
            Intent intent = new Intent(Receive.this, MainActivity.class);
            intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.FLAG_ACTIVITY_CLEAR_TASK
| Intent.FLAG_ACTIVITY_NEW_TASK);
            startActivity(intent);
        })
        .addOnFailureListener(e -> {
            Toast.makeText(getApplicationContext(),"Error!",Toast.LENGTH_SHORT).show();
            Log.w(TAG, "Error!", e);
        })
    
```

```

    });

    });
}

@Override
public void onConnected(@Nullable Bundle bundle) {
    mLocationRequest = new LocationRequest();
    //mLocationRequest.setInterval(1000);
    //mLocationRequest.setFastestInterval(1000);
    mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) !=
    PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(this,
    Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        return;
    }
    LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient, mLocationRequest,
    this);

}

@Override
public void onConnectionSuspended(int i) {

}

@Override
public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {

}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[]
grantResults) {

    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    if (requestCode == REQUEST_CODE) {
        if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            mapFragment.getMapAsync(this);
        } else {
            Toast.makeText(this, "Permission Denied", Toast.LENGTH_SHORT).show();
        }
    }
}

}

```

## XML

```

<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"

```

```

android:layout_height="match_parent"
android:background="@color/black"
tools:context=".Receive">
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Receive"
        android:fontFamily="@font/opensans"
        android:textSize="24sp"
        android:textColor="@color/white"
        android:layout_margin="20dp"
        tools:ignore="HardcodedText" />

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center">
        <androidx.cardview.widget.CardView
            xmlns:Card_View="http://schemas.android.com/apk/res-auto"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_gravity="center"
            android:layout_margin="5dp"
            android:padding="10dp"
            app:cardBackgroundColor="@android:color/transparent"
            Card_View:cardCornerRadius="5dp"
            Card_View:cardElevation="0dp">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginBottom="10dp"
                android:layout_marginRight="10dp"
                android:layout_marginLeft="10dp"
                android:orientation="vertical">

                <com.google.android.material.textfield.TextInputLayout
                    android:id="@+id/nameError"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:layout_marginTop="5dp"

                    android:textColorHint="@color/white"
                    app:hintTextColor="@color/white"
                    app:boxStrokeColor="@color/white">

                    <EditText
                        android:id="@+id/receivename"
                        android:layout_width="match_parent"
                        android:layout_height="wrap_content"

```

```
android:layout_marginTop="10dp"
android:hint="@string/receiver_name"
android:textSize="20sp"
android:fontFamily="@font/opensans"
android:inputType="text"
android:textColor="@color/white"
android:textColorHint="@color/white"
android:backgroundTint="@android:color/transparent"
android:maxLines="1"
android:singleLine="true"
android:importantForAutofill="no"
tools:ignore="UnusedAttribute" />
```

```
</com.google.android.material.textfield.TextInputLayout>
```

```
<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/descriptionError"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:passwordToggleEnabled="true"
    android:layout_marginTop="5dp"
    android:textColor="@color/white"
    android:textColorHint="@color/white"
    app:hintTextColor="@color/white"
    android:textSize="20sp"
    android:fontFamily="@font/opensans"
    app:boxStrokeColor="@color/white">
```

```
<EditText
    android:id="@+id/description"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:hint="@string/description"
    android:textSize="20sp"
    android:fontFamily="@font/opensans"
    android:inputType="textMultiLine"
    android:maxLines="10"
    android:textColor="@color/white"
    android:textColorHint="@color/white"
    android:backgroundTint="@android:color/transparent"
    android:importantForAutofill="no"
    tools:ignore="UnusedAttribute" />
```

```
</com.google.android.material.textfield.TextInputLayout>
```

```
<fragment
    android:id="@+id/google_map"
    xmlns:map="http://schemas.android.com/apk/res-auto"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="200dp"
    android:layout_marginTop="20dp"
    map:uiZoomControls="true"
```

```
tools:ignore="FragmentTagUsage" />
```

```
<Button
    android:id="@+id/submit"
    android:layout_width="match_parent"
    android:layout_height="60dp"
    android:layout_margin="20dp"
    android:backgroundTint="@color/button_background_color"
    android:text="@string/submit"
    android:textColor="@color/button_text_color"
    android:textSize="20sp"
    tools:ignore="UnusedAttribute" />

</LinearLayout>
</androidx.cardview.widget.CardView>
</RelativeLayout>
</LinearLayout>
</ScrollView>
```

### **Food Map Screen:**

#### **JAVA**

```
package com.paurush.bhojan_saajha;

import android.Manifest;
import android.content.pm.PackageManager;
import android.location.Location;
import android.os.Bundle;
import android.util.Log;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;

import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.GeoPoint;
import com.google.firebase.firestore.QueryDocumentSnapshot;

import java.util.Objects;

public class FoodMap extends AppCompatActivity implements OnMapReadyCallback,
```

GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener,  
com.google.android.gms.location.LocationListener {

```
private GoogleMap mMap;
GoogleApiClient mGoogleApiClient;
Location mLastLocation;
LocationRequest mLocationRequest;
SupportMapFragment mapFragment;
private final int REQUEST_CODE = 11;
FirebaseFirestore fStore;
public static final String TAG = "TAG";

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_food_map);

    mapFragment = (SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.google_map);
    if (ActivityCompat.checkSelfPermission(this, android.Manifest.permission.ACCESS_FINE_LOCATION)
== PackageManager.PERMISSION_GRANTED) {
        mapFragment.getMapAsync(this);
    } else {
        ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION}, REQUEST_CODE);
    }

}

@Override
public void onMapReady(@NonNull GoogleMap googleMap) {
    mMap = googleMap;
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        return;
    }
    buildGoogleApiClient();
    mMap.setMyLocationEnabled(true);
}

protected synchronized void buildGoogleApiClient(){
    mGoogleApiClient = new GoogleApiClient.Builder(this)
        .addConnectionCallbacks(this)
        .addOnConnectionFailedListener(this)
        .addApi(LocationServices.API)
        .build();
    mGoogleApiClient.connect();
}

@Override
public void onLocationChanged(@NonNull Location location) {
    mLastLocation = location;
    showLocation();
}
```

```

LatLng latLng = new LatLng(location.getLatitude(),location.getLongitude());

MarkerOptions markerOptions1 = new MarkerOptions().position(latLng).title("You are
here").icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_RED));

mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(latLng,15));
Objects.requireNonNull(mMap.addMarker(markerOptions1)).showInfoWindow();
}

public void showLocation() {
    FirebaseFirestore cloudstorage = FirebaseFirestore.getInstance();
    cloudstorage.collection("user data")
        .get()
        .addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                for (QueryDocumentSnapshot document : task.getResult()) {
                    Log.d(TAG, document.getId() + " => " + document.getData());//
                    if (document.contains("location") && document.contains("name") &&
document.contains("description")) {
                        GeoPoint location = (GeoPoint) document.get("location");
                        String title = (String) document.get("name");
                        String type = (String) document.get("type");
                        String description = (String) document.get("description");

                        assert type != null;
                        if(type.equals("Donor")) {
                            Log.d(TAG, location + " Success " + title);
                            assert location != null;
                            LatLng latLng = new LatLng(location.getLatitude(), location.getLongitude());
                            //mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
                            mMap.addMarker(new
MarkerOptions().position(latLng).title(title+"("+type+)").snippet(description).icon(BitmapDescriptorFactory.d
efaultMarker(BitmapDescriptorFactory.HUE_GREEN)));
                        }
                        else if(type.equals("Receiver")){
                            Log.d(TAG, location + " Success " + title);
                            assert location != null;
                            LatLng latLng = new LatLng(location.getLatitude(), location.getLongitude());
                            //mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
                            mMap.addMarker(new
MarkerOptions().position(latLng).title(title+"("+type+)").snippet(description).icon(BitmapDescriptorFactory.d
efaultMarker(BitmapDescriptorFactory.HUE_BLUE)));
                        }
                    }
                }
            } else {
                Log.d(TAG, "Error fetching data: ", task.getException());
            }
        });
}

@Override
public void onConnected(@Nullable Bundle bundle) {

```

```

        mLocationRequest = new LocationRequest();
        mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
        if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
            return;
        }
        LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient, mLocationRequest,
this);

    }

    @Override
    public void onConnectionSuspended(int i) {

    }

    @Override
    public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {

    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[]
grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        if (requestCode == REQUEST_CODE) {
            if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                mapFragment.getMapAsync(this);
            } else {
                Toast.makeText(this, "Permission Denied", Toast.LENGTH_SHORT).show();
            }
        }
    }
}

```

## XML

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/black"
    android:foregroundServiceType="location"
    tools:context=".FoodMap">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/foodmap"
        android:fontFamily="@font/opensans"
        android:textSize="24sp"
        android:textColor="@color/white"

```



```

        android:layout_margin="20dp"/>
    <fragment
        android:id="@+id/google_map"
        xmlns:map="http://schemas.android.com/apk/res-auto"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="70dp"
        map:uiZoomControls="true"
    />
</RelativeLayout>

```

### History Screen:

#### JAVA

```

package com.paurush.bhojan_saajha;

import android.os.Bundle;
import android.util.Log;
import android.widget.CheckBox;
import android.widget.LinearLayout;
import android.widget.TextView;
import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;
import com.google.firebase.Timestamp;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.CollectionReference;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Locale;
import java.util.Map;

public class History extends AppCompatActivity {

    private final FirebaseFirestore db = FirebaseFirestore.getInstance();
    private final CollectionReference notebookRef = db.collection("user data");
    public static final String TAG = "TAG";
    private LinearLayout showDataLayout;
    private FirebaseAuth fAuth;
    private final List<String> selectedIds = new ArrayList<>(); // To store IDs of selected items
    private final Map<String, CheckBox> checkBoxMap = new HashMap<>(); // To store checkboxes for items

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_history);
    }

```

```

fAuth = FirebaseAuth.getInstance();
showDataLayout = findViewById(R.id.showdata);

findViewById(R.id.delete).setOnClickListener(v -> deleteSelectedNotes());

// Set up the "Select All" checkbox
CheckBox selectAllCheckBox = findViewById(R.id.select_all);
selectAllCheckBox.setOnCheckedChangeListener((buttonView, isChecked) -> {
    for (CheckBox checkBox : checkBoxMap.values()) {
        checkBox.setChecked(isChecked);
        if (isChecked) {
            selectedIds.add(checkBox.getText().toString()); // Update selection logic as needed
        } else {
            selectedIds.clear(); // Clear selection if unchecked
        }
    }
});

loadNotes();
}

public void loadNotes() {
    notebookRef.get()
        .addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                String userID = FirebaseAuth.getCurrentUser() != null ? FirebaseAuth.getCurrentUser().getUid() : null;

                for (QueryDocumentSnapshot document : task.getResult()) {
                    Log.d(TAG, document.getId() + " => " + document.getData());

                    if (document.contains("name") && document.contains("description")
                        && document.contains("userid") && document.contains("timestamp")) {

                        String name = document.getString("name");
                        String description = document.getString("description");
                        String userIdFromDoc = document.getString("userid");
                        Timestamp timestamp = document.getTimestamp("timestamp");

                        // Check if the current user ID matches
                        if (userIdFromDoc != null && userIdFromDoc.equals(userID)) {
                            addCheckBoxToLayout(document.getId(), name, description, timestamp);
                        }
                    }
                }

                // If no data is found, display the no data message properly centered
                if (showDataLayout.getChildCount() == 0) {
                    TextView noDataText = new TextView(this);
                    noDataText.setText(R.string.no_donation_records);
                    noDataText.setTextColor(ContextCompat.getColor(this, android.R.color.white));
                    noDataText.setTextSize(18); // Adjust text size as needed
                    noDataText.setGravity(android.view.Gravity.CENTER); // Center the text horizontally

                    // Set layout parameters to ensure proper margins and width

```

```

        LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.MATCH_PARENT,
            LinearLayout.LayoutParams.WRAP_CONTENT
        );
        params.setMargins(20, 20, 20, 20); // Add some margin around the text
        noDataText.setLayoutParams(params);

        showDataLayout.addView(noDataText);
    }
    } else {
        Log.d(TAG, "Error fetching data: ", task.getException());
    }
    });
}

private void addCheckBoxToLayout(String documentId, String name, String description, Timestamp
timestamp) {
    LinearLayout layout = new LinearLayout(this);
    layout.setOrientation(LinearLayout.VERTICAL);

    // Format the timestamp if needed
    String dateAndTime = formatTimestamp(timestamp);

    // Create a new checkbox with white text to be visible on the black background
    CheckBox checkBox = new CheckBox(this);
    checkBox.setText(getString(R.string.checkbox_text, name, description, dateAndTime));
    checkBox.setTextColor(ContextCompat.getColor(this, android.R.color.white)); // Set text color for
visibility

    checkBox.setOnCheckedChangeListener((buttonView, isChecked) -> {
        if (isChecked) {
            selectedIds.add(documentId);
        } else {
            selectedIds.remove(documentId);
        }
    });

    checkBoxMap.put(documentId, checkBox); // Store checkbox reference
    layout.addView(checkBox);

    showDataLayout.addView(layout);
}

private String formatTimestamp(Timestamp timestamp) {
    if (timestamp != null) {
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.getDefault());
        return sdf.format(timestamp.toDate());
    }
    return "No Date"; // Fallback if timestamp is null
}

private void deleteSelectedNotes() {
    for (String id : selectedIds) {

```

```

        notebookRef.document(id).delete()
        .addOnSuccessListener(aVoid -> Log.d(TAG, "DocumentSnapshot successfully deleted!"))
        .addOnFailureListener(e -> Log.w(TAG, "Error deleting document", e));
    }

    // Clear selections
    selectedIds.clear();
    showDataLayout.removeAllViews(); // Clear the layout and reload notes
    loadNotes();
}
}

```

## XML

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.core.widget.NestedScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/black"
    tools:context=".History">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        tools:ignore="ScrollViewSize">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/history"
            android:fontFamily="@font/opensans"
            android:textSize="26sp"
            android:textColor="@color/white"
            android:layout_margin="20dp"/>

        <CheckBox
            android:id="@+id/select_all"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Select All"
            android:textColor="@color/white"
            android:layout_margin="20dp"/>

        <LinearLayout
            android:id="@+id/showdata"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">
            <!-- Data will be added dynamically here -->
        </LinearLayout>
    </LinearLayout>

```

```
<androidx.cardview.widget.CardView
    android:id="@+id/delete"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:layout_marginTop="20dp"
    android:layout_marginBottom="20dp"
    android:layout_marginLeft="10dp"
    android:layout_marginRight="10dp"
    android:clickable="true"
    app:cardBackgroundColor="@color/red"
    app:cardCornerRadius="12dp"
    app:cardElevation="5dp">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:fontFamily="@font/opensans"
        android:text="@string/delete"
        android:textColor="@color/white"
        android:textSize="18sp" />
    </androidx.cardview.widget.CardView>
</LinearLayout>
</androidx.core.widget.NestedScrollView>
```

### My Pins Screen:

#### JAVA

```
package com.paurush.bhojan_saajha;

import android.Manifest;
import android.content.pm.PackageManager;
import android.location.Location;
import android.os.Bundle;
import android.util.Log;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;

import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
```

```
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.GeoPoint;
import com.google.firebase.firestore.QueryDocumentSnapshot;

import java.util.Objects;

public class MyPin extends AppCompatActivity implements OnMapReadyCallback,
GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener,
com.google.android.gms.location.LocationListener {

    private GoogleMap mMap;
    GoogleApiClient mGoogleApiClient;
    Location mLastLocation;
    LocationRequest mLocationRequest;
    SupportMapFragment mapFragment;
    private final int REQUEST_CODE = 11;
    FirebaseFirestore fStore;
    FirebaseAuth fAuth;
    public static final String TAG = "TAG";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_my_pin);

        fAuth= FirebaseAuth.getInstance();

        mapFragment = (SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.google_map);
        if (ActivityCompat.checkSelfPermission(this, android.Manifest.permission.ACCESS_FINE_LOCATION)
== PackageManager.PERMISSION_GRANTED) {
            mapFragment.getMapAsync(this);
        } else {
            ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION}, REQUEST_CODE);
        }

    }

    @Override
    public void onMapReady(@NonNull GoogleMap googleMap) {
        mMap = googleMap;
        if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
            return;
        }
        buildGoogleApiClient();
        mMap.setMyLocationEnabled(true);
    }
}
```

```

}

protected synchronized void buildGoogleApiClient(){
    mGoogleApiClient = new GoogleApiClient.Builder(this)
        .addConnectionCallbacks(this)
        .addOnConnectionFailedListener(this)
        .addApi(LocationServices.API)
        .build();
    mGoogleApiClient.connect();
}

@Override
public void onLocationChanged(@NonNull Location location) {
    mLastLocation = location;
    showLocation();
    LatLng latLng = new LatLng(location.getLatitude(),location.getLongitude());

    MarkerOptions markerOptions1 = new MarkerOptions().position(latLng).title("You are
here").icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_RED));
    //mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
    //mMap.animateCamera(CameraUpdateFactory.zoomTo(15));
    //mMap.addMarker(markerOptions1).showInfoWindow();
    mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(latLng,15));
    Objects.requireNonNull(mMap.addMarker(markerOptions1)).showInfoWindow();
}

public void showLocation() {
    FirebaseFirestore cloudstorage = FirebaseFirestore.getInstance();
    cloudstorage.collection("user data")
        .get()
        .addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                for (QueryDocumentSnapshot document : task.getResult()) {
                    Log.d(TAG, document.getId() + " => " + document.getData());//
                    if (document.contains("location") && document.contains("name") &&
document.contains("description") && document.contains("userid")) {
                        GeoPoint location = (GeoPoint) document.get("location");
                        String title = (String) document.get("name");
                        String type = (String) document.get("type");
                        String description = (String) document.get("description");
                        String Userid = (String) document.get("userid");
                        String userID = Objects.requireNonNull(fAuth.getCurrentUser()).getUid();

                        assert type != null;
                        assert Userid != null;
                        if(type.equals("Donor") & Userid.equals(userID)) {
                            Log.d(TAG, userID + " Success " + title);
                            assert location != null;
                            LatLng latLng = new LatLng(location.getLatitude(), location.getLongitude());
                            //mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
                            mMap.addMarker(new
MarkerOptions().position(latLng).title(title+"("+type+)").snippet(description).icon(BitmapDescriptorFactory.d
efaultMarker(BitmapDescriptorFactory.HUE_GREEN)));
                        }
                    }
                }
            }
        })
    }

```

```

        else if(type.equals("Receiver") & Userid.equals(userID)){
            Log.d(TAG, location + " Success " + title);
            assert location != null;
            LatLng latLng = new LatLng(location.getLatitude(), location.getLongitude());
            //mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
            mMap.addMarker(new
MarkerOptions().position(latLng).title(title+"("+type+")").snippet(description).icon(BitmapDescriptorFactory.d
efaultMarker(BitmapDescriptorFactory.HUE_BLUE)));
        }
    }
} else {
    Log.d(TAG, "Error fetching data: ", task.getException());
}
});

}

@Override
public void onConnected(@Nullable Bundle bundle) {
    mLocationRequest = new LocationRequest();
    mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        return;
    }
    LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient, mLocationRequest,
this);

}

@Override
public void onConnectionSuspended(int i) {

}

@Override
public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {

}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[]
grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    if (requestCode == REQUEST_CODE) {
        if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            mapFragment.getMapAsync(this);
        } else {
            Toast.makeText(this, "Permission Denied", Toast.LENGTH_SHORT).show();
        }
    }
}
}

```



}

## XML

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/black"
    android:foregroundServiceType="location"
    tools:context=".MyPin">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/mypin"
        android:fontFamily="@font/opensans"
        android:textSize="24sp"
        android:textColor="@color/white"
        android:layout_margin="20dp"/>
    <fragment
        android:id="@+id/google_map"
        xmlns:map="http://schemas.android.com/apk/res-auto"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="70dp"
        map:uiZoomControls="true"
    />
</RelativeLayout>
```

## Contact Screen:

## JAVA

```
package com.paurush.bhojan_saajha;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.util.Patterns;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import com.google.android.material.textfield.TextInputLayout;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.CollectionReference;
import com.google.firebase.firestore.FieldValue;
import com.google.firebase.firestore.FirebaseFirestore;

import java.util.HashMap;
import java.util.Map;
```

```
import java.util.Objects;

public class Contact extends AppCompatActivity {

    EditText name, email, message;
    Button submit;
    boolean isNameValid, isEmailValid, isMessageValid;
    FirebaseAuth fAuth;
    FirebaseFirestore fStore;
    String userID;
    public static final String TAG = "TAG";
    TextInputLayout nameError, emailError, messageError;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_contact);

        name = findViewById(R.id.name);
        email = findViewById(R.id.email);
        message = findViewById(R.id.message);
        submit = findViewById(R.id.submit);
        nameError = findViewById(R.id.nameError);
        emailError = findViewById(R.id.emailError);
        messageError = findViewById(R.id.messageError);

        fAuth=FirebaseAuth.getInstance();
        fStore= FirebaseFirestore.getInstance();

        submit.setOnClickListener(v -> SetValidation());
    }

    public void SetValidation() {

        // Check for a valid name.
        if (name.getText().toString().isEmpty()) {
            nameError.setError(getResources().getString(R.string.name_error));
            isNameValid = false;
        } else {
            isNameValid = true;
            nameError.setErrorEnabled(false);
        }

        // Check for a valid email address.
        if (email.getText().toString().isEmpty()) {
            emailError.setError(getResources().getString(R.string.email_error));
            isEmailValid = false;
        } else if (!Patterns.EMAIL_ADDRESS.matcher(email.getText().toString()).matches()) {
            emailError.setError(getResources().getString(R.string.error_invalid_email));
            isEmailValid = false;
        } else {
            isEmailValid = true;
        }
    }
}
```

```

        emailError.setErrorEnabled(false);
    }

    // Check for a valid phone number.
    if (message.getText().toString().isEmpty()) {
        messageError.setError(getResources().getString(R.string.phone_error));
        isMessageValid = false;
    } else {
        isMessageValid = true;
        messageError.setErrorEnabled(false);
    }

    if (isNameValid && isEmailValid && isMessageValid ) {

        String Name = name.getText().toString().trim();
        String Email= email.getText().toString().trim();
        String Message= message.getText().toString().trim();
        userID = Objects.requireNonNull(fAuth.getCurrentUser()).getUid();
        //DocumentReference documentReference = fStore.collection("donate").document(userID);
        CollectionReference collectionReference = fStore.collection("contact data");

        Map<String,Object> user = new HashMap<>();
        user.put("timestamp", FieldValue.serverTimestamp());
        user.put("name",Name);
        user.put("email",Email);
        user.put("message",Message);
        user.put("userid",userID);

        collectionReference.add(user)
            .addOnSuccessListener(documentReference -> {
                Toast.makeText(getApplicationContext(),"Success!",Toast.LENGTH_SHORT).show();
                Log.d(TAG,"Successfully! We will shortly revert you back.");
                //startActivity(new Intent(getApplicationContext(),MainActivity.class));
                Intent intent = new Intent(Contact.this, MainActivity.class);
                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.FLAG_ACTIVITY_CLEAR_TASK
| Intent.FLAG_ACTIVITY_NEW_TASK);
                startActivity(intent);
            })
            .addOnFailureListener(e -> {
                Toast.makeText(getApplicationContext(),"Error!",Toast.LENGTH_SHORT).show();
                Log.w(TAG, "Error!", e);
            });
    }

}
}

```

## **XML**

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"

```

```
android:layout_height="match_parent"
android:background="@drawable/btwoo"
android:padding="16dp"
android:gravity="center"
tools:context=".Contact">
```

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:gravity="center"
    android:layout_marginTop="-50dp"
    android:layout_marginRight="10dp"
    android:layout_marginLeft="10dp">
```

```
<ImageView
    android:id="@+id/profile"
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:background="@drawable/round_background"
    android:padding="15dp"
    android:src="@drawable/contact"
    tools:ignore="ContentDescription" />
```

```
<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/nameError"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="5dp"
    app:hintAnimationEnabled="true"
    android:textColorHint="@color/black"
    app:hintTextColor="@color/black"
    app:boxStrokeColor="@color/black">
```

```
<EditText
    android:id="@+id/name"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:hint="@string/name"
    android:textSize="20sp"
    android:fontFamily="@font/opensans"
    android:inputType="textNoSuggestions"
    android:background="@android:color/transparent"
    android:textColor="@color/black"
    android:textColorHint="@color/black"
    android:maxLines="1"
    android:singleLine="true"
    android:importantForAutofill="no" />
```

```
</com.google.android.material.textfield.TextInputLayout>
```

```
<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/emailError"
```

```
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="5dp"
app:hintAnimationEnabled="true"
android:textColorHint="@color/black"
app:hintTextColor="@color/black"
app:boxStrokeColor="@color/black">
```

```
<EditText
    android:id="@+id/email"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:hint="@string/prompt_email"
    android:textSize="20sp"
    android:fontFamily="@font/opensans"
    android:inputType="textEmailAddress"
    android:background="@android:color/transparent"
    android:maxLines="1"
    android:singleLine="true"
    android:textColor="@color/black"
    android:textColorHint="@color/black"
    android:importantForAutofill="no" />
```

```
</com.google.android.material.textfield.TextInputLayout>
```

```
<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/messageError"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="5dp"
    app:hintAnimationEnabled="true"
    android:textColorHint="@color/black"
    app:hintTextColor="@color/black"
    app:boxStrokeColor="@color/black">
```

```
<EditText
    android:id="@+id/message"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:hint="@string/message"
    android:textSize="20sp"
    android:fontFamily="@font/opensans"
    android:background="@android:color/transparent"
    android:inputType="text"
    android:maxLines="20"
    android:singleLine="true"
    android:textColor="@color/black"
    android:textColorHint="@color/black"
    android:importantForAutofill="no" />
```

```
</com.google.android.material.textfield.TextInputLayout>
```

```
<Button
    android:id="@+id/submit"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:layout_margin="20dp"
    android:background="@drawable/button_rounded"
    android:text="@string/submit"
    android:textColor="@android:color/white"
    android:textSize="16sp"/>

</LinearLayout>
</LinearLayout>
```

### About Us Screen:

#### JAVA

```
package com.paurush.bhojan_saajha;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.cardview.widget.CardView;

public class About extends AppCompatActivity {

    CardView instagram,facebook,twitter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_about);

        instagram = findViewById(R.id.instagram);
        facebook = findViewById(R.id.facebook);
        twitter = findViewById(R.id.twitter);

        instagram.setOnClickListener(v -> {
            Intent myWebLink = new Intent(Intent.ACTION_VIEW);
            myWebLink.setData(Uri.parse("http://www.instagram.com"));
            startActivity(myWebLink);
        });
        facebook.setOnClickListener(v -> {
            Intent myWebLink = new Intent(Intent.ACTION_VIEW);
            myWebLink.setData(Uri.parse("http://www.facebook.com"));
            startActivity(myWebLink);
        });
        twitter.setOnClickListener(v -> {
            Intent myWebLink = new Intent(Intent.ACTION_VIEW);
            myWebLink.setData(Uri.parse("http://www.twitter.com"));
            startActivity(myWebLink);
        });
    }
}
```

```
}  
}
```

## XML

```
<?xml version="1.0" encoding="utf-8"?>  
<ScrollView  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:background="@color/black"  
    tools:context=".About">  
    <LinearLayout  
        android:layout_width="match_parent"  
        android:layout_height="match_parent"  
        android:orientation="vertical"  
        tools:ignore="ScrollViewSize">  
        <TextView  
            android:layout_width="wrap_content"  
            android:layout_height="wrap_content"  
            android:text="@string/about"  
            android:fontFamily="@font/opensans"  
            android:textSize="26sp"  
            android:textColor="@color/white"  
            android:layout_margin="20dp"/>  
  
        <ImageView  
            android:layout_width="200dp"  
            android:layout_height="200dp"  
            android:layout_gravity="center_horizontal"  
            android:layout_marginTop="18dp"  
            android:layout_marginBottom="28dp"  
            android:src="@drawable/splash"  
            tools:ignore="ContentDescription" />  
  
        <TextView  
            android:layout_width="wrap_content"  
            android:layout_height="wrap_content"  
            android:layout_margin="10dp"  
            android:fontFamily="@font/nunitosans_light"  
            android:text="@string/aboutus"  
            android:textColor="@color/white"  
            android:justificationMode="inter_word"  
            android:textSize="20sp" />  
  
        <androidx.cardview.widget.CardView  
            android:id="@+id/instagram"  
            android:layout_width="match_parent"  
            android:layout_height="60dp"  
            android:layout_margin="10dp"  
            android:clickable="true"  
            app:cardBackgroundColor="@color/gray"  
            app:cardCornerRadius="12dp"
```

```
app:cardElevation="5dp"
tools:ignore="KeyboardInaccessibleWidget">
```

```
<ImageView
    android:layout_width="40dp"
    android:layout_height="40dp"
    android:layout_gravity="center_vertical"
    android:layout_marginLeft="40dp"
    android:layout_marginRight="10dp"
    android:src="@drawable/instagram"
    tools:ignore="ContentDescription" />
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:fontFamily="@font/opensans"
    android:text="Instagram"
    android:textColor="@color/white"
    android:textSize="25sp" />
```

```
</androidx.cardview.widget.CardView>
```

```
<androidx.cardview.widget.CardView
    android:id="@+id/facebook"
    android:layout_width="match_parent"
    android:layout_height="60dp"
    android:layout_margin="10dp"
    android:clickable="true"
    app:cardBackgroundColor="@color/gray"
    app:cardCornerRadius="12dp"
    app:cardElevation="5dp"
    tools:ignore="KeyboardInaccessibleWidget">
```

```
<ImageView
    android:layout_width="40dp"
    android:layout_height="40dp"
    android:layout_gravity="center_vertical"
    android:layout_marginLeft="40dp"
    android:layout_marginRight="10dp"
    android:src="@drawable/facebook"
    tools:ignore="ContentDescription" />
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:fontFamily="@font/opensans"
    android:text="Facebook"
    android:textColor="@color/white"
    android:textSize="25sp" />
```

```
</androidx.cardview.widget.CardView>
```



```

<androidx.cardview.widget.CardView
    android:id="@+id/twitter"
    android:layout_width="match_parent"
    android:layout_height="60dp"
    android:layout_margin="10dp"
    android:clickable="true"
    app:cardBackgroundColor="@color/gray"
    app:cardCornerRadius="12dp"
    app:cardElevation="5dp"
    tools:ignore="KeyboardInaccessibleWidget">
    .....0\1.....
    <ImageView
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:layout_gravity="center_vertical"
        android:layout_marginLeft="40dp"
        android:layout_marginRight="10dp"
        android:src="@drawable/twitter"
        tools:ignore="ContentDescription" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:fontFamily="@font/opensans"
        android:text="Twitter"
        android:textColor="@color/white"
        android:textSize="25sp" />
    </androidx.cardview.widget.CardView>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:fontFamily="@font/opensans_light"
        android:text="Version 1.0.0"
        android:textColor="@color/white"
        android:textSize="20sp" />
    </LinearLayout>
</ScrollView>

```

### **Model.java**

```

package com.paurush.bhojan_saajha;

public class model {
    String name,type,description,userid;

    public model() {

    }

    public model(String name, String type, String description, String userid) {
        this.name = name;
    }
}

```

```
this.type = type;
this.description = description;
this.userid = userid;
}

public String getUserid() {
    return userid;
}

public void setUserid(String userid) {
    this.userid = userid;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getType() {
    return type;
}

public void setType(String type) {
    this.type = type;
}

public String getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description;
}
}
```

### **MyAdapter.java**

```
package com.paurush.bhojan_saajha;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.google.firebase.auth.FirebaseAuth;

import java.util.ArrayList;
import java.util.Objects;

public class MyAdapter extends RecyclerView.Adapter<MyAdapter.myviewholder> {
```

```

ArrayList<model> datalist;
FirebaseAuth fAuth= FirebaseAuth.getInstance();
public String userID = Objects.requireNonNull(fAuth.getCurrentUser()).getUid();
public String uid;

public MyAdapter(ArrayList<model> datalist) {
    this.datalist = datalist;
}

@NonNull
@Override
public myviewholder onCreateViewHolder (@NonNull ViewGroup parent, int viewType){
    View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.singlerow, parent, false);
    return new myviewholder(view);
}

@Override
public void onBindViewHolder(@NonNull myviewholder holder, int position) {
    holder.tname.setText(datalist.get(position).getName());
    holder.ttype.setText(datalist.get(position).getType());
    holder.tdescription.setText(datalist.get(position).getDescription());
}

public void deleteItem(int position){
    //getSnapshots().getSnapshot(position).getReference().delete();
    //notifyDataSetChanged();
}

@Override
public int getItemCount() {
    return datalist.size();
}

public static class myviewholder extends RecyclerView.ViewHolder
{
    TextView tname,ttype,tdescription;
    public myviewholder(@NonNull View itemView) {
        super(itemView);
        tname = itemView.findViewById(R.id.name);
        ttype = itemView.findViewById(R.id.type);
        tdescription = itemView.findViewById(R.id.description);
    }
}
}

```

### **UserdataActivity.java**

```

package com.paurush.bhojan_saajha;

import android.os.Bundle;
import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import com.google.firebase.auth.FirebaseAuth;

```

```
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.Query;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
public class UserdataActivity extends AppCompatActivity {

    RecyclerView recyclerView;
    ArrayList<model> datalist;
    FirebaseFirestore db;
    MyAdapter adapter;
    FirebaseAuth fAuth = FirebaseAuth.getInstance();
    public String userID = Objects.requireNonNull(fAuth.getCurrentUser()).getUid();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_userdata);

        // Initialize RecyclerView and Adapter
        recyclerView = findViewById(R.id.rec_view);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        datalist = new ArrayList<>();
        adapter = new MyAdapter(datalist);
        recyclerView.setAdapter(adapter);

        // Initialize Firestore
        db = FirebaseFirestore.getInstance();

        // Fetch user data from Firestore
        db.collection("user data").orderBy("timestamp", Query.Direction.DESENDING).get()
            .addOnSuccessListener(queryDocumentSnapshots -> {
                List<DocumentSnapshot> list = queryDocumentSnapshots.getDocuments();
                int initialSize = datalist.size(); // Store initial size of the datalist

                for (DocumentSnapshot d : list) {
                    model obj = d.toObject(model.class);
                    String Userid = (String) d.get("userid");
                    assert Userid != null;
                    if (Userid.equals(userID)) {
                        datalist.add(obj); // Add object to datalist
                    }
                }

                int newSize = datalist.size(); // Get new size after additions
                // Notify adapter of the new items added
                adapter.notifyItemRangeInserted(initialSize, newSize - initialSize);
            })
            .addOnFailureListener(e -> {
                // Handle the error
            });
    }
}
```



# ***Conclusion and Future Enhancements***

## 10. CONCLUSION AND FUTURE ENHANCEMENTS

---

### CONCLUSION:

The "**Bhojan Saajha**" app was developed to address the critical issue of food wastage by connecting restaurants with individuals in need. By providing a user-friendly platform, it bridges the gap between excess food and those who can benefit from it. The app successfully integrates various functionalities like user and donor registration, food listing, real-time notifications, and map-based search for nearby donations.

This project not only promotes food security and reduces wastage but also fosters community support by facilitating easy food sharing. The successful development of the app highlights its potential for large-scale implementation, helping to make a positive impact on society. Going forward, **Bhojan Saajha** can be further improved and scaled to achieve even greater outcomes.

### FUTURE ENHANCEMENTS:

The **Bhojan Saajha** app can be enhanced in several ways to increase its functionality and reach:

1. **Broader Donation Categories:** Expand the platform to support donations beyond food, such as clothes, books, and other essentials.
2. **User Interface and Experience Enhancements:** Based on user feedback, further refine the UI/UX to improve ease of use, visual appeal, and overall experience.
3. **Advanced Analytics and Reporting:** Integrate more advanced analytics to track food donations, identify trends, and report on the impact of the platform, helping both donors and recipients.
4. **Geographical Expansion:** Expand the app to other cities, regions, or even countries to increase its reach and impact, providing more people access to surplus food.
5. **Collaborations with NGOs and Local Organizations:** Build partnerships with non-governmental organizations and local community groups to streamline food distribution and increase visibility of the platform.
6. **Sustainability Initiatives:** Add features to promote sustainable practices among users and donors, such as tracking the environmental impact of reducing food waste.

These enhancements will further strengthen **Bhojan Saajha's** role in reducing food wastage and fostering community welfare while scaling up to support a broader range of social needs.

# ***Bibliography***

## 11. BIBLIOGRAPHY

---

### 1) Books Referred:

- ☉ Schildt, H. (2019). *Java: The Complete Reference*. 11th Edition. McGraw-Hill Education.
- ☉ Phillips, B., Stewart, C., Hardy, K., & Marsicano, B. (2018). *Android Programming: The Big Nerd Ranch Guide*. 4th Edition. Big Nerd Ranch Guides.
- ☉ Craig Walls. (2016). *Spring Boot in Action*. Manning Publications.

### 2) Websites Referred:

- ☉ **Android Developers.** (n.d.). *Android SDK Documentation*. Retrieved from <https://developer.android.com>
- ☉ **Firebase Documentation.** (n.d.). *Firebase Realtime Database*. Retrieved from <https://firebase.google.com/docs/database>
- ☉ **Spring Framework Documentation.** (n.d.). *Spring Boot Framework Guide*. Retrieved from <https://spring.io/projects/spring-boot>
- ☉ **Gradle Documentation.** (n.d.). *Gradle Build Tool User Guide*. Retrieved from <https://gradle.org/documentation/>
- ☉ **JUnit Documentation.** (n.d.). *JUnit 5 User Guide*. Retrieved from <https://junit.org/junit5/docs/current/user-guide/>
- ☉ **Espresso Testing Documentation.** (n.d.). *Espresso Testing Framework for Android*. Retrieved from <https://developer.android.com/training/testing/espresso>
- ☉ **Google Maps Platform Documentation.** (n.d.). *Maps SDK for Android*. Retrieved from <https://developers.google.com/maps/documentation/android-sdk>
- ☉ **W3Schools.** (n.d.). *Java Tutorial*. Retrieved from <https://www.w3schools.com/java/>
- ☉ **Stack Overflow.** (n.d.). *Android Firebase Integration Q&A*. Retrieved from <https://stackoverflow.com/questions/tagged/firebase>