1. Introduction

The QrToConnect application is an innovative Android platform to contact management, shearing and communication. Built upon the Firebase Realtime Database , Firebase Cloud Messaging(FCM) and leveraging the Google ZXing library for QR code scanning, this app revolutionizes the way users handle their contacts.

With a focus on enhancing interpersonal connections, QrToConnect introduces cutting-edge features including chatting and group functionality. Users can effortlessly create groups, facilitating seamless communication among saved contacts. Additionally, the app supports image sharing within conversations, ensuring a dynamic and engaging interaction experience.

Enhanced with push notifications, users stay informed in real-time when receiving messages, images, or calls. The integration of ZEGOCLOUD further enriches the experience, enabling high-quality video and audio calls over the internet.

Beyond online communication, QrToConnect ensures connectivity even in offline scenarios through traditional mobile networks. Users can also utilize the app to send SMS messages and emails, expanding the avenues of communication.

Furthermore, the app incorporates a dial pad keypad, empowering users to initiate calls to contacts not using the application. This seamless integration enhances accessibility and ensures connectivity regardless of the recipient's platform.

An exciting addition is the creation of broadcast channels, offering users the ability to share messages and images with a broader audience. Users can request to join these channels, curated by administrators for a personalized and secure interaction space. Moreover, the global search functionality enables users to discover and join relevant broadcast channels effortlessly.

With a renewed focus on user experience, QrToConnect boasts an improved UI/UX design, ensuring intuitive navigation and enhanced user engagement.

Key features of QrToConnect include:

- Firebase authentication
- Adding and Deleting contacts
- Blocking contacts
- Searching for contacts
- Sharing contacts
- Adding Contacts into Favorite
- Scanning QR codes to add contacts
- Group messaging and chatting
- Image sharing in chats
- Real-time push notifications for messages, images, and calls
- High-quality video and audio calls via ZEGOCLOUD
- Offline communication through SMS and email
- Dial pad keypad for calling non-app users
- Creation and management of broadcast channels
- Global search for relevant broadcast channels

1.1 Project Profile :-

Project Title:	**QrToConnect**
Project Type:	Mobile Application
Technology:	Android, Java
Front End:	XML, Android Layouts
Back End:	Firebase Real-time Database
Framework:	Android SDK
Internal Guide:	Prof. Nilam Patel
Submitted To:	Shree Uttar Gujarat BCA Collage
Developed By:	Akshay Raval (4438) Krishna Kapadia (4333) Rudra Rathod (4437)

2. Environment Description

2.1 Hardware and Software Requirements:-

The QrToConnect Android Application requires specific hardware and software resources to function optimally.

Hardware Requirements:

1. **Processor**:

- Explanation: The processor acts as the brain of the device, responsible for executing instructions and tasks. For your application, which involves decoding QR codes, processing user interactions, and rendering the user interface, a processor with a clock speed of 1.5 GHz or higher ensures smooth and responsive performance. This ensures that users can quickly scan QR codes, navigate through the app, and perform various actions without experiencing delays or lags.

- Usage in Application: The processor is utilized during various tasks such as QR code decoding, data processing, and rendering UI elements. A faster processor ensures that these tasks are executed swiftly, providing a seamless user experience.

2. Memory (RAM):

- Explanation: RAM (Random Access Memory) is used by the operating system and applications to store data temporarily while they are in use. For your application, which may run alongside other background processes and services, a minimum of 2 GB of RAM ensures efficient multitasking without performance degradation. This allows users to switch between your app and other apps smoothly without experiencing slowdowns or app crashes.

- Usage in Application: The application utilizes RAM to store app data, cache images, and maintain the app's state. Having sufficient RAM ensures that the app can handle multiple tasks simultaneously, such as scanning QR codes, displaying contacts, and interacting with the user interface.

3. **Storage**:

- Explanation: Storage space is required to store the application itself, app data, user-generated contacts, cached images, and other files. While the app itself requires minimal storage space (typically around 50 MB), additional storage may be necessary to accommodate a considerable number of contacts generated from QR codes. Therefore, having ample storage space ensures that users can use the app without worrying about running out of storage.

- Usage in Application: The application utilizes storage space to store app data, including user-generated contacts, cached images, and temporary files. Users can save contacts generated from QR codes directly to their device's storage, allowing them to access and manage contacts offline.

Software Requirements:

1. Android OS (Operating System):

- Explanation: The application is designed to run on Android devices with a minimum OS version of Android 7.0 (Nougat). This ensures compatibility with a wide range of devices and allows you to leverage newer features and APIs provided by the Android platform. Targeting Android 7.0 or higher also ensures that users with older devices can still access and use the application, providing broader accessibility.

- Usage in Application: The application utilizes features and APIs introduced in Android 7.0 and higher to enhance functionality, improve performance, and ensure compatibility with modern Android devices. By targeting this OS version, you can provide a better user experience and take advantage of the latest Android developments.

2. App Permissions:

- Explanation: App permissions are necessary for the proper functioning of the application and to ensure user privacy and security. Common permissions requested by the QrToConnect App include access to contacts, storage, and network connectivity. These permissions enable the app to perform essential tasks such as saving contacts, storing data, and accessing online services.

  - CALL_PHONE: This permission allows the application to initiate phone calls without user intervention. It's required if the app includes features that allow users to make phone calls directly from the app.

  - SEND_SMS: This permission allows the application to send SMS (text) messages. It's necessary if the app includes features that allow users to send text messages directly from the app.

  - RECEIVE_SMS: This permission allows the application to receive SMS messages. It's needed if the app includes features that involve receiving SMS messages, such as OTP verification or message-based authentication.

  - VIBRATE: This permission allows the application to control the device's vibration motor. It's used to provide haptic feedback to users, such as vibrating notifications.

  - CAMERA: This permission grants the application access to the device's camera. It's required if the app includes features that involve capturing photos or scanning QR codes using the device's camera.

  - ACCESS_WIFI_STATE: This permission allows the application to access information about Wi-Fi networks. It's used to determine the status and properties of the device's Wi-Fi connection.

- INTERNET: This permission allows the application to access the internet. It's necessary for the app to communicate with remote servers, download data, and access online services.

- READ_EXTERNAL_STORAGE: This permission allows the application to read from external storage (e.g., SD card). It's required if the app needs to access files stored on the device's external storage.

- WRITE_EXTERNAL_STORAGE: This permission allows the application to write to external storage (e.g., SD card). It's necessary if the app needs to create, modify, or delete files on the device's external storage.

- ACCESS_NETWORK_STATE: This permission allows the application to access information about the device's network state, such as whether the device is connected to a network and the type of network connection (e.g., Wi-Fi, mobile data).

- FOREGROUND_SERVICE_DATA_SYNC: This permission allows the application to access data sync services while running in the foreground. It's needed if the app requires continuous data synchronization with a remote server or backend service.

- FOREGROUND_SERVICE: This permission allows the application to run foreground services. Foreground services are used for tasks that are noticeable to the user, such as playing music or handling ongoing notifications.

- ACCESS_NOTIFICATION_POLICY: This permission allows the application to access the notification policy, including Do Not Disturb settings. It's required if the app needs to modify notification behavior or interact with system notifications.

- WAKE_LOCK: This permission allows the application to prevent the device from entering sleep mode. It's used if the app needs to keep the device awake to perform background tasks or maintain certain functionalities.

- Usage in Application: The application requests permissions from the user to access contacts, storage, and network connectivity. These permissions allow the app to scan QR codes, save contacts to the device's address book, store app data, and communicate with external servers or APIs.

3. APIs and Libraries:

- Explanation: The application utilizes various Android APIs and libraries provided by Google and third-party developers to enhance functionality, improve performance, and streamline development. These APIs and libraries encompass a wide range of functionalities, including location services, image loading, and UI components.

- Usage in Application: Specific APIs and libraries used in your application may include:

- Location Services API: Used for displaying nearby contacts or generating location-based QR codes.

- Image Loading Libraries (e.g., Picasso, Glide): Used for efficient image loading and caching, allowing the app to display images retrieved from QR codes or user avatars.

- UI Components: Android UI components and libraries are used to create the application's user interface, including layouts, widgets, and navigation components. These components ensure a consistent and intuitive user experience across different devices and screen sizes.

2.2 Technologies Used:-

The QrToConnect Android Application leverages several key technologies to deliver a robust and user-friendly experience.

1. Android Operating System:

- The foundation of the application is the Android operating system. It provides the necessary framework and ecosystem for app development, ensuring compatibility with a wide range of Android devices.
- The application targets Android 7.0 (Nougat) and above to reach a broad user base while taking advantage of the latest features and security enhancements.

2. XML (Extensible Markup Language):

- XML serves as the backbone for structuring the user interface (UI) in your application, offering a versatile and customizable approach to designing layouts. With XML, developers can define the arrangement and appearance of various UI elements with precision.
- Its hierarchical nature enables the nesting of elements, allowing for complex and visually appealing layouts. Furthermore, XML supports attribute-based styling, enabling developers to fine-tune the appearance of UI components such as colors, fonts, and dimensions.
- Through XML, your application achieves a consistent and visually appealing user experience, enhancing usability and user engagement.

3. **Java**:

- Java stands as the powerhouse behind the backend logic and functionality of your application, providing a robust and versatile programming language for Android development.
- Leveraging Java's object-oriented paradigm, developers can implement intricate algorithms, manage data structures efficiently, and orchestrate the application's lifecycle seamlessly.
- Java's rich ecosystem of libraries and frameworks further accelerates development, offering solutions for tasks ranging from network communication to data processing. By harnessing the power of Java, your application delivers a reliable, scalable, and performant user experience, ensuring smooth operation across a diverse range of devices and scenarios.

4. **Firebase**:

- Firebase emerges as the cornerstone of your application's backend infrastructure, offering a comprehensive suite of services for mobile and web application development.

- Firebase Realtime Database serves as the heart of your application, facilitating real-time data synchronization and seamless collaboration between users. Its NoSQL data model empowers developers to store and retrieve data with ease, while its synchronization capabilities ensure that updates propagate instantly across all connected devices.
- Complementing the Realtime Database, Firestore provides advanced querying and scaling capabilities, accommodating the evolving needs of your application as it scales.

- Moreover, Firebase Authentication safeguards user data and privacy, offering secure authentication mechanisms such as email/password, social logins, and phone authentication. With Firebase Analytics, developers gain valuable insights into user behavior and engagement, informing data-driven decisions to optimize the application's performance and user experience.

- In essence, Firebase empowers your application with a robust, scalable, and secure backend infrastructure, enabling seamless communication, collaboration, and data management.

5. **ZEGOCLOUD**:

- ZEGOCLOUD emerges as a pivotal component in augmenting your application's communication capabilities, offering a reliable and feature-rich platform for video and audio calling. Leveraging ZEGOCLOUD's SDKs and APIs, developers can seamlessly integrate high-quality video and audio calling functionalities into their applications, enhancing real-time communication experiences for users.

- ZEGOCLOUD's advanced features, such as screen sharing and chat integration, further enrich the communication ecosystem, empowering users with versatile tools for collaboration and interaction. Moreover, ZEGOCLOUD's cross-platform compatibility ensures broad accessibility across devices and operating systems, fostering inclusivity and connectivity among users.

- By incorporating ZEGOCLOUD into your application, you elevate the communication experience to new heights, enabling seamless, immersive, and engaging interactions between users.

6. SMS and Email Sending:

- The integration of SMS and email sending functionalities enriches your application with additional avenues for communication, extending its reach and accessibility to users.

- Through Android's built-in APIs and possibly third-party libraries or services, developers can seamlessly incorporate SMS and email sending capabilities into the application's feature set. Users can leverage these functionalities to communicate with contacts who may not be using the application or to engage in asynchronous communication channels.

- Whether sending urgent notifications, sharing important updates, or facilitating external communication, SMS and email capabilities enhance the versatility and utility of your application, empowering users with flexible communication options tailored to their preferences and needs.

7. Firebase Cloud Messaging (FCM):

- Firebase Cloud Messaging (FCM) emerges as a vital component in enabling real-time notifications and updates for your application, ensuring timely communication and engagement with users.

- FCM's robust infrastructure facilitates the delivery of push notifications across a diverse array of devices and platforms, ensuring broad reach and reliability. Developers can leverage FCM's intuitive APIs and dashboard to orchestrate targeted messaging campaigns, segment users based on behavior and preferences, and track key engagement metrics in real-time.

- By integrating FCM into your application, you establish a direct channel of communication with users, keeping them informed, engaged, and connected to the latest updates and developments. Whether notifying users of incoming messages, updates, or events, FCM empowers your application with the agility and responsiveness needed to thrive in today's dynamic digital landscape.

8. QR Code Generation and Scanning:

- QR code generation and scanning functionalities enhance the interoperability and convenience of your application, enabling seamless data exchange and contact management. Leveraging third-party libraries such as ZXing (Zebra Crossing) for Android, developers can effortlessly integrate QR code generation and scanning capabilities into the application's feature set.

- Users can generate QR codes containing contact information, URLs, or other data types, simplifying the process of sharing and exchanging information with others. Similarly, users can scan QR codes to add new contacts or access relevant content, eliminating the need for manual data entry and streamlining the user experience.

- By harnessing the power of QR technology, your application facilitates efficient data transfer, enhances user productivity, and fosters seamless connectivity in today's digital ecosystem.

3. System Requirement Specification

3.1 Data Dictionary :-

- **User Details Table :-**

Field Name	Data Type	Description
firstname	String	First name of the user.
lastname	String	Last name of the user.
mobile	String	Mobile phone number of the user.
email	String	Email address of the user.
imageURL	String	URL of the image associated with the user.
status	String	Status of the user.
id	String	Unique identifier.
typing	String	Indicates if the user is currently typing.
token	String	Token for authentication and authorization.
fullname	String	Full name of the user.
password	String	Password for authentication.
companyName	String	Name of the company the user works for.
unreadMessage	int	Number of unread messages for the user.
chatLastMessageModel	ChatLastMessageModel	Object containing information about the last message in a chat.
chatUnreadMessageModel	ChatUnreadMessageModel	Object containing information about the number of unread messages in a chat.

- **Contact Details Table :-**

Field Name	Data Type	Description
contactEmail	String	Email address of the contact.
contactFirstName	String	First name of the contact.
contactLastName	String	Last name of the contact.
contactMobile	String	Mobile phone number of the contact.
contactCompanyName	String	Name of the company the contact works for.
contactNotes	String	Additional notes about the contact.
imageURL	String	URL of the image associated with the contact.
key	String	Unique identifier for the contact.

- **Broadcast Channel Member Table :-**

Field Name	Data Type	Description
id	String	Unique identifier for the broadcast channel member.
role	String	Role of the broadcast channel member.
token	String	Token for authentication and authorization.

- **Broadcast Channel Table :-**

Field Name	Data Type	Description
adminId	String	Unique identifier for the broadcast channel administrator.
adminName	String	Name of the broadcast channel administrator.
channelId	String	Unique identifier for the broadcast channel.
channelImage	String	URL of the image associated with the broadcast channel.
channelName	String	Name of the broadcast channel.
role	String	Role of the user in the broadcast channel.

- **Channel Request Table :-**

Field Name	Data Type	Description
reqUserId	String	Unique identifier for the user requesting to join the channel.
reqChannelId	String	Unique identifier for the channel being requested to join.
reqImage	String	URL of the image associated with the user requesting to join the channel.
reqName	String	Name of the user requesting to join the channel.
reqChannelName	String	Name of the channel being requested to join.
status	String	Status of the request.
reqToken	String	Token for authentication and authorization.
creatorUserId	String	Unique identifier for the user who created the channel.

- **Chat Last Message Table :-**

Field Name	Data Type	Description
senderId	String	Unique identifier for the user who sent the last message in a chat.
message	String	Content of the last message in a chat.
date	String	Date and time when the last message in a chat was sent.

- **Chat Unread Message Table :-**

Field Name	Data Type	Description
senderId	String	Unique identifier for the user who sent the unread message.
message	String	Content of the unread message.
date	String	Date and time when the unread message was sent.
senderRoom	String	Unique identifier for the room where the unread message was sent.

- **Group Last Message Table :-**

Field Name	Data Type	Description
senderId	String	Unique identifier for the user who sent the last message in a group.
message	String	Content of the last message in a group.
date	String	Date and time when the last message in a group was sent.
type	String	Type of the last message in a group.

- **Group Member Table :-**

Field Name	Data Type	Description
id	String	Unique identifier for the group member.
role	String	Role of the group member.
token	String	Token of the group member.
unreadMessage	int	Number of unread messages for the group member.

- **Message Table :-**

Field Name	Data Type	Description
uId	String	Unique identifier for the message.
message	String	Content of the message.
messageId	String	Unique identifier for the message in the database.
type	String	Type of the message (e.g. text, image, file).
isSelected	Boolean	Flag indicating whether the message is selected or not.
timestamp	Long	Timestamp of the message.

- **All Broadcast Channel :-**

Field Name	**Data Type**	**Description**
adminId	String	Unique identifier for the administrator of the broadcast channel.
adminName	String	Name of the administrator of the broadcast channel.
channelId	String	Unique identifier for the broadcast channel.
channelImage	String	URL of the image for the broadcast channel.
channelName	String	Name of the broadcast channel.
role	String	Role of the user in the broadcast channel.
token	String	Token for authentication and authorization.

- **Group Message Table :-**

Field Name	**Data Type**	**Description**
type	String	Type of the message (e.g. text, image, file).
message	String	Content of the message.
date	String	Date and time when the message was sent.
senderId	String	Unique identifier for the sender of the message.
name	String	Name of the sender of the message.
resized	boolean	Flag indicating whether the image message has been resized or not.
imageWidth	int	Width of the image message after resizing.
imageHeight	int	Height of the image message after resizing.

- **Group Table :-**

Field Name	**Data Type**	**Description**
id	String	Unique identifier for the group.
adminId	String	Unique identifier for the administrator of the group.
adminName	String	Name of the administrator of the group.
createAt	String	Date and time when the group was created.
image	String	URL of the image for the group.
name	String	Name of the group.
members	List<GroupMemberModel>	List of members in the group.
lastMessageModel	GroupLastMessageModel	Object containing information about the last message in the group.
isAdmin	boolean	Flag indicating whether the user is an administrator of the group or not.
unreadMessage	int	Number of unread messages in the group.

- **Block Contact Model Table :-**

Attribute Name	Type	Description
blockEmail	String	Email address of the blocked contact.
blockFirstName	String	First name of the blocked contact.
blockLastName	String	Last name of the blocked contact.
blockMobile	String	Mobile number of the blocked contact.
blockCompanyName	String	Company name of the blocked contact.
blockNotes	String	Notes about the blocked contact.
blockImageURL	String	URL of the image associated with the blocked contact.
blockKey	String	Unique identifier for the blocked contact.

3.2 Feasibility Analysis :-

3.2.1 Economic Feasibility :-

Economic feasibility analysis is a crucial aspect of evaluating the viability of any software application. It involves assessing whether the benefits derived from the application outweigh the costs associated with its development and maintenance. In the case of your application, let's delve into the economic feasibility to provide insights into its potential financial viability:

1. Cost of Development: The economic feasibility analysis begins with an examination of the costs involved in developing the application. This encompasses expenses such as software development tools, hardware infrastructure, salaries of developers, and any other resources required during the development phase. By meticulously estimating these costs, you can gain a clearer understanding of the initial investment required.

2. Revenue Generation: Understanding how the application will generate revenue is paramount. Consider whether the application will be monetized through direct sales, subscription models, in-app purchases, advertisements, or any other revenue streams. Each revenue model comes with its own set of considerations, such as user adoption rates, pricing strategies, and market competitiveness.

4. **Market Potentia**l: Conducting market research to gauge the demand for your application is essential. Analyze factors such as target audience size, demographics, purchasing power, and competition within the market. By identifying niche markets or unmet needs, you can position your application for greater success and potentially higher returns on investment.

4. Return on Investment (ROI): Assessing the potential return on investment is critical for determining the economic viability of the application. Calculate the projected revenue against the initial development costs and ongoing expenses (such as maintenance, updates, and marketing). A favorable ROI indicates that the application has the potential to generate profits and recoup the initial investment within a reasonable timeframe.

5. Scalability and Longevity: Consider the scalability and longevity of the application in the marketplace. Will it be able to adapt to changing technological trends, user preferences, and competitive landscapes? Investing in features that enhance scalability and longevity can contribute to the sustained success of the application and maximize its economic feasibility over time.

6. Risk Assessment: Evaluate potential risks and uncertainties that may impact the economic feasibility of the application. This includes factors such as market volatility, changing regulatory landscapes, technological disruptions, and unforeseen challenges during development and deployment. Mitigation strategies should be devised to address these risks and minimize their impact on the project's financial outcomes.

7. Cost-Benefit Analysis: Finally, conduct a comprehensive cost-benefit analysis to weigh the potential benefits against the costs associated with developing and maintaining the application. By comparing the projected financial gains with the investment required, you can make informed decisions about the economic viability of pursuing the project further.

3.2.2 Technical Feasibility :-

Technical feasibility is a crucial aspect to consider when evaluating the viability of a software application. It involves assessing whether the proposed system can be successfully developed, implemented, and maintained with the available technology resources and infrastructure. In the case of your application, several technical factors contribute to its feasibility:

1. Platform Compatibility: Your application appears to be developed for the Android platform using Java and Firebase services. Android is a widely used platform, and Java is a robust programming language supported by a large community of developers. Additionally, Firebase provides a comprehensive suite of backend services, including authentication, real-time database, and cloud storage, which simplifies backend development and integration. The compatibility of these technologies ensures that your application can be developed effectively.

2. Integration with Third-Party Libraries: Your application utilizes several third-party libraries such as Glide for image loading and Karumi Dexter for handling permissions. These libraries enhance the functionality of your application by providing pre-built solutions for common tasks. The availability of these libraries indicates that your application can leverage existing resources to expedite development and reduce development effort.

3. Security Considerations: Security is paramount in any application, especially when handling user data. Firebase Authentication provides robust authentication mechanisms, ensuring secure access to user accounts. Additionally, Firebase Realtime Database and Firebase Storage offer built-in security rules to control access to data stored in the cloud. By leveraging Firebase's security features, your application can maintain the confidentiality and integrity of user data, enhancing its technical feasibility.

4. Scalability: As your application grows and attracts more users, scalability becomes a critical consideration. Firebase offers scalable backend services that can handle a large number of concurrent users and data requests. The distributed nature of Firebase ensures that your application can scale seamlessly as the user base expands, maintaining performance and reliability.

5. Cross-Device Compatibility: Android applications are designed to run on a variety of devices with different screen sizes, resolutions, and hardware configurations. Your application's user interface should be responsive and adaptable to ensure a consistent user experience across devices. By following Android's design guidelines and best practices, your application can achieve cross-device compatibility, enhancing its technical feasibility.

6. Development Effort and Complexity: Assessing the development effort and complexity is essential to determine the feasibility of implementing your application. The availability of well-documented APIs, development tools, and community support for Android and Firebase technologies simplifies development tasks and reduces complexity. Moreover, the modular structure of your application and the use of design patterns such as Model-View-ViewModel (MVVM) facilitate code organization and maintenance, further enhancing its technical feasibility.

3.2.3 Behavioral Feasibility :-

The behavioral feasibility of an application assesses its ability to meet the needs and expectations of its intended users in terms of functionality, usability, and user experience. Let's delve into the behavioral feasibility of the application represented by the provided Java files:

1. User Authentication and Onboarding Experience: The application offers a welcoming and intuitive user authentication process through email and password. This approach ensures that users can easily log in to the application and access its features without any friction. Additionally, the sign-up process provides a seamless experience for new users, allowing them to quickly create an account and start using the application.

2. User Interface (UI) Design and Navigation: The UI design of the application appears to be user-friendly and visually appealing, enhancing the overall user experience. The use of clear labels, input fields, buttons, and navigation elements facilitates smooth interaction with the application. Moreover, the navigation flow within the application seems logical and intuitive, guiding users to different sections and functionalities effortlessly.

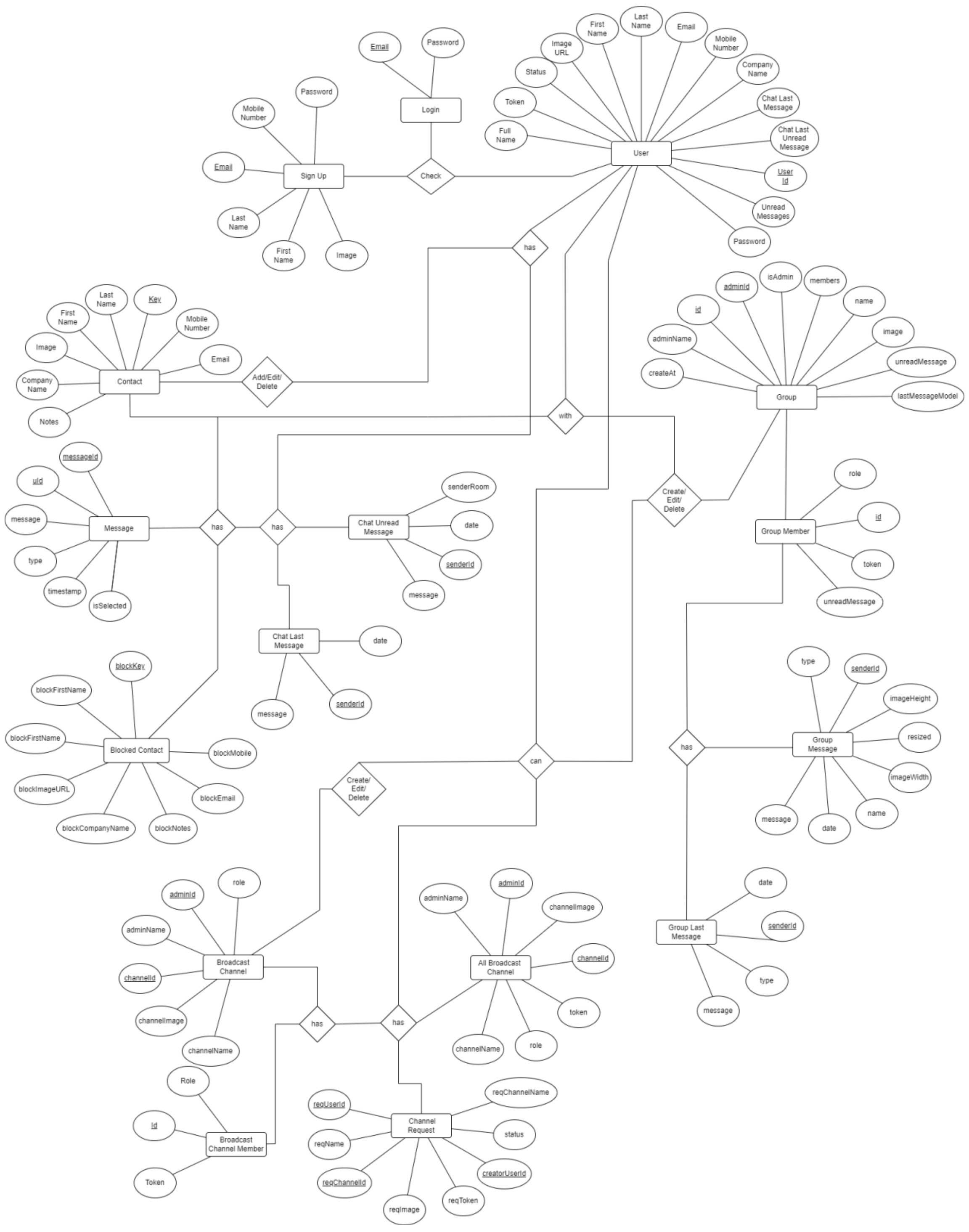
3. Contact Management and Updates: The application allows users to manage their contacts efficiently by providing features for adding, editing, and updating contact information. The inclusion of functionalities such as image uploading, contact details editing, and data synchronization with Firebase Database enhances the application's usability and utility. Users can easily update contact information and view their contacts' details without encountering any usability issues.

4. Permissions Management: The application appropriately handles permissions related to accessing device features such as the camera, gallery, contacts, and audio recording. By requesting necessary permissions at runtime using the Dexter library, the application ensures compliance with user privacy preferences and Android platform requirements. This proactive approach to permissions management contributes to a positive user experience and builds trust among users.

5. Feedback and Error Handling: The application incorporates mechanisms for providing feedback to users and handling errors gracefully. Toast messages, progress dialogs, and alert dialogs are used effectively to communicate important information, such as successful operations, required fields, and error notifications. Additionally, error handling is implemented robustly to prevent crashes and maintain application stability under various scenarios.

6. Integration with Firebase Services: The integration of Firebase Authentication, Realtime Database, and Storage services enables seamless data management and synchronization across multiple devices. Users can securely log in to the application, access their data from anywhere, and benefit from real-time updates and synchronization. This integration enhances the reliability and responsiveness of the application, contributing to a positive user experience.

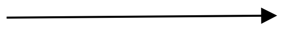
3.3 E-R Diagram :-



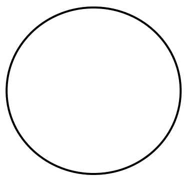
3.4 Data Flow Diagram :-

The Data Flow Diagrams Are An Intuitive Way Of Showning How Data Is Processed By a System. The Symbols Used In The DFD For This Project Are Shown Bellow.

- Data Flow:
  Following Symbol Is Used To Show Data Flow.The Data Flow Is Packet Of Data.



- Processes :
  People, Procedures or Devices That Use or Produce Data.

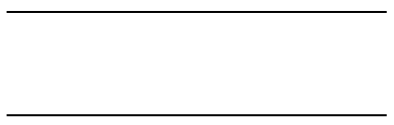


- Source or Destination Of Data :

  The Are Entities, Which Interact With The System From Outside Its Boundaries.

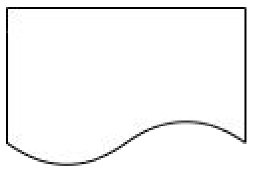
- Data Storage :

  Here Data Are Stored or Referenced Process In The System.

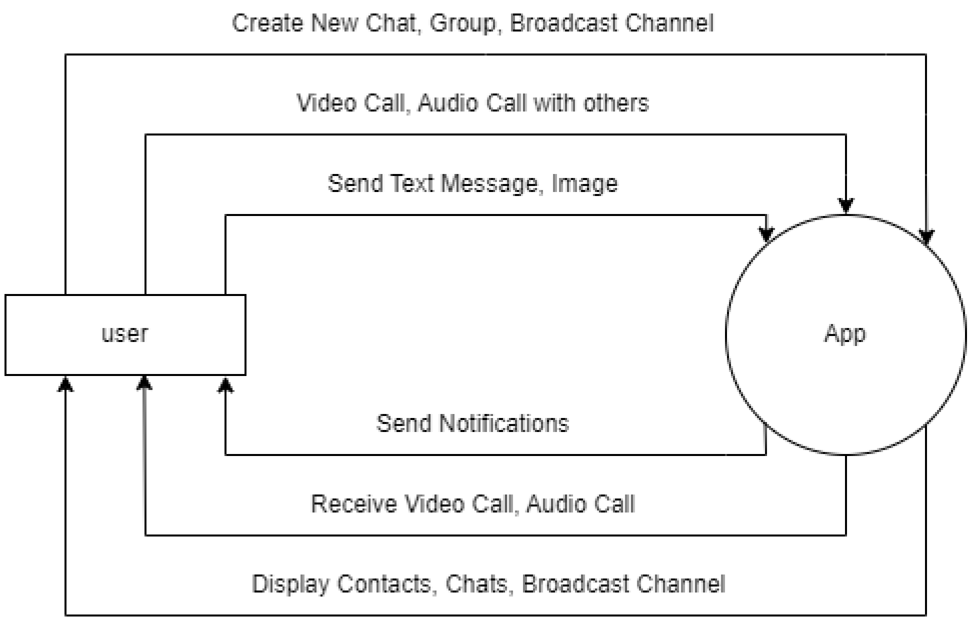


- Another Activity :

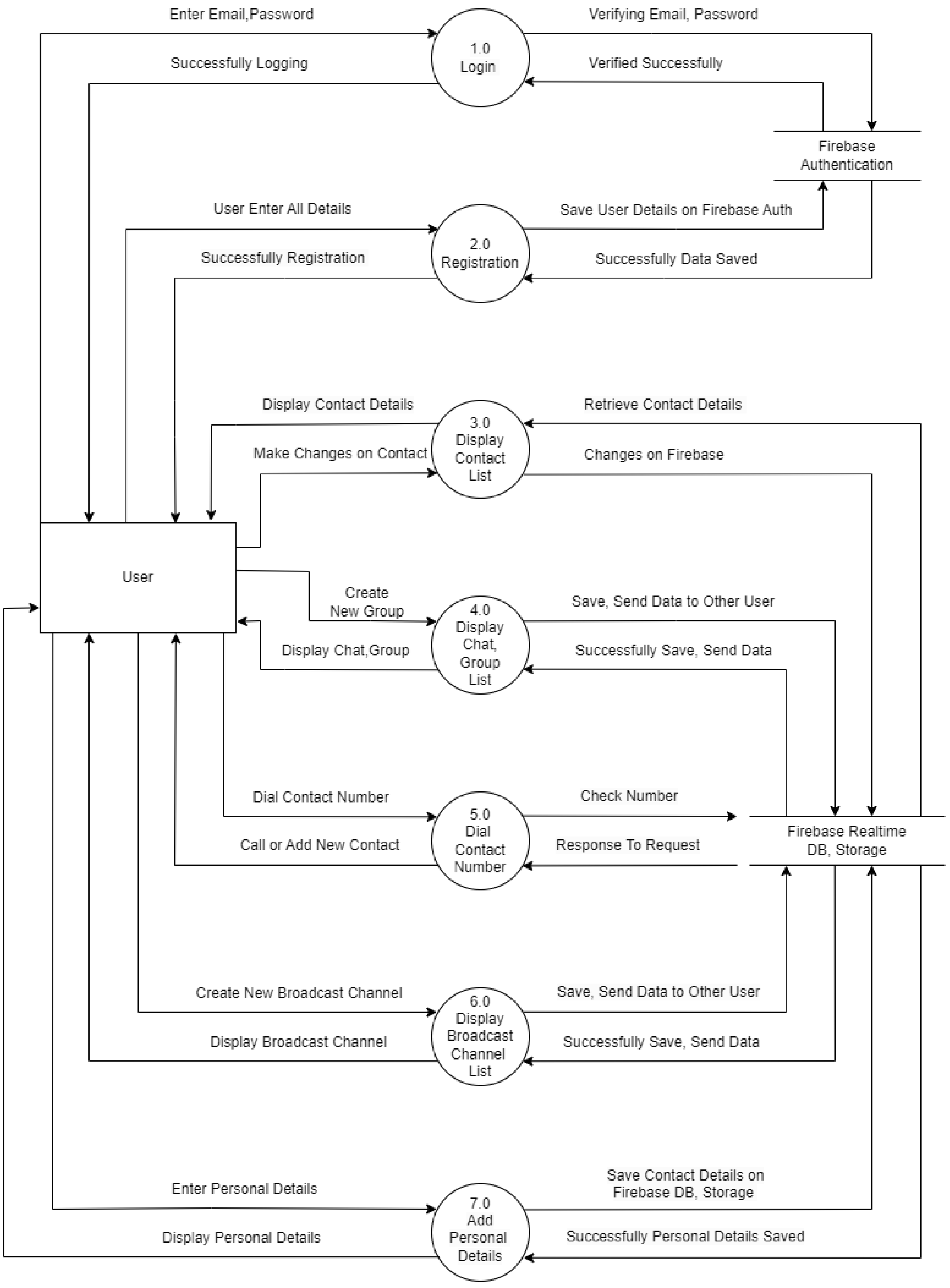
  This is use for redirecting / using another Activity of device/ Application



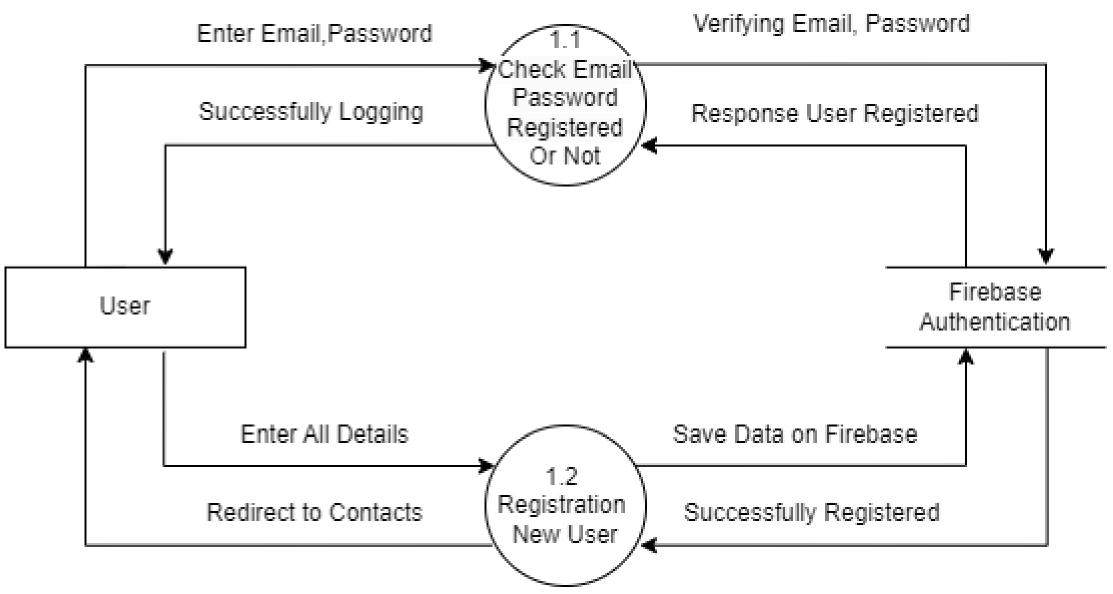
- **Context Level DFD, which can show basic flow of Application :-**


Create New Chat, Group, Broadcast Channel
Video Call, Audio Call with others
Send Text Message, Image
user
App
Send Notifications
Receive Video Call, Audio Call
Display Contacts, Chats, Broadcast Channel

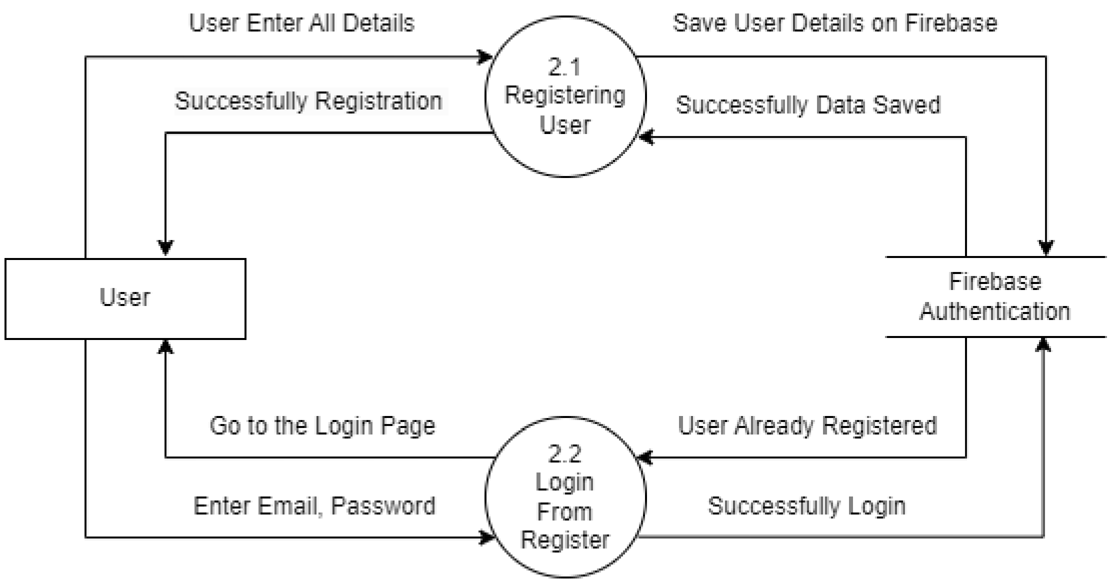
- **1st Level DFD, which can show main flow of Application :-**



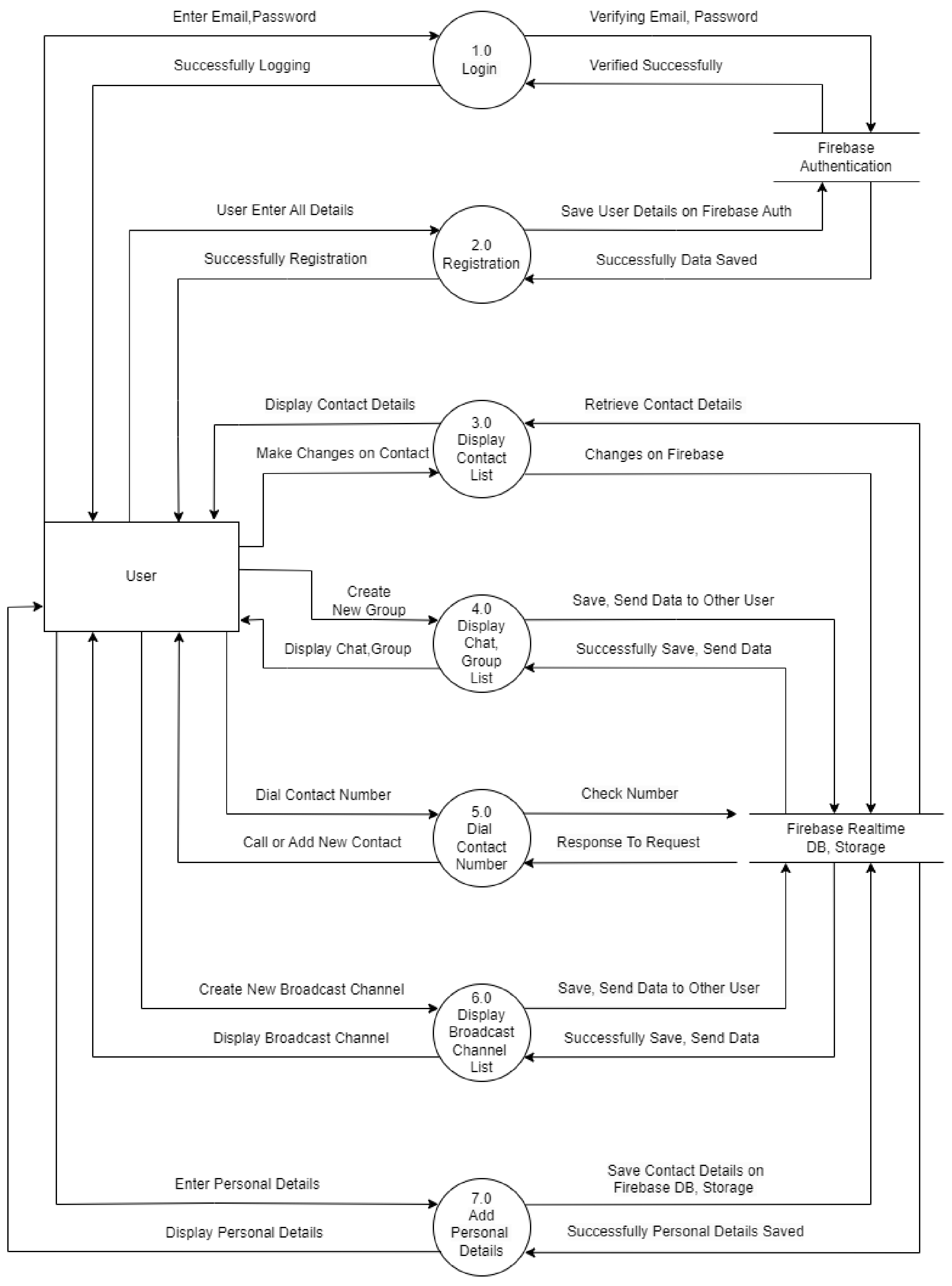
- **2nd Level DFD for Login Process :-**



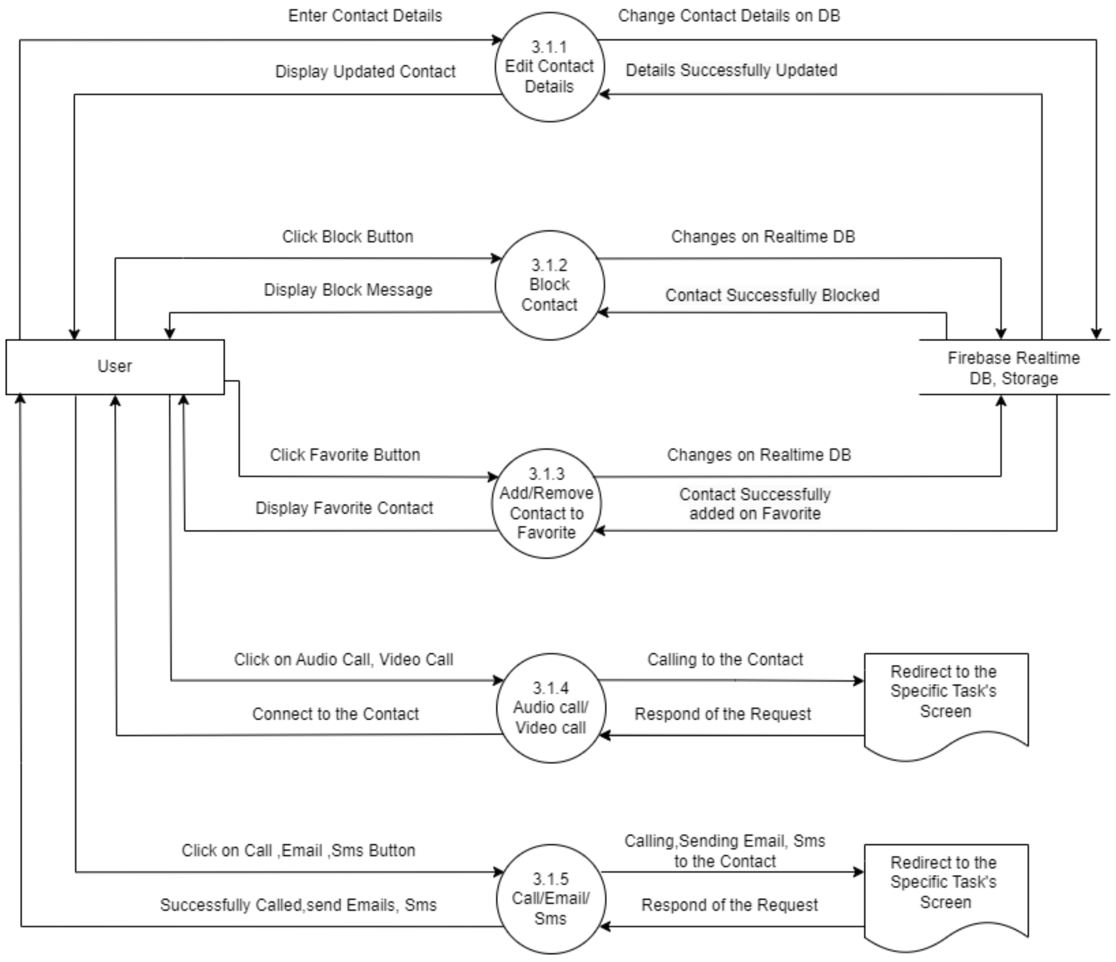
- **2nd Level DFD for Registration Process :-**



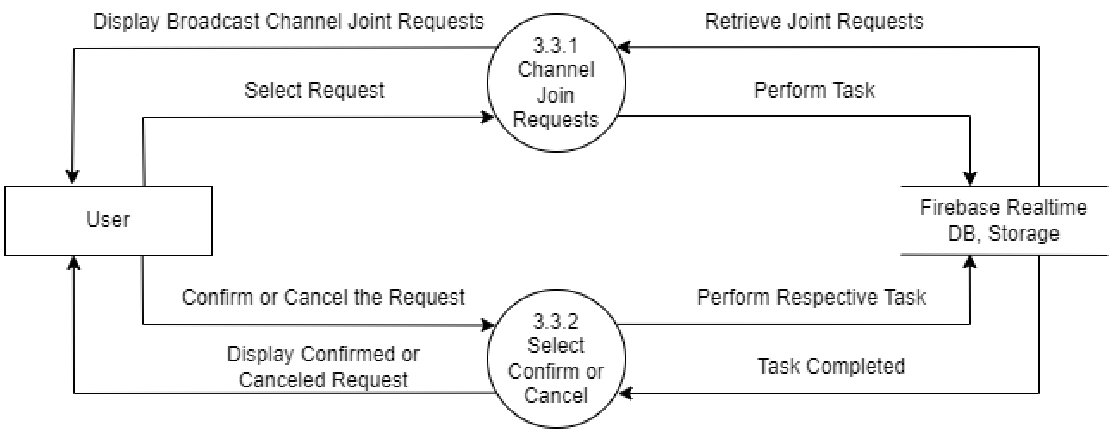
- **2nd Level DFD for Contact Fragment Process :-**



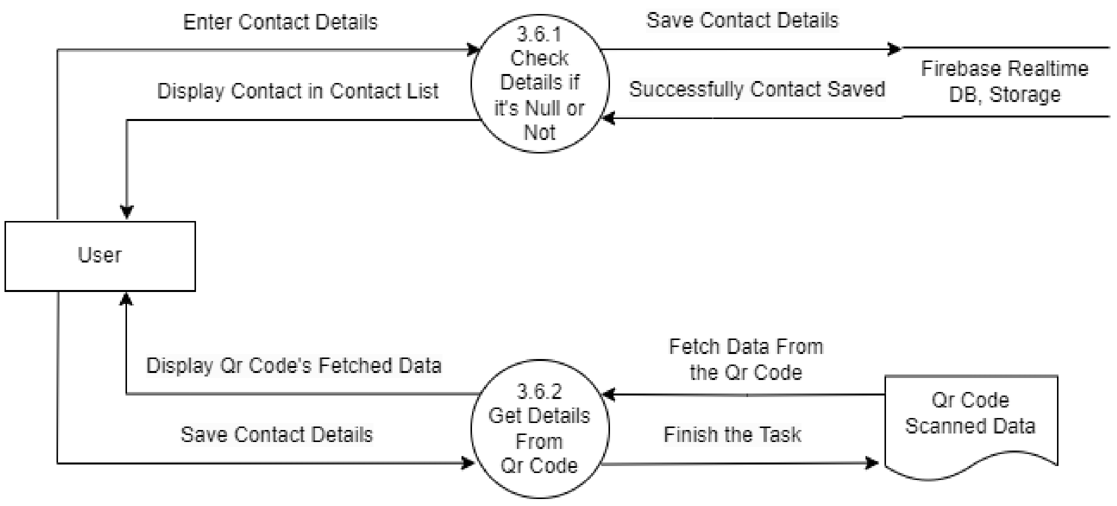
- **2nd Level DFD for Contact Details Process :-**


Enter Contact Details
Change Contact Details on DB
3.1.1 Edit Contact Details
Display Updated Contact
Details Successfully Updated
Click Block Button
Changes on Realtime DB
3.1.2 Block Contact
Display Block Message
Contact Successfully Blocked
User
Firebase Realtime DB, Storage
Click Favorite Button
Changes on Realtime DB
3.1.3 Add/Remove Contact to Favorite
Display Favorite Contact
Contact Successfully added on Favorite
Click on Audio Call, Video Call
Calling to the Contact
3.1.4 Audio call/ Video call
Connect to the Contact
Respond of the Request
Redirect to the Specific Task's Screen
Click on Call ,Email ,Sms Button
Calling,Sending Email, Sms to the Contact
3.1.5 Call/Email/ Sms
Successfully Called,send Emails, Sms
Respond of the Request
Redirect to the Specific Task's Screen

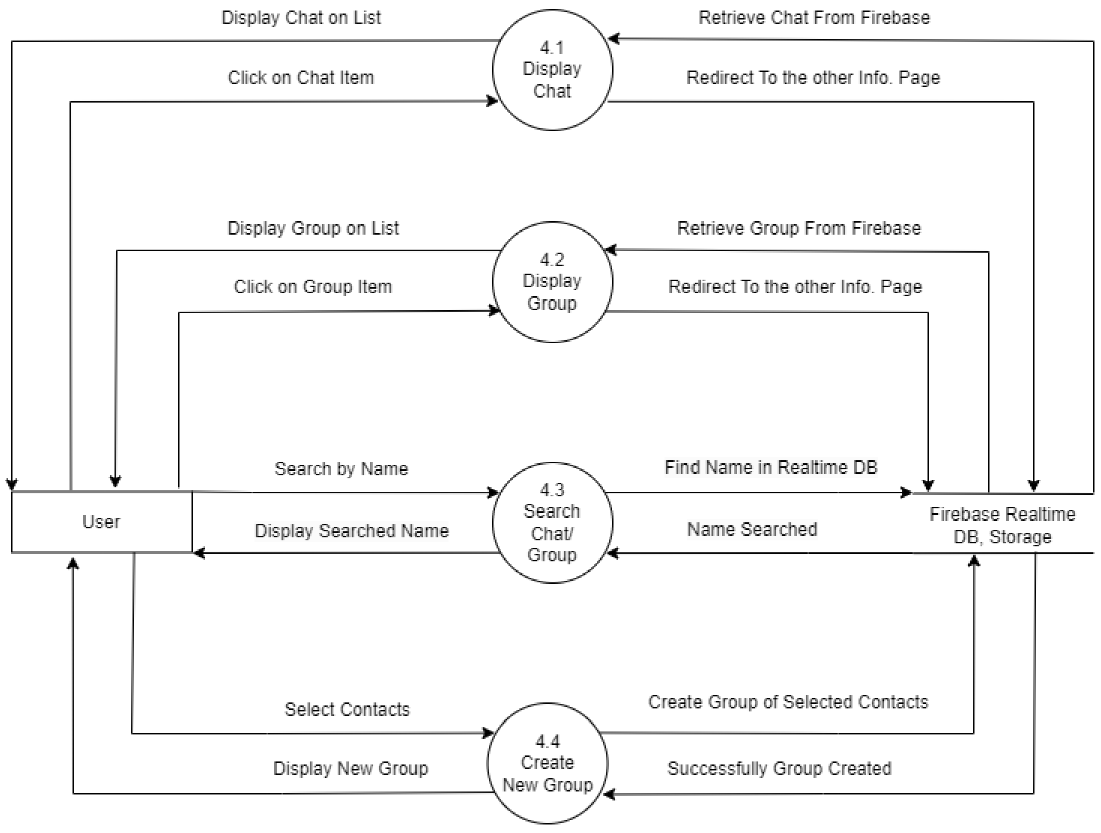
- **2nd Level DFD for Joining Request on Channel Process :-**


Display Broadcast Channel Joint Requests
3.3.1 Channel Join Requests
Retrieve Joint Requests
Select Request
Perform Task
User
Firebase Realtime DB, Storage
Confirm or Cancel the Request
Perform Respective Task
3.3.2 Select Confirm or Cancel
Display Confirmed or Canceled Request
Task Completed

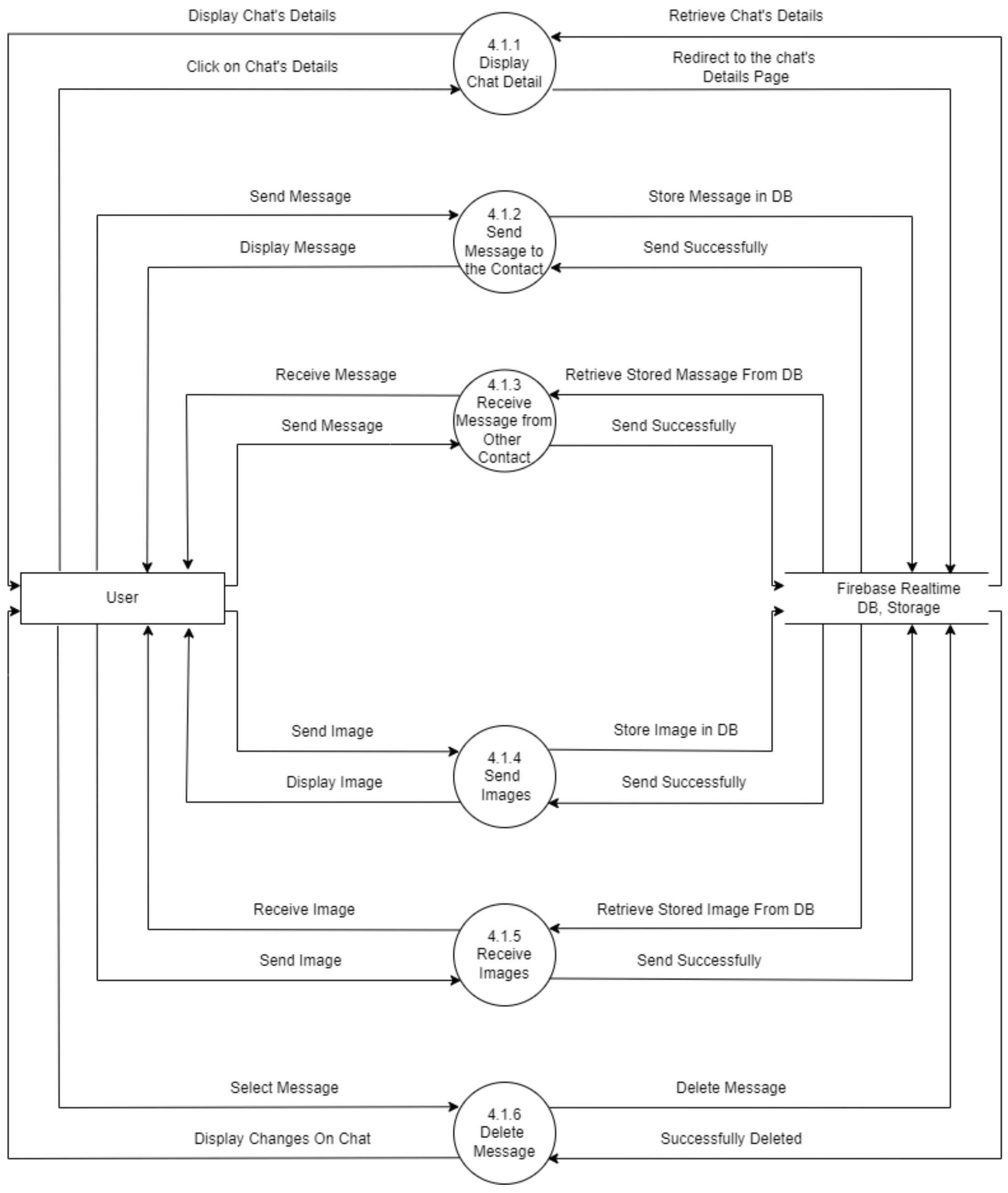
- **2nd Level DFD for Add Contact Process :-**


Enter Contact Details
3.6.1 Check Details if it's Null or Not
Save Contact Details
Display Contact in Contact List
Successfully Contact Saved
Firebase Realtime DB, Storage
User
Display Qr Code's Fetched Data
Fetch Data From the Qr Code
3.6.2 Get Details From Qr Code
Qr Code Scanned Data
Save Contact Details
Finish the Task

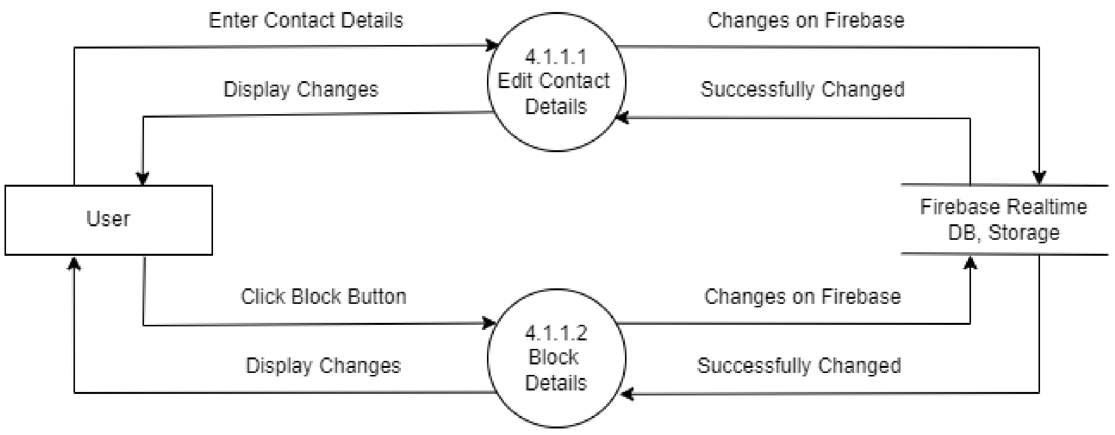
- **2nd Level DFD for Chat Fragment's Process :-**



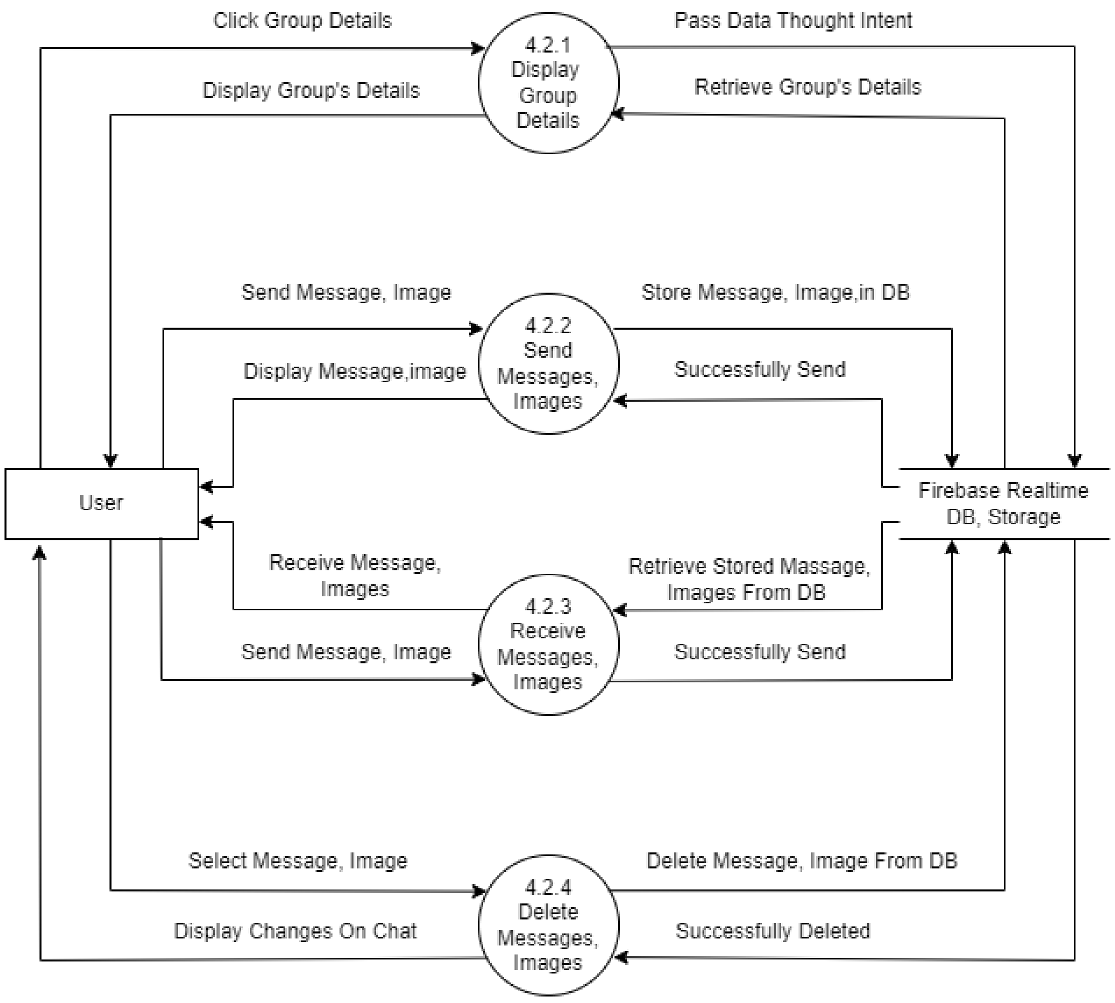
- **2nd Level DFD for Chat Messaging Process :**



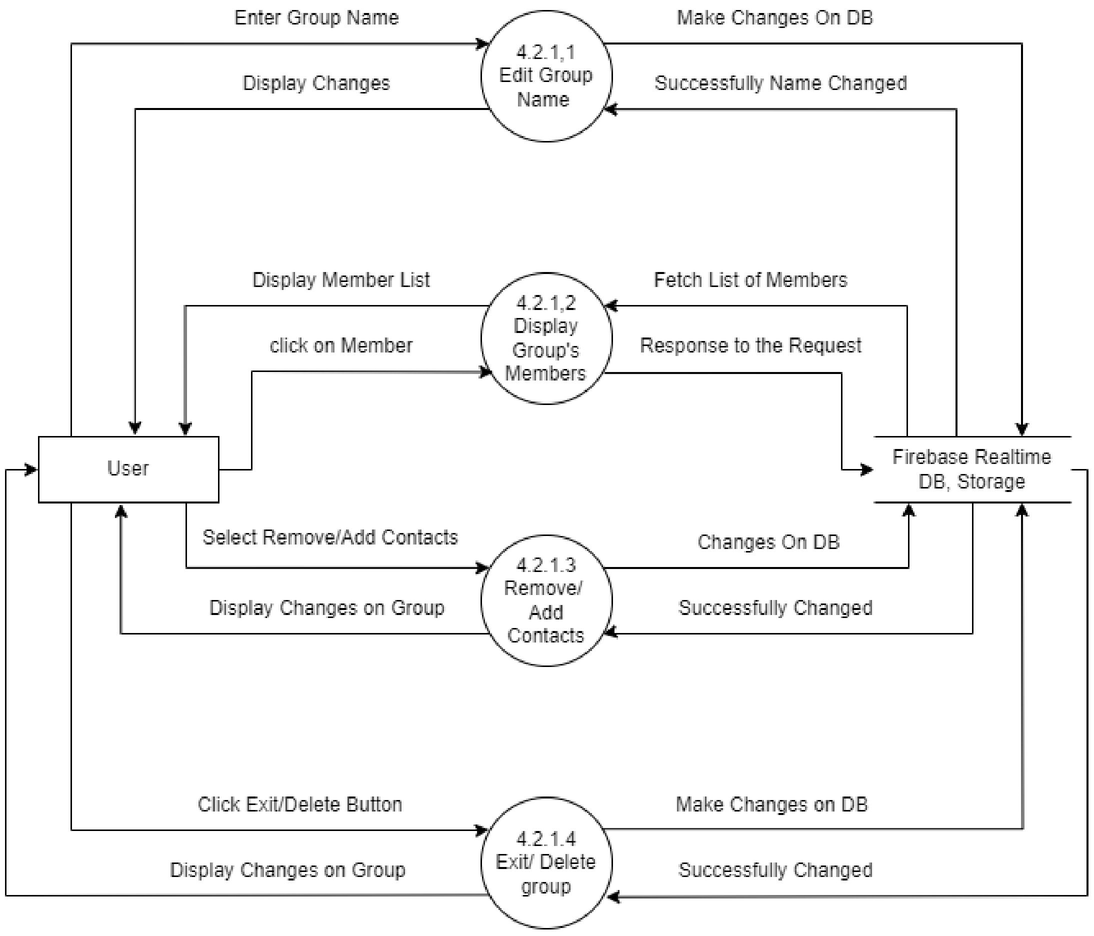
- **2nd Level DFD for Chat User's Details Process :-**



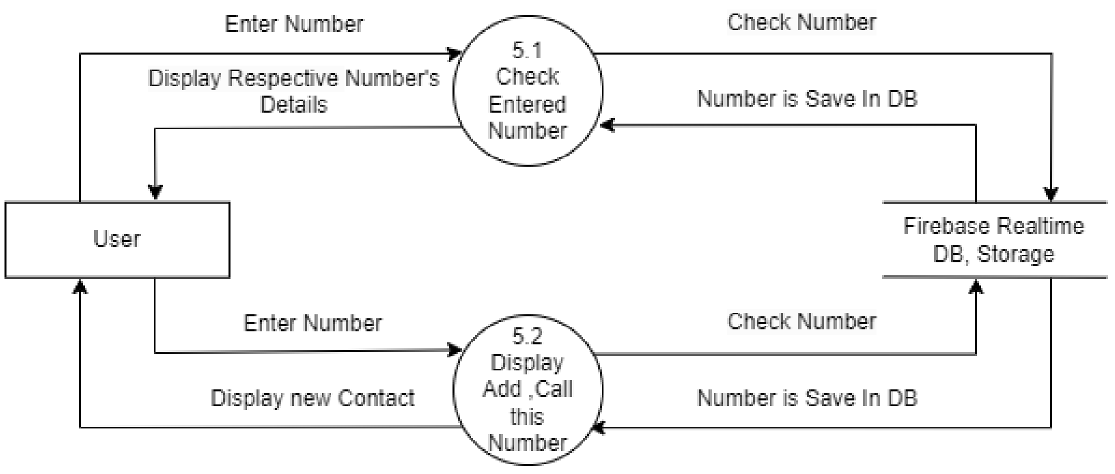
- **2nd Level DFD for Group Messaging Process :-**



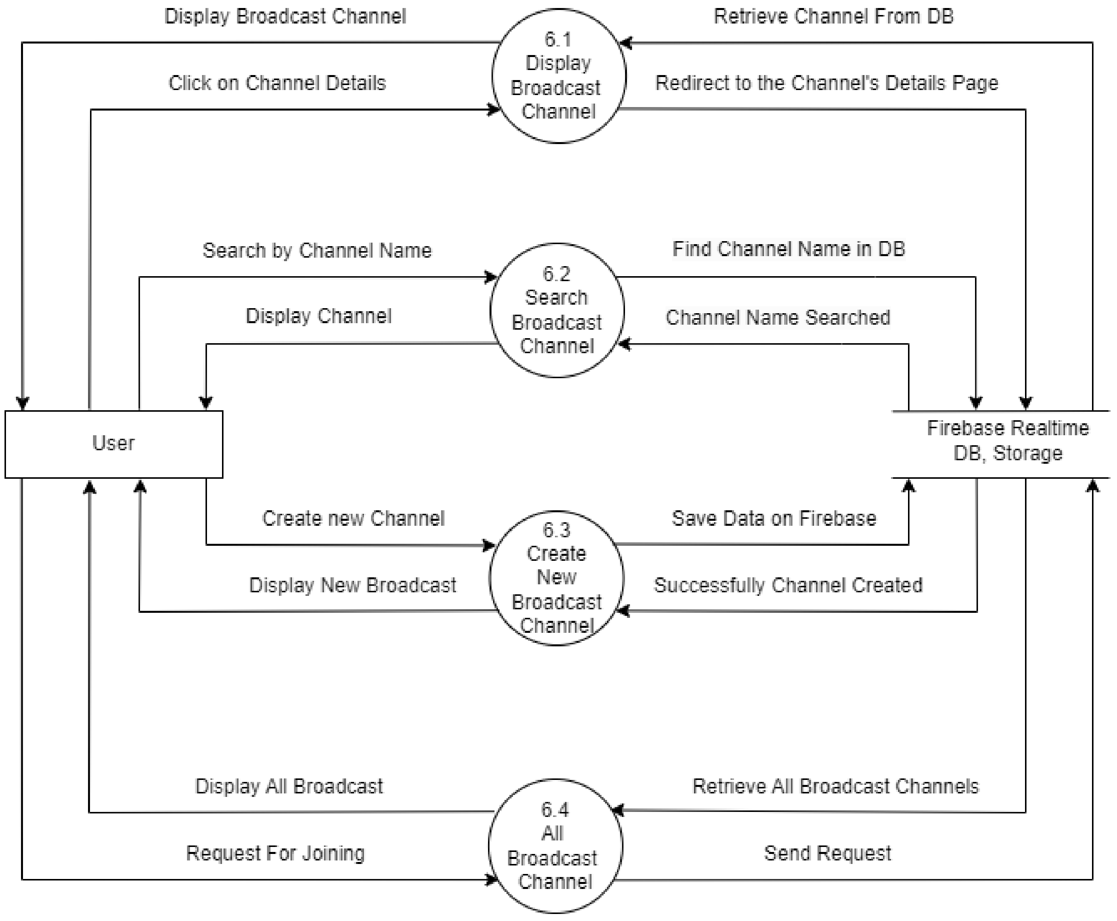
- **2nd Level DFD for Group's Details Process :-**



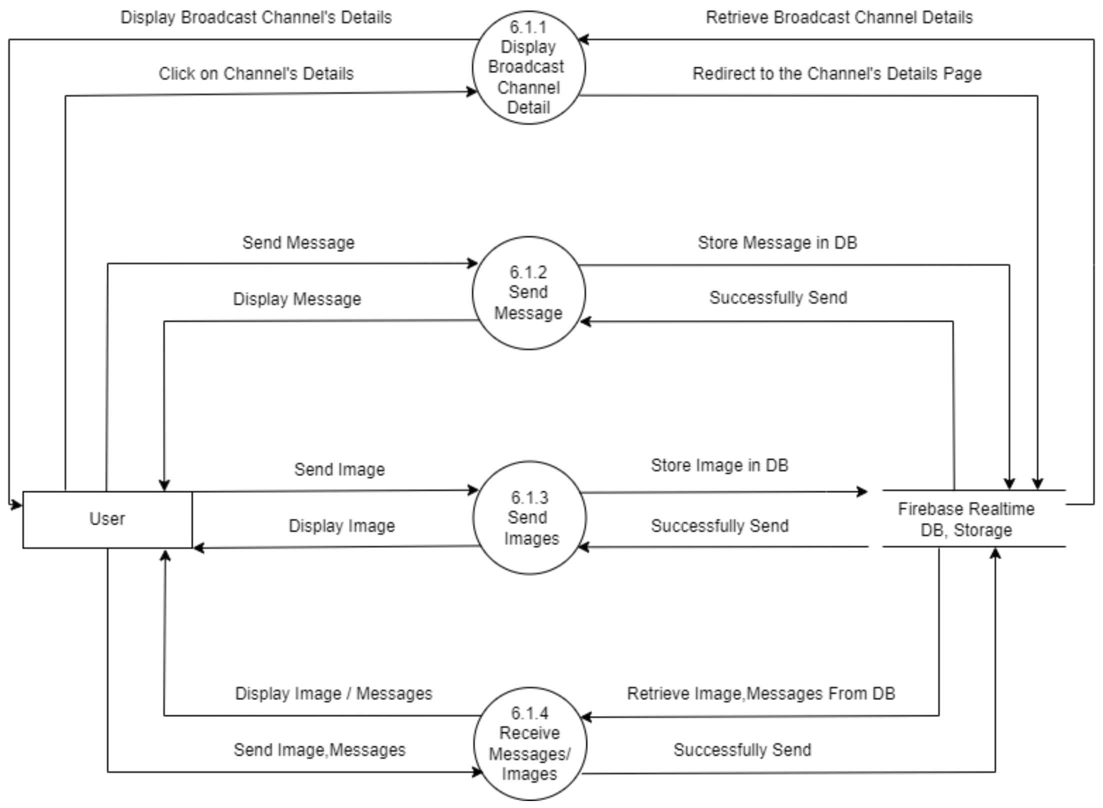
- **2nd Level DFD for Dialing Process :-**



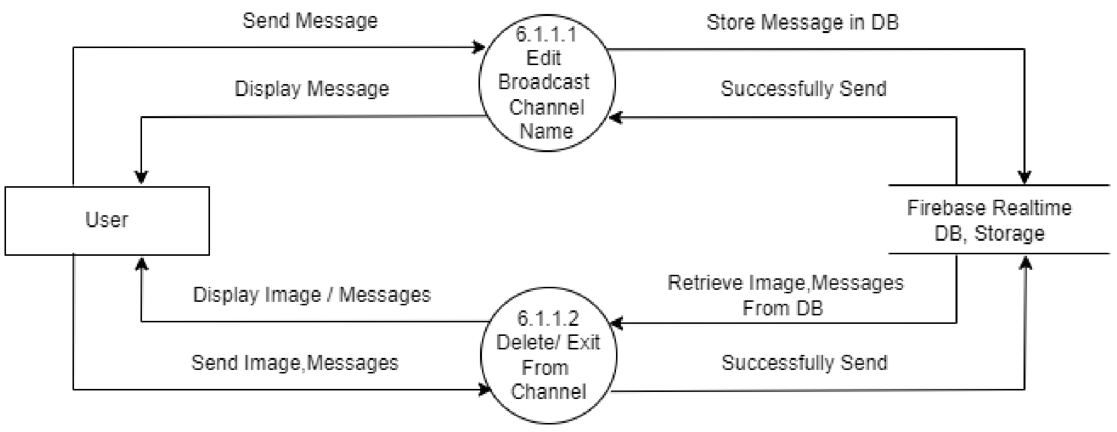
- **2nd Level DFD for Broadcast Fragment's Process :-**


Display Broadcast Channel
Retrieve Channel From DB
6.1 Display Broadcast Channel
Click on Channel Details
Redirect to the Channel's Details Page
Search by Channel Name
Find Channel Name in DB
6.2 Search Broadcast Channel
Display Channel
Channel Name Searched
User
Firebase Realtime DB, Storage
Create new Channel
Save Data on Firebase
6.3 Create New Broadcast Channel
Display New Broadcast
Successfully Channel Created
Display All Broadcast
Retrieve All Broadcast Channels
6.4 All Broadcast Channel
Request For Joining
Send Request

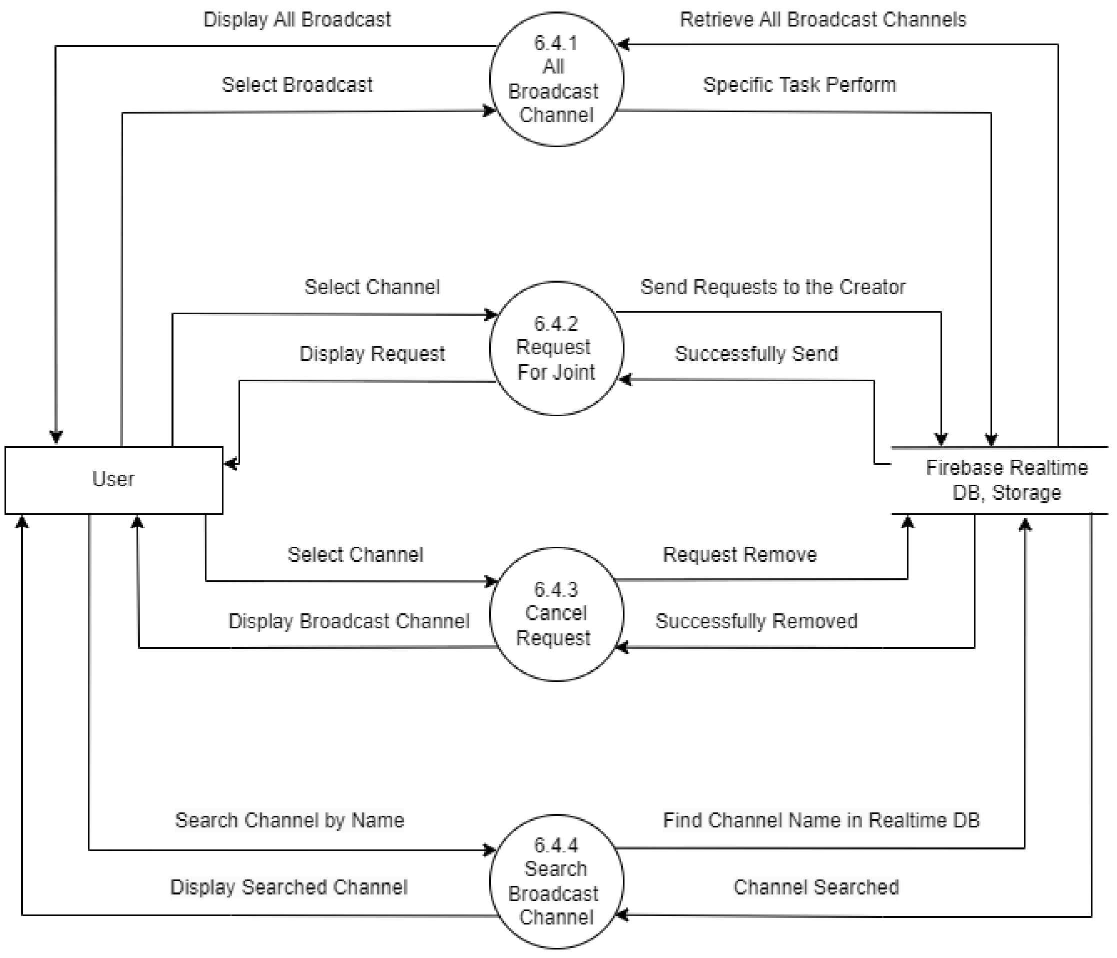
- **2nd Level DFD for Broadcast Channel's Details Process :-**


Display Broadcast Channel's Details
6.1.1 Display Broadcast Channel Detail
Retrieve Broadcast Channel Details
Click on Channel's Details
Redirect to the Channel's Details Page
Send Message
6.1.2 Send Message
Store Message in DB
Display Message
Successfully Send
Send Image
6.1.3 Send Images
Store Image in DB
User
Display Image
Successfully Send
Firebase Realtime DB, Storage
Display Image / Messages
6.1.4 Receive Messages/ Images
Retrieve Image,Messages From DB
Send Image,Messages
Successfully Send

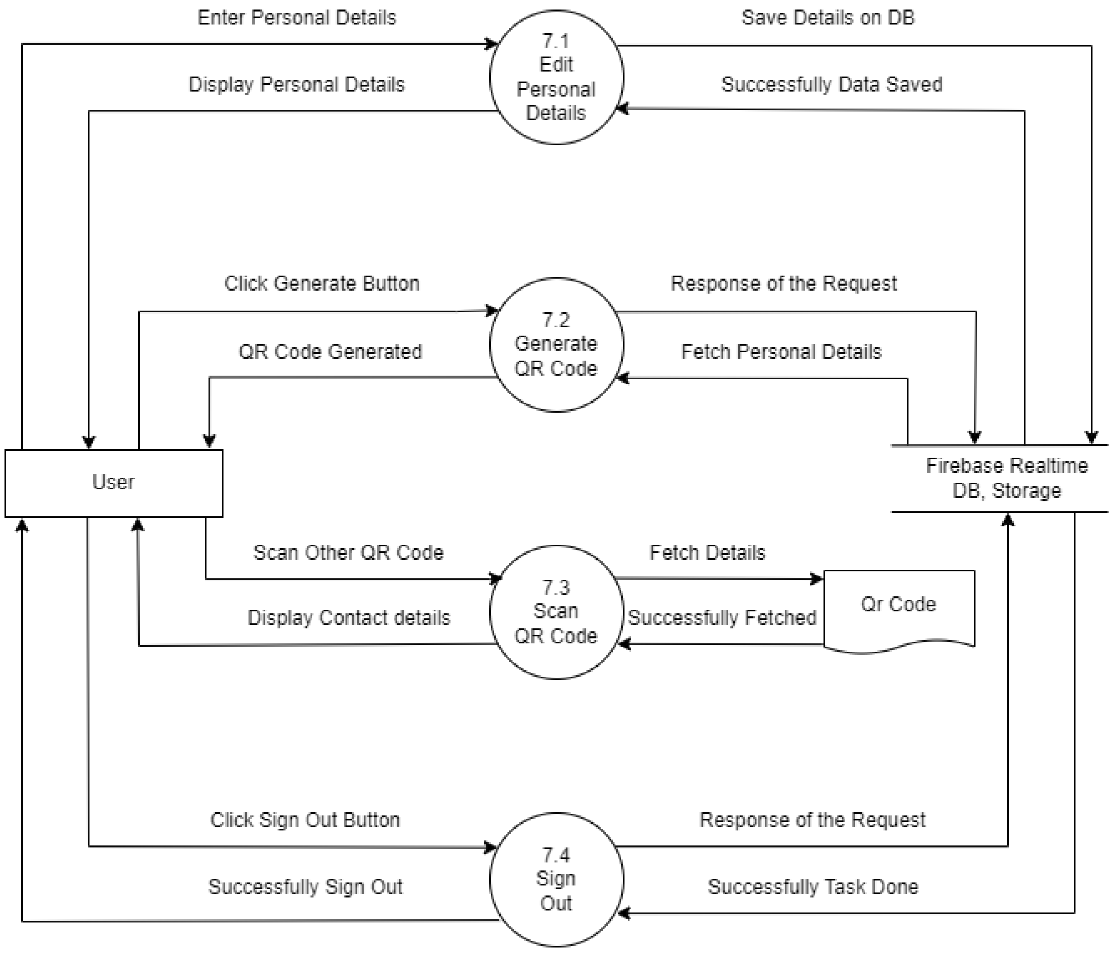
2nd Level DFD for Broadcast Channel's Edit Process :-



- ## 2nd Level DFD for All Broadcast Activity's Process :-



- **2nd Level DFD for Account Fragment's Process :-**



4. Software Project Planning

4.1 Scope :-

1. Contact Management and Interaction:

- Users can efficiently manage their contacts and engage in communication through various channels, including chatting, group messaging, and image sharing.
- The app allows users to create, edit, delete, and search for contacts, ensuring streamlined contact management.

2. Real-time Communication:

- Users can communicate in real-time through high-quality video and audio calls facilitated by ZEGOCLOUD integration.
- Firebase Cloud Messaging enables real-time push notifications for messages, images, and calls, keeping users informed and engaged.

3. Offline Communication:

- The application supports offline communication through traditional mobile networks, including SMS messages and emails.
- Users can continue to interact and communicate even in low-connectivity environments, ensuring uninterrupted communication.

4. Broadcast Channels:

- Users can create broadcast channels to share messages and images with a wider audience.
- Administrators can manage and curate these channels, ensuring a personalized and secure interaction space for users.

5. QR Code Integration:

- QR code generation and scanning functionalities facilitate efficient data exchange and contact management.
- Users can generate QR codes containing contact information and scan QR codes to add new contacts or access relevant content seamlessly.

6. Enhanced User Experience:

- The application boasts an intuitive user interface and experience, ensuring smooth navigation and enhanced user engagement.
- XML-based UI design and Java-powered backend logic contribute to a visually appealing and user-friendly application.

7. Backend Infrastructure:

- Firebase serves as the backbone of the application's backend infrastructure, providing real-time data synchronization, secure authentication, and insightful analytics.
- Firebase Realtime Database and Firestore enable seamless collaboration between users, while Firebase Authentication ensures user data security and privacy.

8. Cross-platform Compatibility:

- ZEGOCLOUD's cross-platform compatibility ensures broad accessibility across devices and operating systems, fostering inclusivity and connectivity among users.

4.2 Objectives:-

1. Facilitate Seamless Communication: Our primary objective is to provide users with a platform that facilitates seamless communication with their contacts. Through features such as chatting, group messaging, and high-quality video/audio calls, we aim to create an environment where users can effortlessly connect and interact with others, fostering meaningful relationships and productive conversations.

2. Streamline Contact Management: We are committed to simplifying the process of contact management for our users. By offering tools to create, edit, delete, and search for contacts with ease, we aim to enhance organization and accessibility, empowering users to stay on top of their contacts and communication networks effortlessly.

3. Enhance User Experience: Our utmost priority is to deliver an exceptional user experience that delights and engages our users. Through intuitive and visually appealing UI/UX design, we strive to create an environment that not only looks great but also feels intuitive and seamless to navigate, ensuring that users enjoy every interaction with our application.

4. Enable Real-time Interaction: Real-time communication is essential in today's fast-paced digital world, and our goal is to enable users to stay connected and informed in real-time. By implementing push notifications for messages, images, and calls, we ensure that users are always in the loop, even when they are actively engaged in conversations or tasks.

5. Promote Offline Connectivity: We recognize that connectivity is not always guaranteed, especially in low-connectivity environments. Therefore, we aim to provide robust support for offline communication through SMS messages and emails, ensuring that users can stay connected and continue their conversations regardless of their network status.

6. Empower Broadcasting: Our application empowers users to share their messages and images with a wider audience through broadcast channels. By allowing users to create and manage these channels, we provide a personalized and secure space for interaction, fostering community engagement and collaboration.

7. Simplify Data Exchange: We understand the importance of simplifying data exchange and contact management for our users. Through QR code integration, we enable users to effortlessly add new contacts or access relevant content by simply scanning QR codes, eliminating the need for manual data entry and streamlining the user experience.

8. Ensure Backend Reliability: A reliable backend infrastructure is crucial for the smooth operation of our application. By leveraging Firebase as the backbone of our backend infrastructure, we ensure real-time data synchronization, secure authentication, and insightful analytics, providing a solid foundation for our users to rely on.

9. Promote Cross-platform Compatibility: Our application is designed to be accessible across devices and operating systems. By leveraging ZEGOCLOUD's cross-platform compatibility, we ensure that users can access our application seamlessly from any device, fostering inclusivity and connectivity among our user base.

4.3 Expected Advantages :-

1. Efficient Contact Management: Users can easily manage their contacts with features like swipe actions (e.g., edit and delete), custom dial keypad, and the ability to block/unblock contacts.

2. Seamless Communication: The app offers various communication channels including online calling through ZEGOCLOUD, SMS, email, and normal calling features, ensuring users can connect with their contacts in multiple ways.

3. Enhanced Messaging: Users can enjoy a rich messaging experience with features like chat functionality, image sharing, and the ability to see unread message counts and message lists in chats and groups.

4. Group Communication: The app allows users to create and manage groups, share messages and images, add or remove group members, and update group settings, enhancing collaboration among group members.

5. Broadcast Channels: Users can create broadcast channels to share messages and images with a wider audience. They can also see all broadcast channels, request to join, and receive notifications for joining requests.

6. QR Code Functionality: Users can create their QR codes containing personal details and share them. Other users can scan these codes to get contact details and interact with them, simplifying the process of sharing and saving contact information.

7. Real-time Notifications: The app sends real-time notifications for incoming messages, images, and calls, ensuring users stay informed and engaged with their contacts.

8. Cross-device Synchronization: Users can sign out from the application and log in on another device to access their data, ensuring data continuity and accessibility across devices.

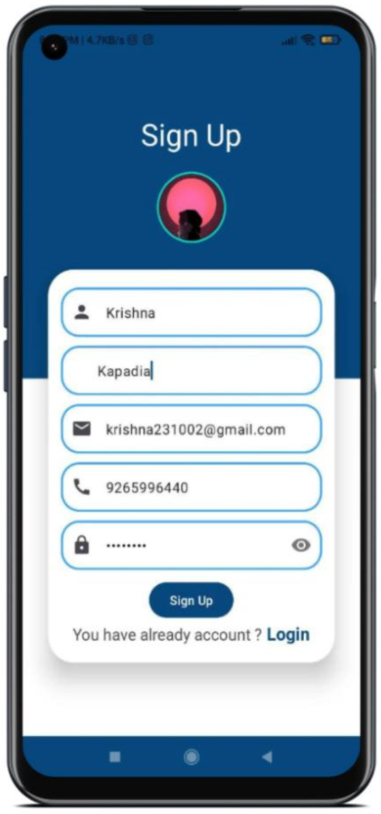
9. Testing Capabilities: The app includes testing capabilities for group messages, group chats, user profiles, and broadcast messages, ensuring that information is stored accurately and can be retrieved smoothly.

10. User-Friendly Interface: The app features a custom search view in the app bar, allowing users to easily search for contacts and messages. It also displays pending unread messages counts and organizes messages in chat and group lists for easy access.

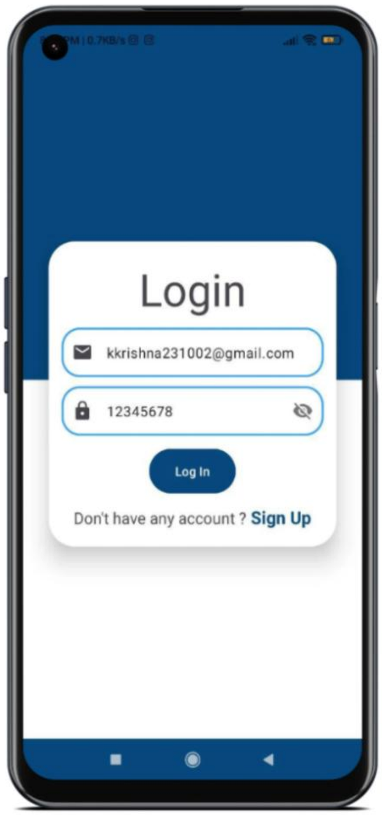
5. System Design

5.1 Input Design :-

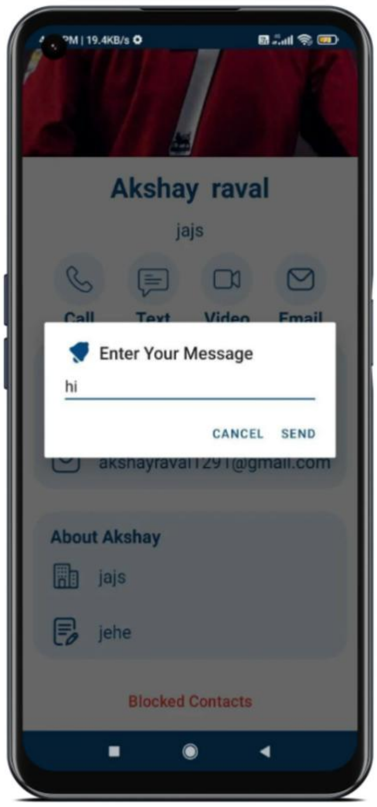
Sign Up



Login



Send Text Message



Akshay raval

jajs

Enter Your Message

hi

CANCEL SEND

akshayraval1291@gmail.com

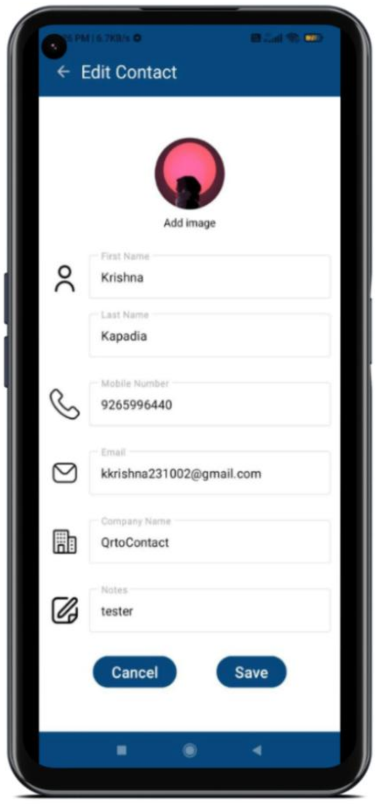
About Akshay

jajs

jehe

Blocked Contacts

Edit Contacts



Edit Contact

Add image

First Name: Krishna

Last Name: Kapadia

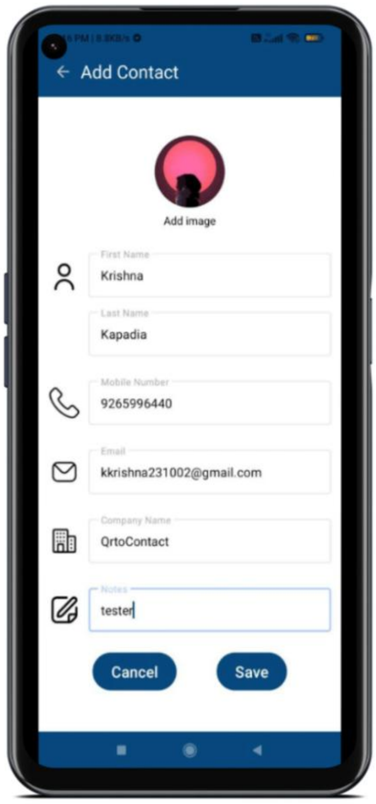
Mobile Number: 9265996440

Email: kkrishna231002@gmail.com

Company Name: QrtoContact

Notes: tester

Add Contacts



Add Contact

Add image

First Name: Krishna

Last Name: Kapadia

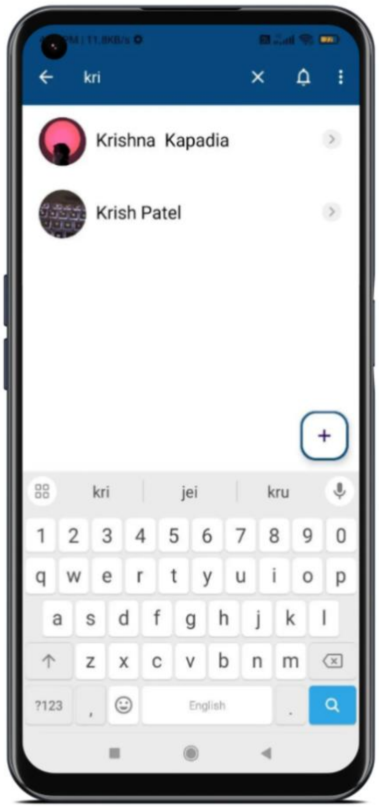
Mobile Number: 9265996440

Email: kkrishna231002@gmail.com

Company Name: QrtoContact

Notes: tester

Search Contacts

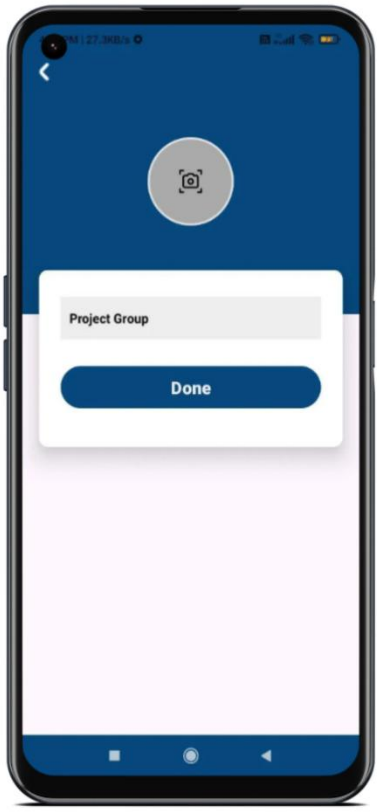


kri

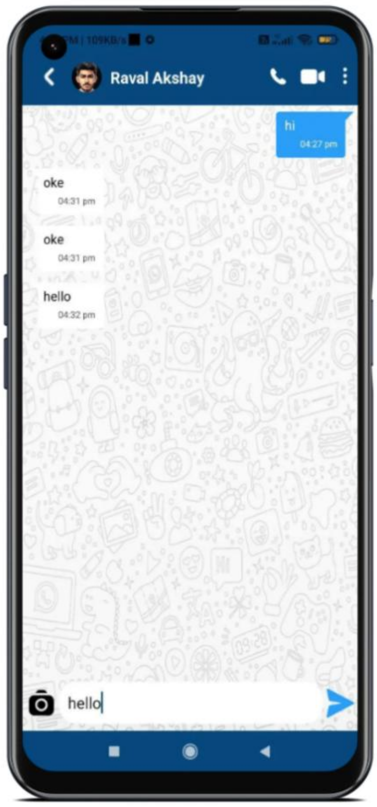
Krishna Kapadia

Krish Patel

Create Group



Enter Chat Message



hi
04:27 pm

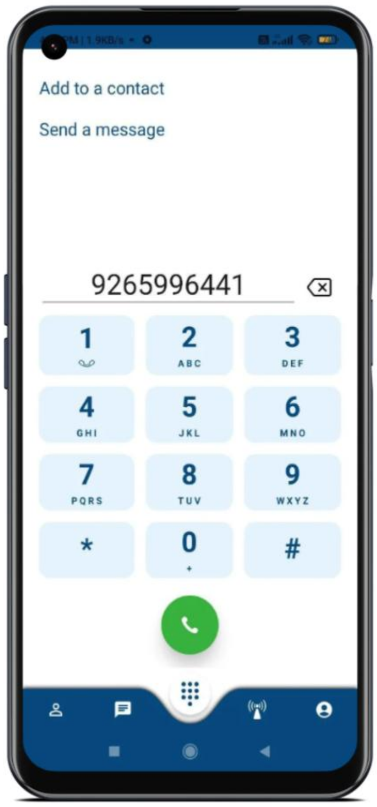
oke
04:31 pm

oke
04:31 pm

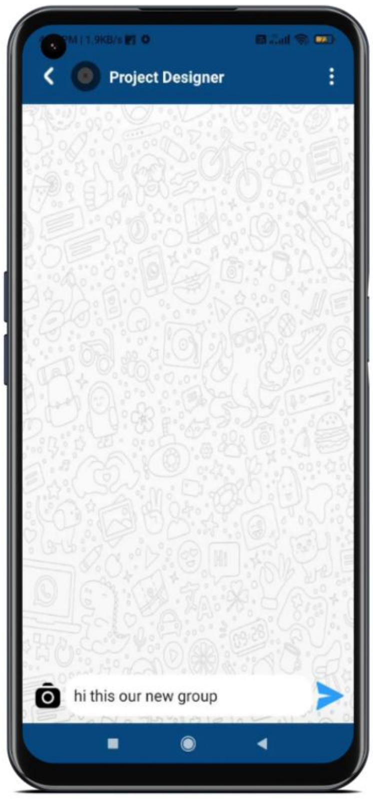
hello
04:32 pm

hello

Dial Contacts

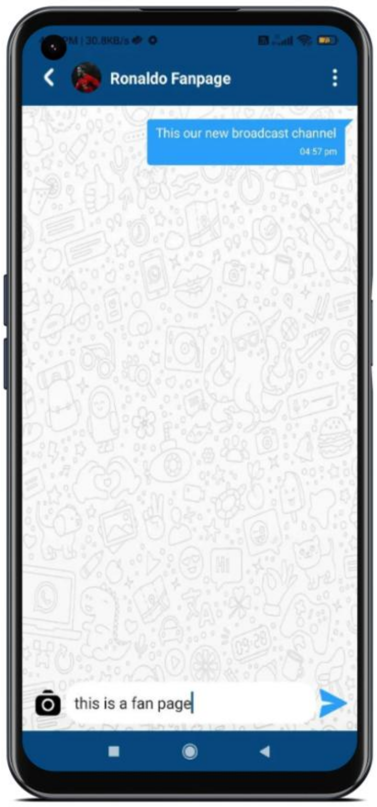


Enter Group Message

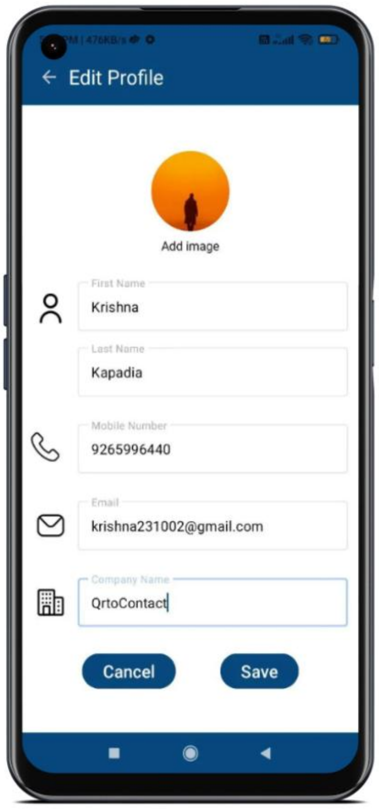


hi this our new group

Enter Broadcast Message



Edit Profile



First Name: Krishna

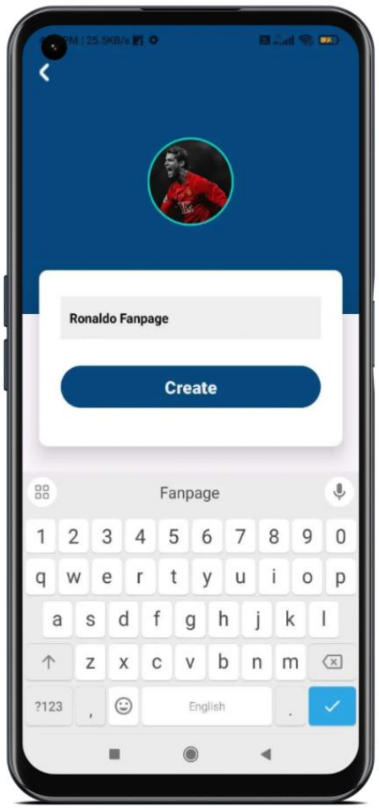
Last Name: Kapadia

Mobile Number: 9265996440

Email: krishna231002@gmail.com

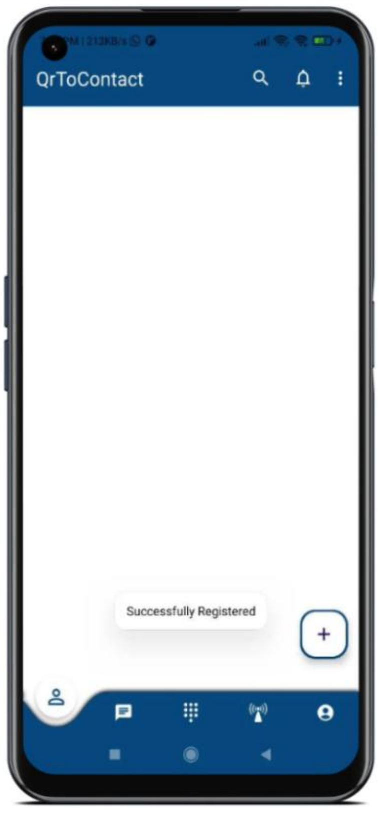
Company Name: QrtoContact

Create Broadcast Channel

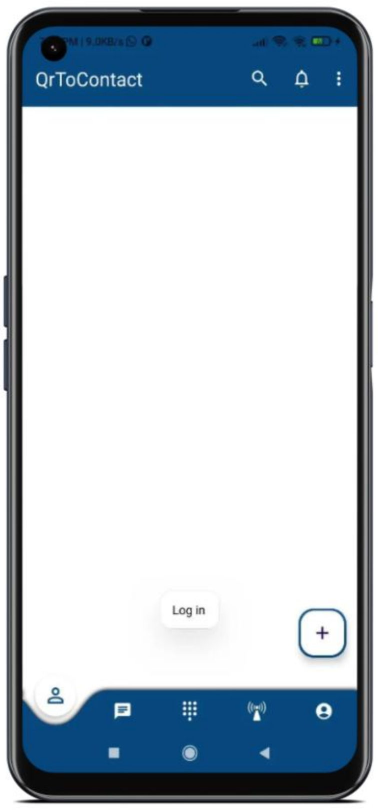


5.2 Output Design :-

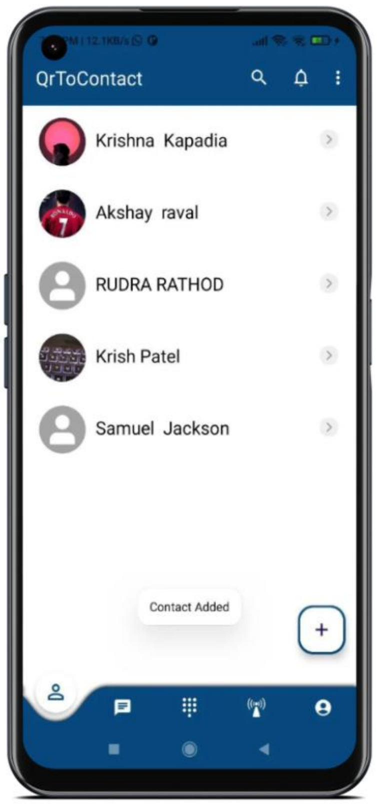
Successfully Sing Up



Successfully Login



Contact Added


QrToContact

Krishna Kapadia

Akshay raval

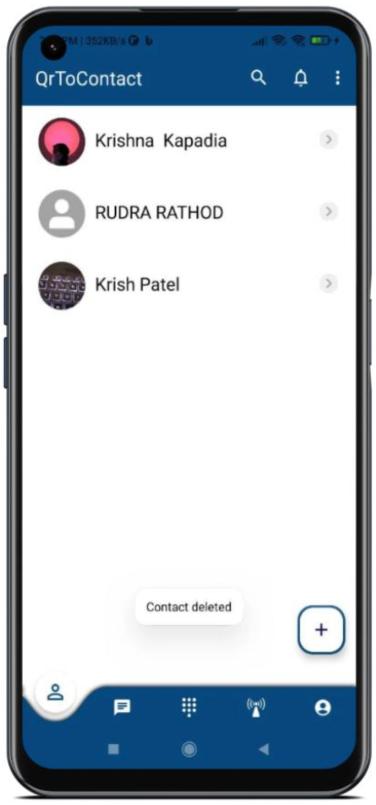
RUDRA RATHOD

Krish Patel

Samuel Jackson

Contact Added

Contact Deleted


QrToContact

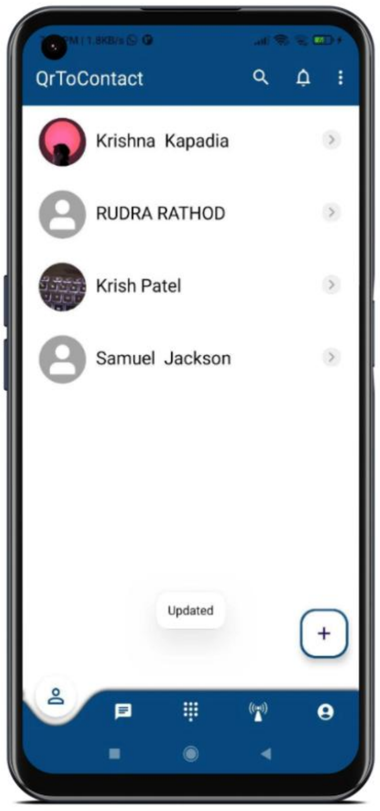
Krishna Kapadia

RUDRA RATHOD

Krish Patel

Contact deleted

Contact Updated


QrToContact

Krishna Kapadia

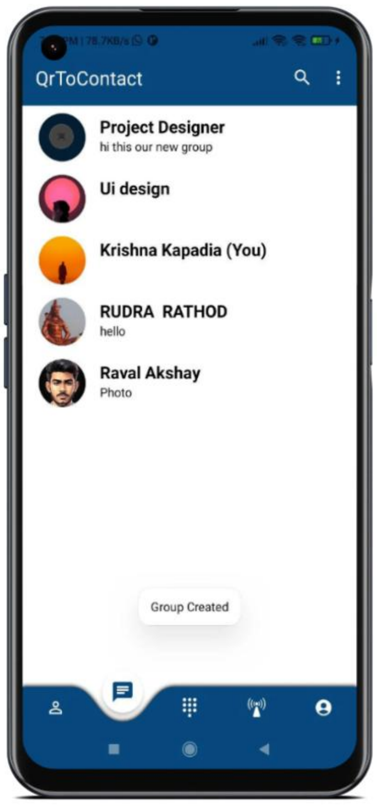
RUDRA RATHOD

Krish Patel

Samuel Jackson

Updated

Group Created


QrToContact

Project Designer
hi this our new group

Ui design

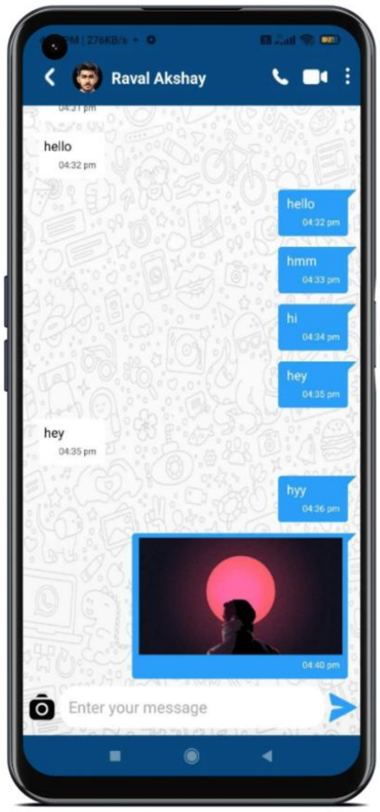
Krishna Kapadia (You)

RUDRA RATHOD
hello

Raval Akshay
Photo

Group Created

Message Send in Chat


Raval Akshay

04:31 pm

hello
04:32 pm

hello
04:32 pm

hmm
04:33 pm

hi
04:34 pm

hey
04:35 pm

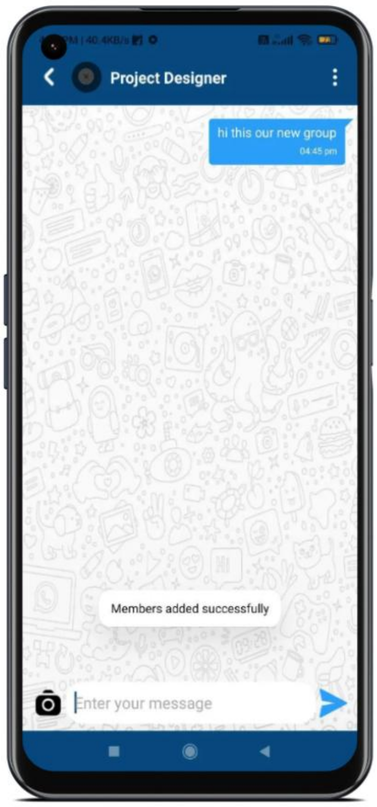
hey
04:35 pm

hyy
04:36 pm

04:40 pm

Enter your message

Member Added

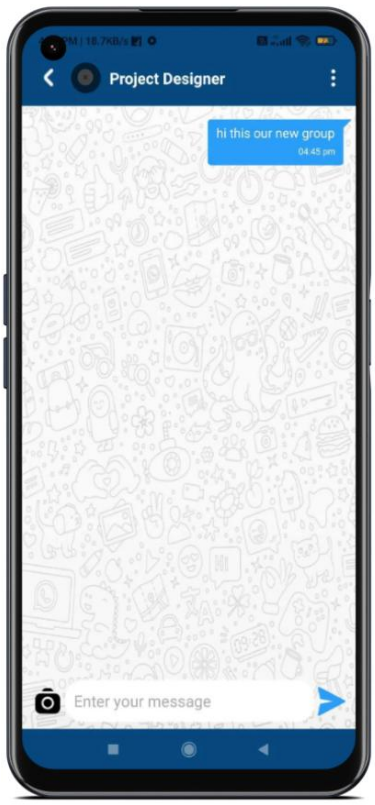

Project Designer

hi this our new group
04:45 pm

Members added successfully

Enter your message

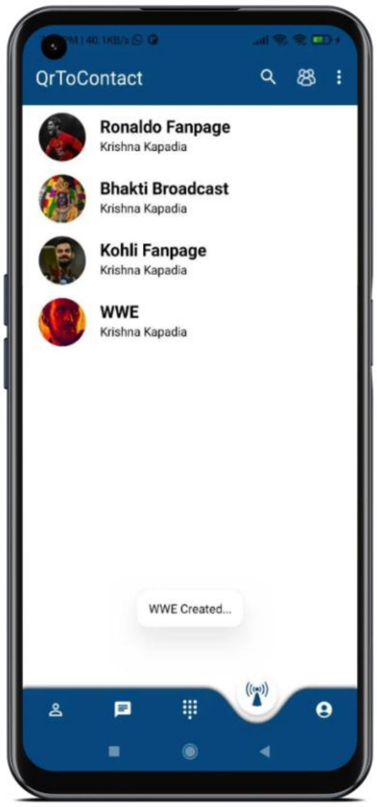
Message Send in Group


Project Designer

hi this our new group
04:45 pm

Enter your message

Broadcast Channel Created


QrToContact

Ronaldo Fanpage
Krishna Kapadia

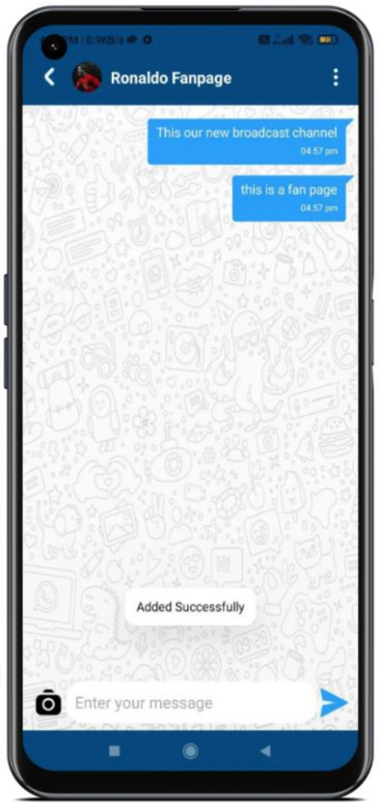
Bhakti Broadcast
Krishna Kapadia

Kohli Fanpage
Krishna Kapadia

WWE
Krishna Kapadia

WWE Created...

Member Add in Channel



Ronaldo Fanpage

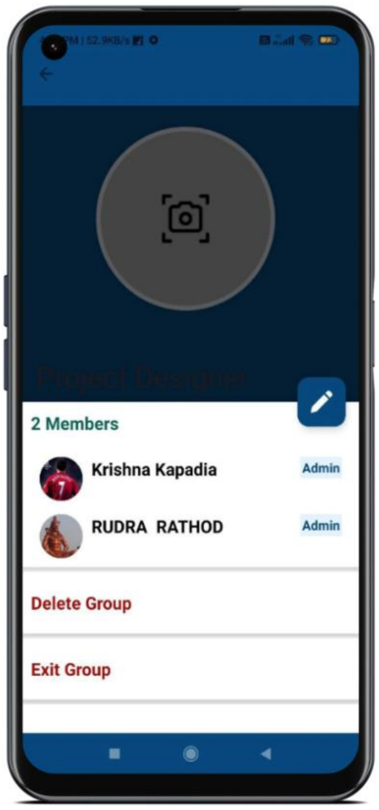
This our new broadcast channel
04:57 pm

this is a fan page
04:57 pm

Added Successfully

Enter your message

Make Member Admin



2 Members

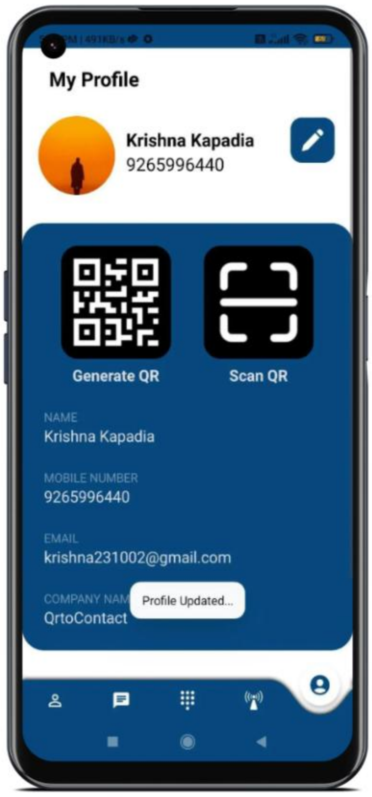
Krishna Kapadia Admin

RUDRA RATHOD Admin

Delete Group

Exit Group

Account Details



My Profile

Krishna Kapadia
9265996440

NAME
Krishna Kapadia

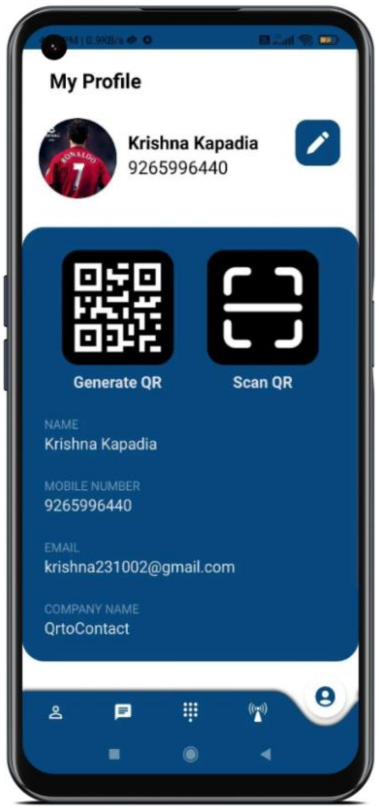
MOBILE NUMBER
9265996440

EMAIL
krishna231002@gmail.com

COMPANY NAM
QrtoContact

Profile Updated...

Details Updated



My Profile

Krishna Kapadia
9265996440

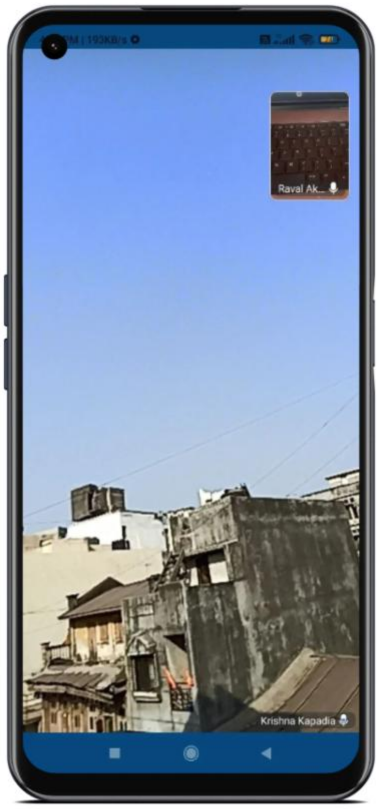
NAME
Krishna Kapadia

MOBILE NUMBER
9265996440

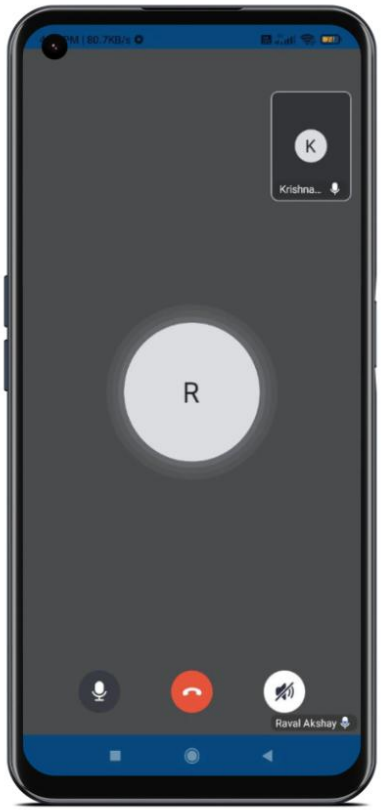
EMAIL
krishna231002@gmail.com

COMPANY NAME
QrtoContact

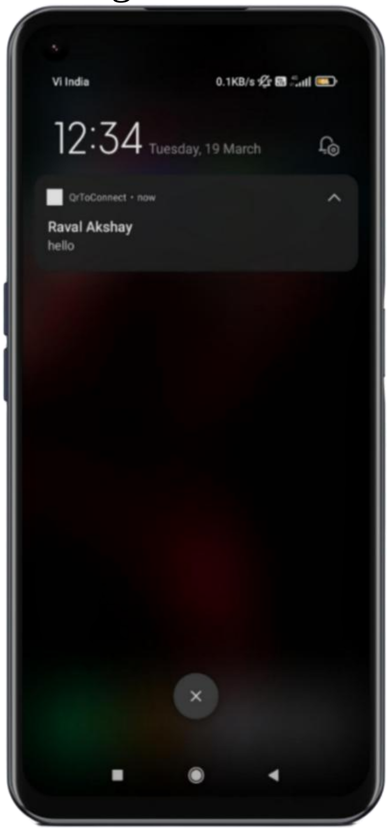
Video Call



Internet Audio Call

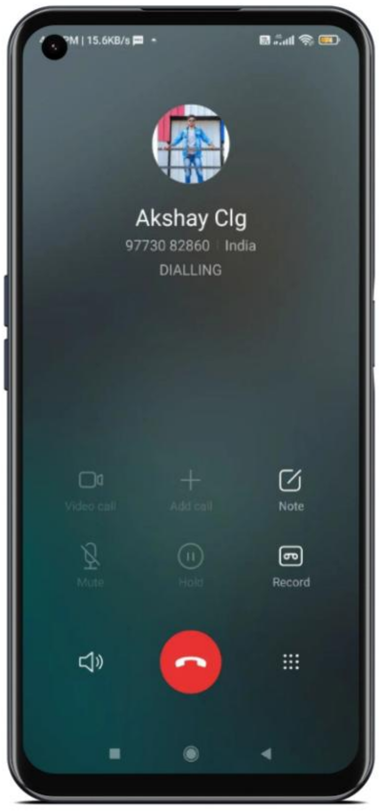


Message Notification

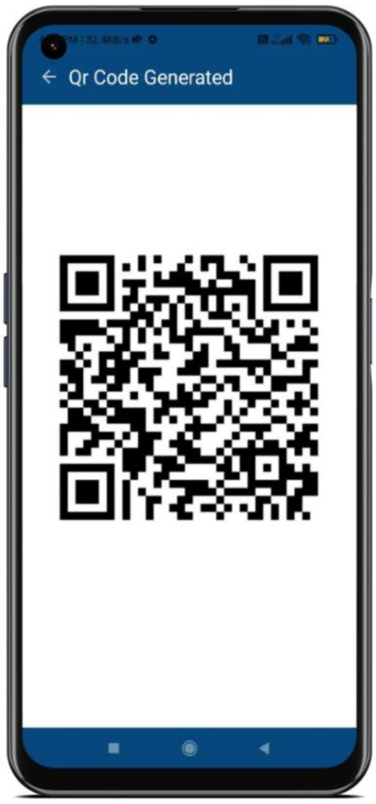


Raval Akshay
hello

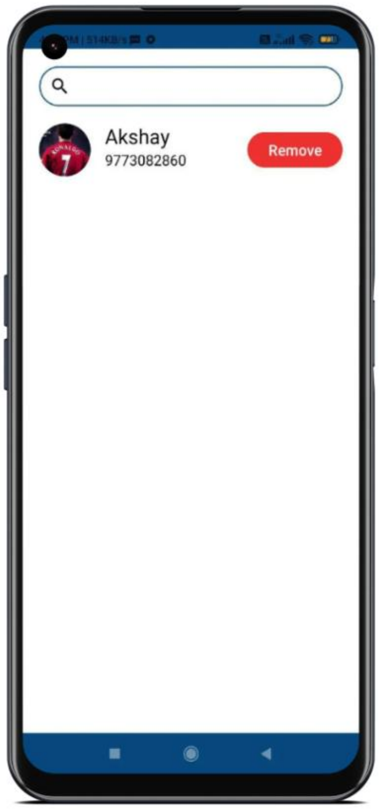
Normal Call



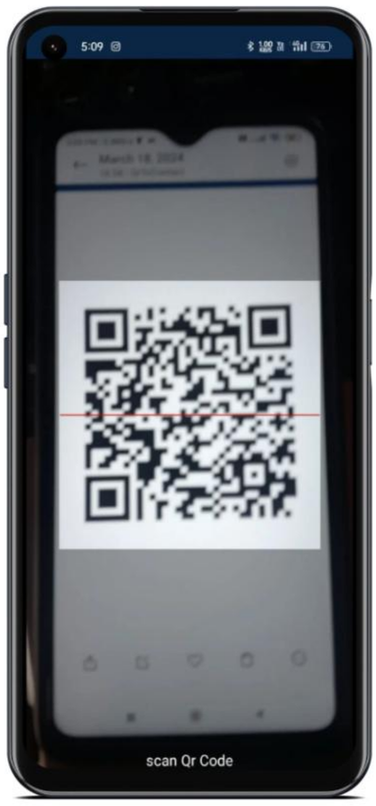
Qr Code Generated



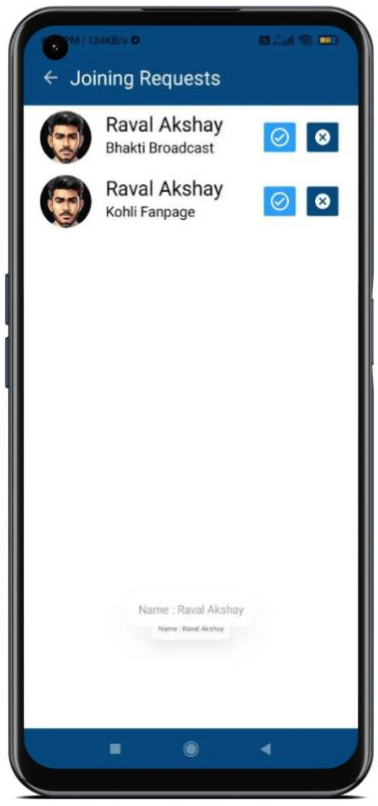
Blocked Contact



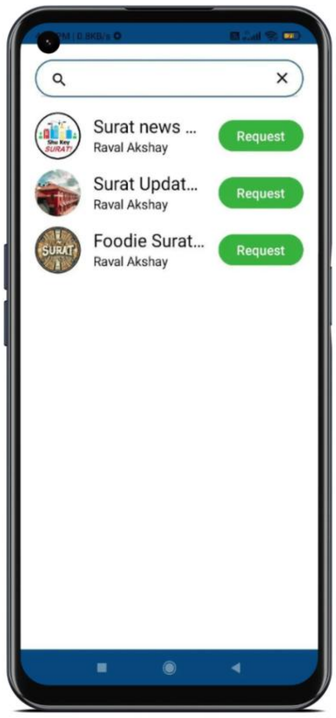
Qr Code Scanner



Request For Joining



Send Request



6. Software Testing

6.1 Introduction:

Testing is an integral part of software development, playing a crucial role in detecting errors and ensuring the reliability and quality of software products. It is a systematic process of evaluating a software application or system to identify discrepancies between expected and actual results. The results obtained from testing are not only used to validate the correctness of the software but also play a significant role in maintenance activities, helping developers identify and rectify issues in subsequent iterations.

6.2 Objectives of Testing:

1. Detecting Errors: The primary objective of testing is to uncover errors or defects in the software. By systematically executing test cases, developers aim to identify discrepancies between expected and actual outcomes, ensuring that all aspects of the software function as intended.

2. Ensuring Reliability: Testing is essential for ensuring the reliability of software systems. By verifying that the software behaves consistently under various conditions, developers can instill confidence in users that the application will perform reliably in real-world scenarios.

3. Validating Requirements: Testing serves to validate that the software meets the specified requirements and performs the intended functions. By comparing the actual behavior of the software against predefined criteria, developers can ensure alignment with user expectations and project goals.

4. Improving Quality: Through rigorous testing, developers aim to improve the overall quality of the software. By identifying and rectifying errors early in the development process, testing helps minimize defects and vulnerabilities, leading to a more robust and stable product.

5. Enhancing User Satisfaction: Testing contributes to enhancing user satisfaction by ensuring that the software meets or exceeds user expectations. By identifying and addressing issues related to usability, functionality, and performance, developers can deliver a product that provides a positive user experience.

6. Facilitating Maintenance: The results of testing are invaluable for maintenance activities, providing developers with insights into potential areas for improvement and optimization. By analyzing test outcomes and identifying recurring issues, developers can make informed decisions to enhance the software's quality and maintainability over time.

7. Mitigating Risks: Testing helps mitigate risks associated with software development by identifying and addressing potential issues early in the process. By proactively identifying defects and vulnerabilities, developers can minimize the likelihood of costly errors and failures in production environments.

8. Ensuring Compliance: Testing ensures that the software complies with relevant standards, regulations, and industry best practices. By conducting thorough testing, developers can verify that the software meets legal and regulatory requirements, as well as industry-specific standards and guidelines.

6.3 Levels of Testings:

1. Unit Testing:

- Focuses on verifying the smallest unit of software, typically a module or function.
- Tests specific paths in the module's control structure to ensure complete coverage.
- Aims to detect errors within individual units of code and ensure they function correctly in isolation.

2. Integration Testing:

- Systematic technique for constructing the program's structure while testing interfaces between components.
- Conducts tests to uncover errors associated with interfacing between different modules or components.
- Ensures that integrated components work together as expected and interface correctly.

3. Validation Testing:

- Occurs after Integration Testing when the software is completely assembled.
- Uses a series of Black Box tests to demonstrate conformity with the requirements.
- Focuses on ensuring that the software meets specified requirements and functions correctly in its intended environment.

4. System Testing:

- Series of tests aimed at fully exercising the computer-based system.
- Includes various types of tests such as recovery testing, security testing, stress testing, and performance testing.
- Verifies that all system elements have been properly integrated and perform allocated functions.

5. Performance Testing:

- Evaluates the run-time performance of the application within the context of an integrated system.
- Tests the proper response time for user actions, which is critical for maintaining and enhancing user satisfaction.
- Ensures that the application meets performance requirements under normal operating conditions.

6. Load Testing:

- Demonstrates how the application performs under concurrent user sessions for typical user scenarios.
- Sets up common scenarios executed for a short period to observe how the application operates under multiple-user loads.
- Helps identify performance bottlenecks and ensure the application can handle expected user loads.

7. Stress Testing:

- Examines how the application behaves under maximum user load.
- Removes think time for load scripts and executes them against the server to overload the application.
- Helps identify robustness issues and potential failure points under extreme user activity conditions.

6.4 Test Cases :

1. Login:

These test cases validate the login functionality of the application. They ensure that entering invalid email or password prompts errors, while providing valid credentials results in successful login. Additionally, they check for proper handling when required information is not filled.

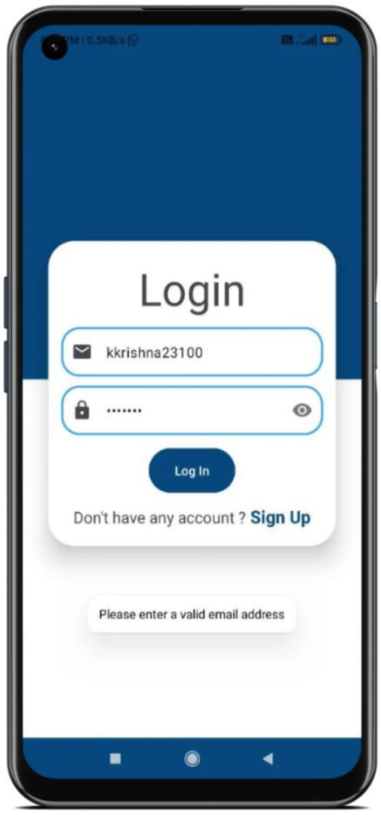
Sr. no.	**Measure**	**Expected Result**	**Actual Result**	**Remark**
1	Enter invalid Email	Error	Enter a valid Email	Error
2	Enter invalid Password	Error	Enter a valid Password	Error
3	If not fill any information	Error	Enter proper details	Error
4	Enter valid Email, and Password	Log in	Log in	Error

2. Registration:

This test case evaluates the registration process of an application. It checks for handling of various scenarios such as email already registered, invalid email format, invalid password, missing information, and successful registration with valid details. Each step is verified against the expected and actual results for accuracy.

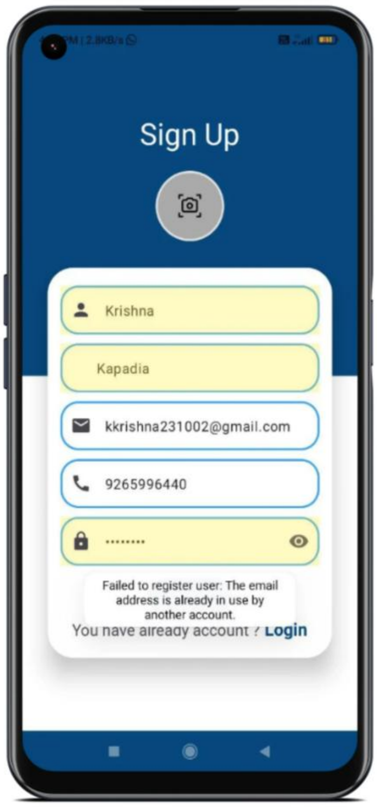
Sr. no.	Measure	Expected Result	Actual Result	Remark
1	Enter registered Email	Error	Email already registered	Error
2	Enter invalid Email	Error	Enter a valid email	Error
3	Enter invalid Password	Error	Enter a valid password	Error
4	If not fill any information	Error	Required	Error
5	Enter valid Email, First name, Last name, Email, Mobile,Password	Successfully Registered	Successfully Registered	Successfully Registered

Login Test



Please enter a valid email address

Registration Test



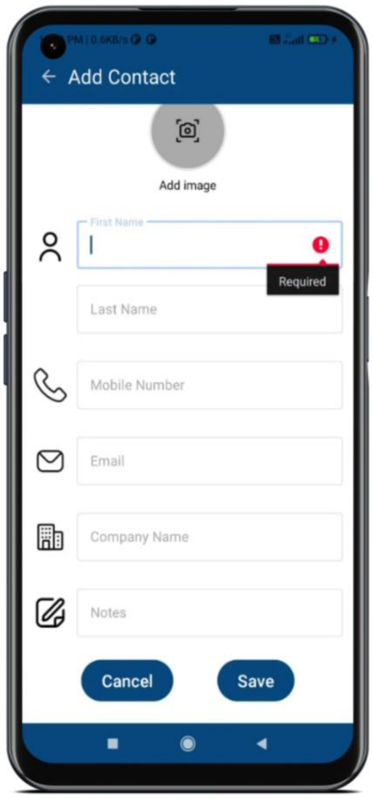
Failed to register user: The email address is already in use by another account.

3. Add Contacts :

This test case evaluates the contact addition process in an application. It verifies three scenarios: validating email input, ensuring required fields are filled (Firstname and Mobile), and successfully adding a contact with both Firstname and Mobile provided. The test ensures accurate error handling and contact addition functionality.

Sr. no.	Measure	Expected Result	Actual Result	Remark
1	Enter invalid email	Error	Enter a valid email	Error
2	If Firstname or Mobile is not fill	Error	Required	Error
3	Enter Firstname and Mobile	Contact Added	Contact Added	Contact Added

Add Contact Test

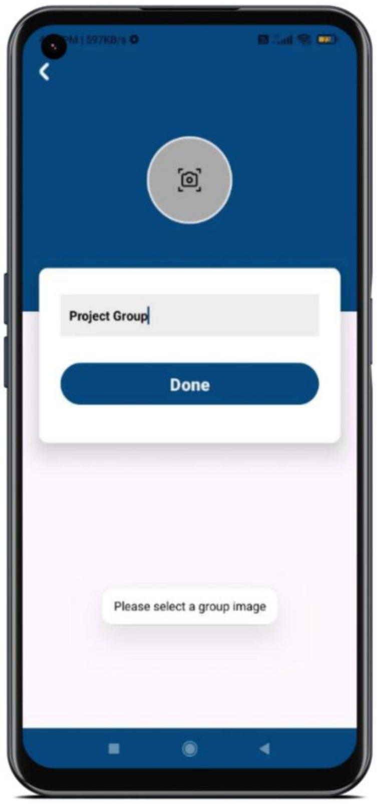


4. Create Group :

This test case verifies the functionality of creating a group in an application. It checks if an error is prompted when an image is not selected or the group name is left blank. Additionally, it ensures that a group is successfully created when both an image and a name are provided.

Sr. no.	Measure	Expected Result	Actual Result	Remark
1	If image is not picked	Error	Select a group Image	Error
2	If group name is not fill	Error	Required	Error
3	Enter Group Image and name	Group Created	Group Created	Group Created

Create Group Test



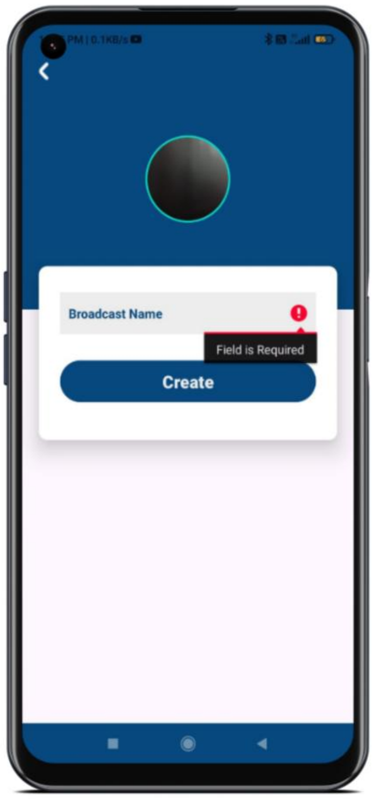
Please select a group image

5. Create Broadcast :

This test case verifies the functionality of broadcasting in an application. It checks if an image is selected and if the broadcast name is filled out. It ensures that both conditions are met for successful broadcasting, highlighting error prompts when criteria aren't fulfilled and confirming successful creation when they are.

Sr. no.	Measure	Expected Result	Actual Result	Remark
1	If image is not picked	Error	Select a Broadcast Image	Error
2	If broadcast name is not fill	Error	Required	Error
3	Enter Broadcast Image and name	Created	Created	Created

Broadcast Channel Creating Test

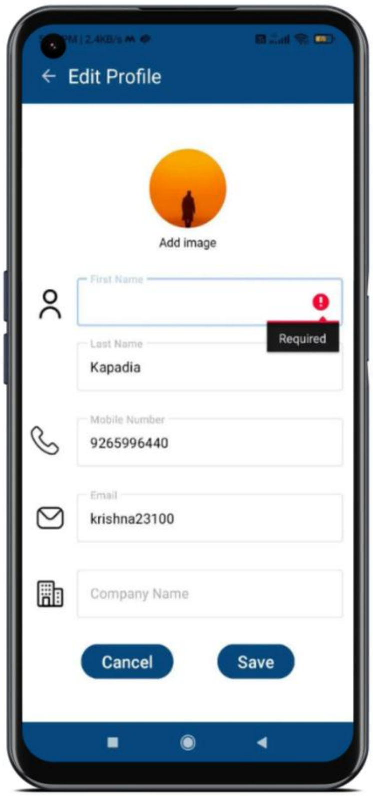


6. Edit Profile Details :

This test case verifies essential functionalities of a profile editing feature. It checks for error handling when no image is selected, prompts for a valid email if entered incorrectly, ensures all fields are filled, and confirms successful profile updates with correct image, first name, last name, valid email, and mobile number.

Sr. no.	Measure	Expected Result	Actual Result	Remark
1	If image is not picked	Error	Select a Image	Error
2	Enter invalid email	Error	Enter a valid email	Error
3	If any field is not fill	Error	Required	Error
4	Enter Image ,Firstname , lastname, valid Email and Mobile no.	Profile Updated..	Profile Updated..	Profile Updated..

Profile Editing Test

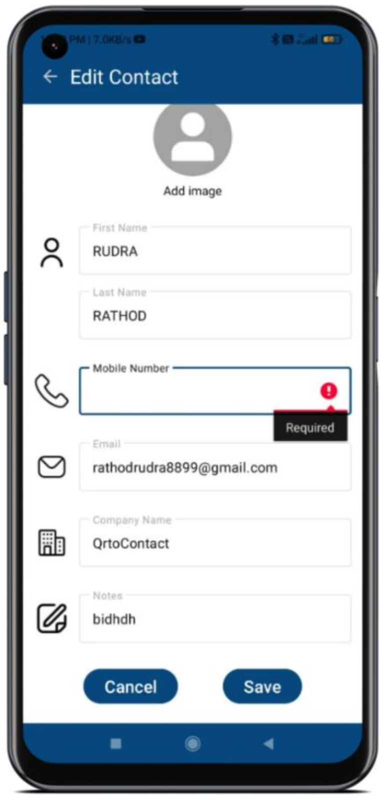


7. Update contact:

This test case validates contact update functionality by checking for errors when entering an invalid email, ensuring both first name and mobile fields are filled, and confirming successful updates with valid inputs. It ensures accurate error handling and data updating, crucial for maintaining data integrity and user experience in the application.

Sr. no.	Measure	Expected Result	Actual Result	Remark
1.	Enter invalid email	Error	Enter a valid email	Error
2.	If Firstname and Mobile is not fill	Error	Required	Error
3.	Enter Firstname and mobile	Updated	Updated	Updated

Updating Contact Details Test


Edit Contact

First Name: RUDRA

Last Name: RATHOD

Mobile Number:

Required

Email: rathodrudra8899@gmail.com

Company Name: QrtoContact

Notes: bidhdh

7. Limitation and System Enhancement

7.1 Limitation :-

1. Dependency on Internet Connection: While the application supports offline communication through SMS and email, many features such as online calling, real-time messaging, and broadcast channels rely on an active internet connection, limiting usability in low-connectivity or offline environments.

2. Compatibility Restrictions: Although efforts have been made to ensure cross-platform compatibility, certain features like ZEGOCLOUD integration for video and audio calls may have limitations on compatibility with specific devices or operating systems, potentially excluding some users from accessing these functionalities.

3. Resource Consumption: Features such as real-time notifications and synchronization with Firebase may consume significant device resources like battery and data, potentially affecting device performance and user experience, especially on older or low-end devices.

4. Privacy Concerns: While Firebase Authentication ensures secure user authentication, there may still be privacy concerns regarding the storage and handling of user data, especially with features like QR code sharing of personal details and real-time messaging.

5. Testing and Maintenance Overhead: The inclusion of testing capabilities necessitates ongoing maintenance and updates to ensure the smooth operation of the application, which may require additional time and resources for development and testing.

6. Limited Offline Functionality: While the application supports offline communication through SMS and email, certain features like real-time messaging and online calling may have limited functionality or be inaccessible without an active internet connection, potentially hindering user engagement in offline scenarios.

7. Platform Dependency: While efforts have been made to ensure compatibility with a wide range of Android devices, certain features or optimizations may be specific to the Android platform, limiting accessibility for users on other platforms such as iOS or web.

8. Security Risks: Despite Firebase Authentication and other security measures, the application may still be vulnerable to security risks such as data breaches, unauthorized access, or malware attacks, necessitating ongoing vigilance and updates to address potential vulnerabilities and threats.

7.2 System Enhancement :-

1. Enhanced UI/UX: Improvements can be made to the user interface and experience to ensure it is visually appealing, intuitive to navigate, and engaging for users.

2. Document Sharing: Addition of document sharing functionality in chat, group, and broadcast channels, similar to the existing features for sharing messages and images, to facilitate richer communication.

3. Personal Broadcasting: Implementation of personal broadcasting features, allowing users to broadcast messages or images directly to selected contacts, enhancing personalized communication.

4. Pin Facility: Introduction of a "pin" feature to prioritize important chats, broadcasts, and broadcast channels by placing them at the top of their respective lists for easier access.

5. Improved Video and Audio Calling: Enhancements to video and audio calling functionality, including fixing button icon changes and adding group video and audio calling features to enable seamless communication.

6. Enhanced Search Functionality: Addition of a more robust search functionality where needed, allowing users to quickly find contacts, messages, or content within the application.

7. Communication Security: Implementation of encryption and end-to-end communication security measures to enhance the security and privacy of user communications within the application.

8. Swipe Actions: Addition of swipe gestures for calling and deleting contacts, enabling users to perform actions like calling by swiping left and deleting contacts by swiping right.

9. Last Message and Unread Message Count Display: Display of the last message and unread message count in group chats, broadcasts, and broadcast channels to provide users with quick insights into the conversation activity.

10. Live Streaming Feature: Introduction of live streaming functionality in broadcasts, similar to Zoom meetings, to facilitate real-time engagement and interaction with a larger audience.

11. Message Replay from Notifications: Implementation of a feature that allows users to directly reply to messages from notifications, enhancing the convenience and efficiency of communication.

8. Bibliography

1. https://youtube.com/playlist?list
2. https://youtu.be/T3y3370UE8w?si=03s0EXVFO4dEzEWZ
3. https://stackoverflow.com/questions/72642545/
4. https://g.co/bard/share/518e71b57118
5. https://www.freepik.com/icon/
6. https://app.diagrams.net/
7. https://youtu.be/q-DnUKbGsgA?si=EseeKRAg-zqhvrIw
8. https://www.flaticon.com/free-animated-icon/
9. https://stackoverflow.com/questions/15228812/crop-image-in-android