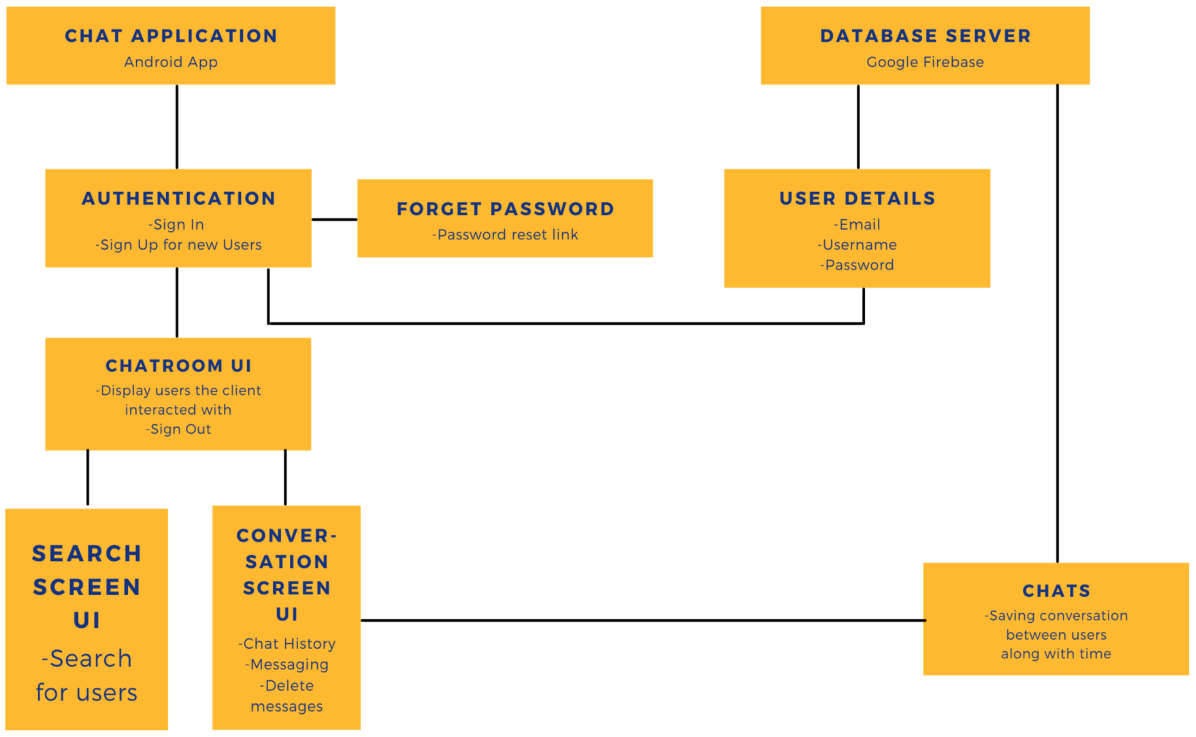
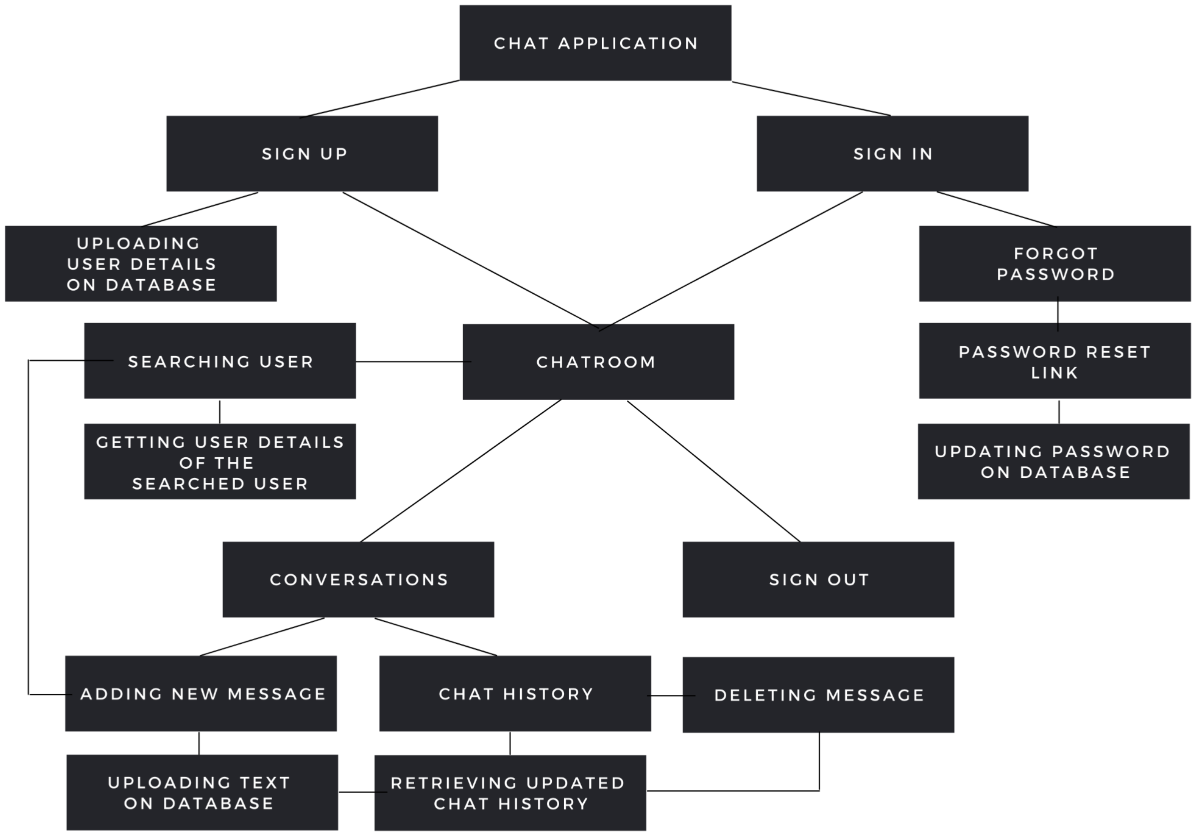
CSL2060
SOFTWARE DESIGN AND ARCHITECTURE

VOILA
CHAT APPLICATION

Software Design and Architecture




CHAT APPLICATION
SIGN UP
SIGN IN
UPLOADING USER DETAILS ON DATABASE
FORGOT PASSWORD
SEARCHING USER
CHATROOM
PASSWORD RESET LINK
GETTING USER DETAILS OF THE SEARCHED USER
UPDATING PASSWORD ON DATABASE
CONVERSATIONS
SIGN OUT
ADDING NEW MESSAGE
CHAT HISTORY
DELETING MESSAGE
UPLOADING TEXT ON DATABASE
RETRIEVING UPDATED CHAT HISTORY

How do we approach it?

The chat consists of two major parts:

1. **Chat Application** which is an android-based chat app.
2. **Database Server** which is an external server responsible for the chat operation. This is the place where all the chat magic happens.

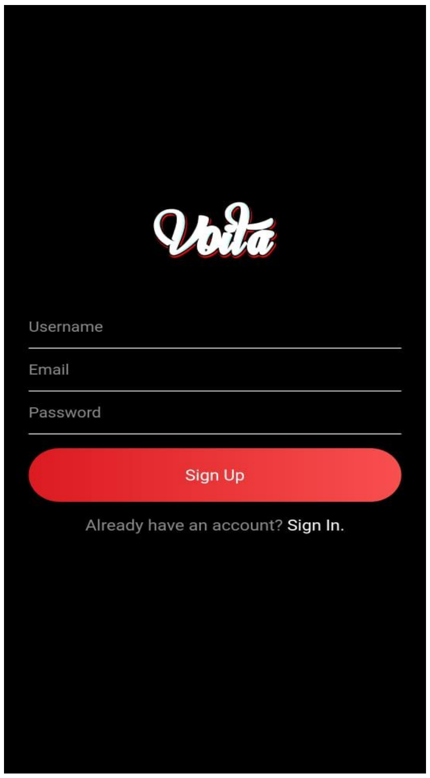
Both parts contain various components that communicate to each other and bring the chat into action.

Database server is a core of the chat architecture that handles message delivery and dispatch. In our version of chat architecture, it includes the following components:

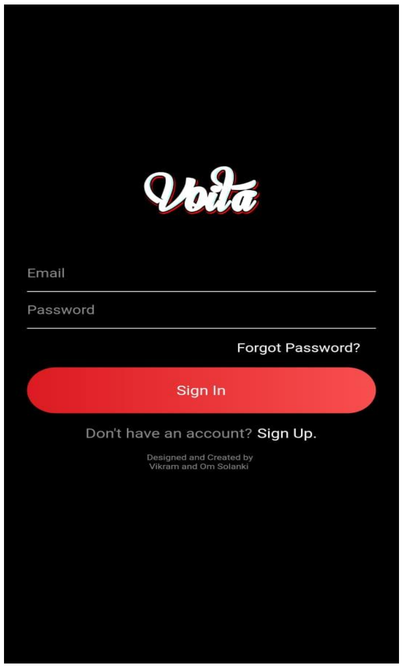
- **User Details** which contains necessary details about the user such as username, email and password. This component handles tasks which are not directly linked to the message delivery and dispatch.
- **Chats** is responsible for transmitting and saving conversation between users along with the time of messaging (hour and minute). This connection is open two ways; that means users don't have to make requests to the server if there are any messages for them, they just get them right away.

Chat Application is the other major part of the chat architecture, the one that users directly interact with. It includes following components:

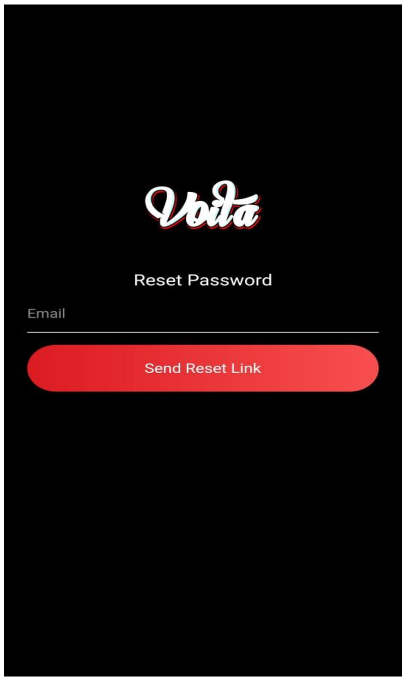
- **Sign Up** New users can sign up by entering credentials (username, email and password). These details entered by the user are updated on the database server (User Detail component).



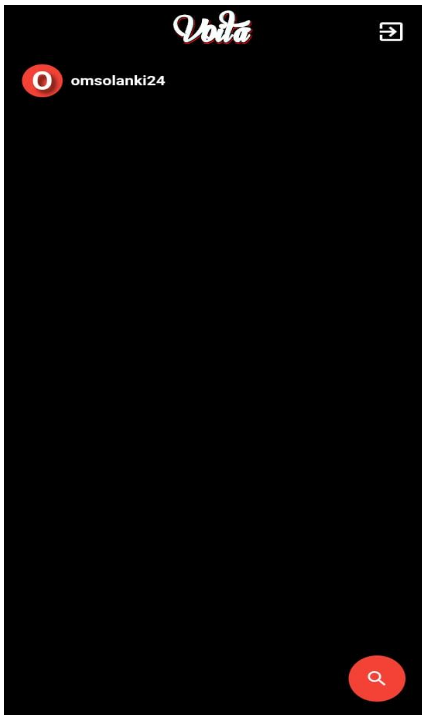
- **Sign In** Registered Users can sign in by entering valid credentials.



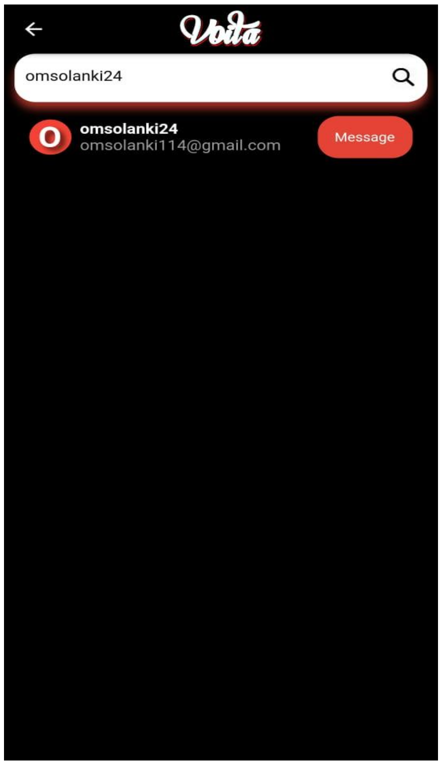
- **Forgot Password** In case, any user forgets his/her password, he/she may use the “Forgot Password?” functionality to reset password via reset password link. The password will be updated on the database server instantly after the password reset.



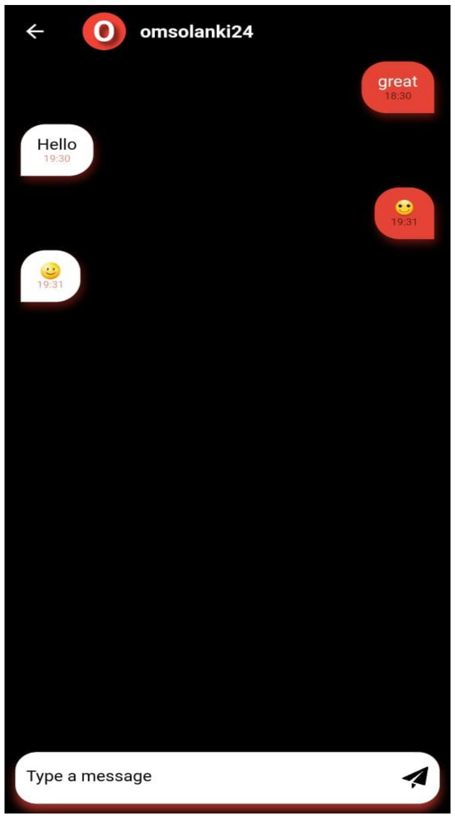
- **Chatroom** Once the user is signed in, he/she will be introduced to this section of the app where the user can see other users whom he/she had recently interacted with.



- **Search** A search option is available in the chatroom, which on clicking enables the client to search for other users by their username. If the searched user exists in the database, the client can see the username and email of the user along with an option to message.



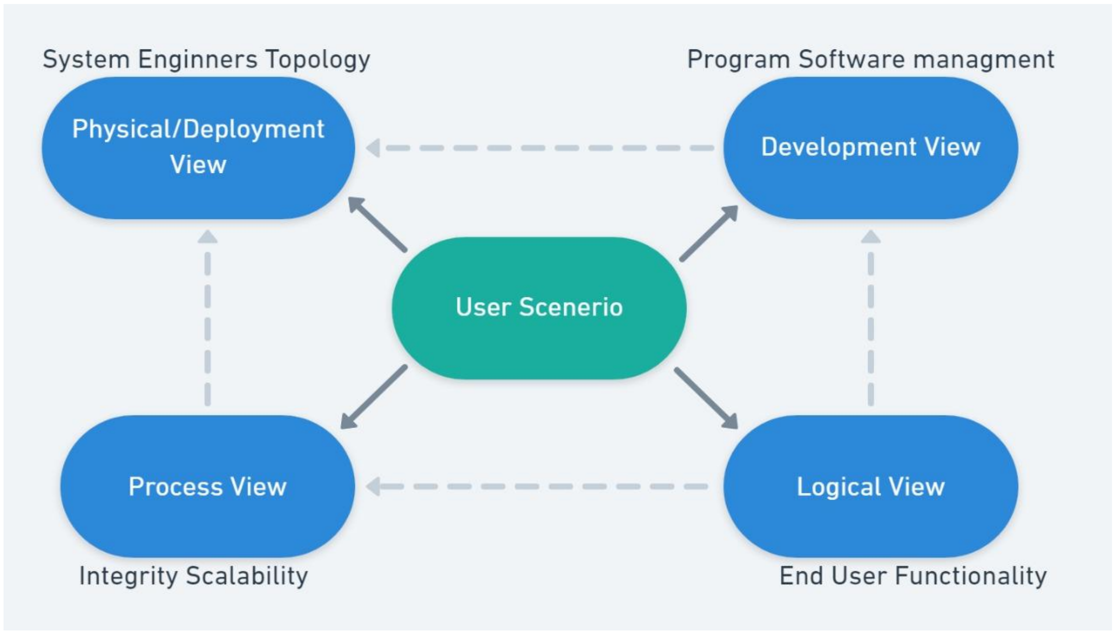
- **Conversations** User can chat with any other user just by clicking on their username. Once clicked, the user will be prompted to the conversation screen where he/she can see the chat history (if it exists). Users can add new messages and the text will be updated on the database server (Chats component). Along with this, chat history will also get updated. Users also have an option to delete any message just by long pressing on it.



- **Sign Out** User can sign out of the app by tapping on the sign out button available in the chatroom screen. Once signed out, the user will see the sign in screen.

4+1 Architecture View

- **Structure**



- **Physical or Deployment View**

Physical View in general contains the names of the different components in handling the submodules present in the main module (Adding a new message to other users in my case).

Different components used are:

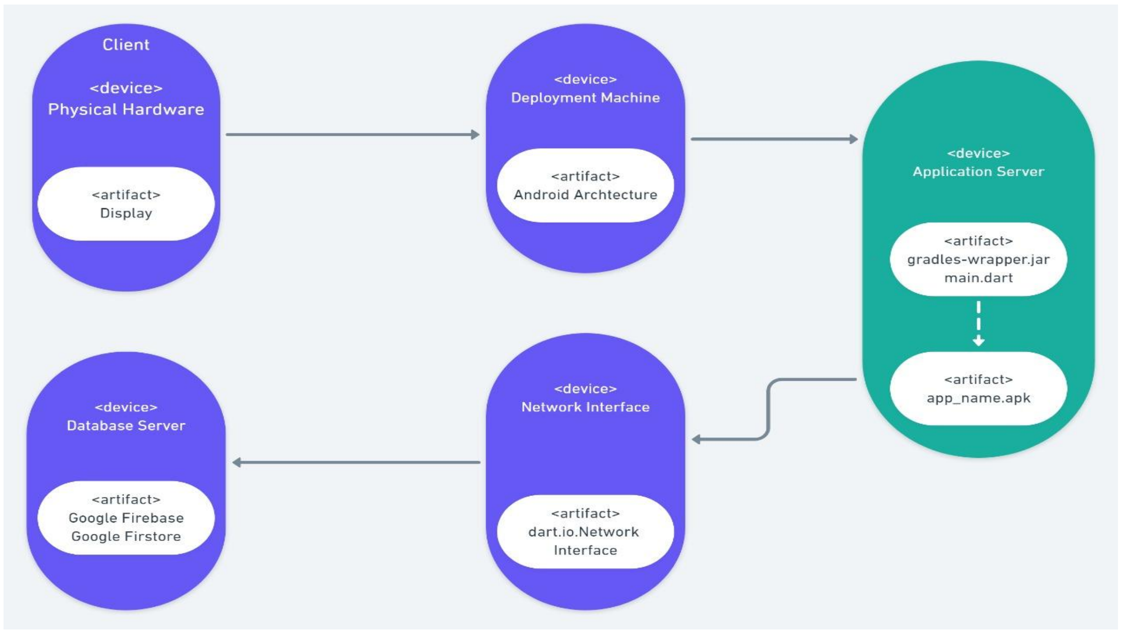
Physical Hardware is the touch screen of an android smartphone for both user input and showing output.

Deployment machine is an android architecture on which an app will be deployed.

Application will interact with android server by app_name.apk to main.dart as flutter-dart framework is used.

Network interface will let the application connect to the internet to connect with the database and other users.

Database Server Google Firestore is used to store user info and all data (conversations).

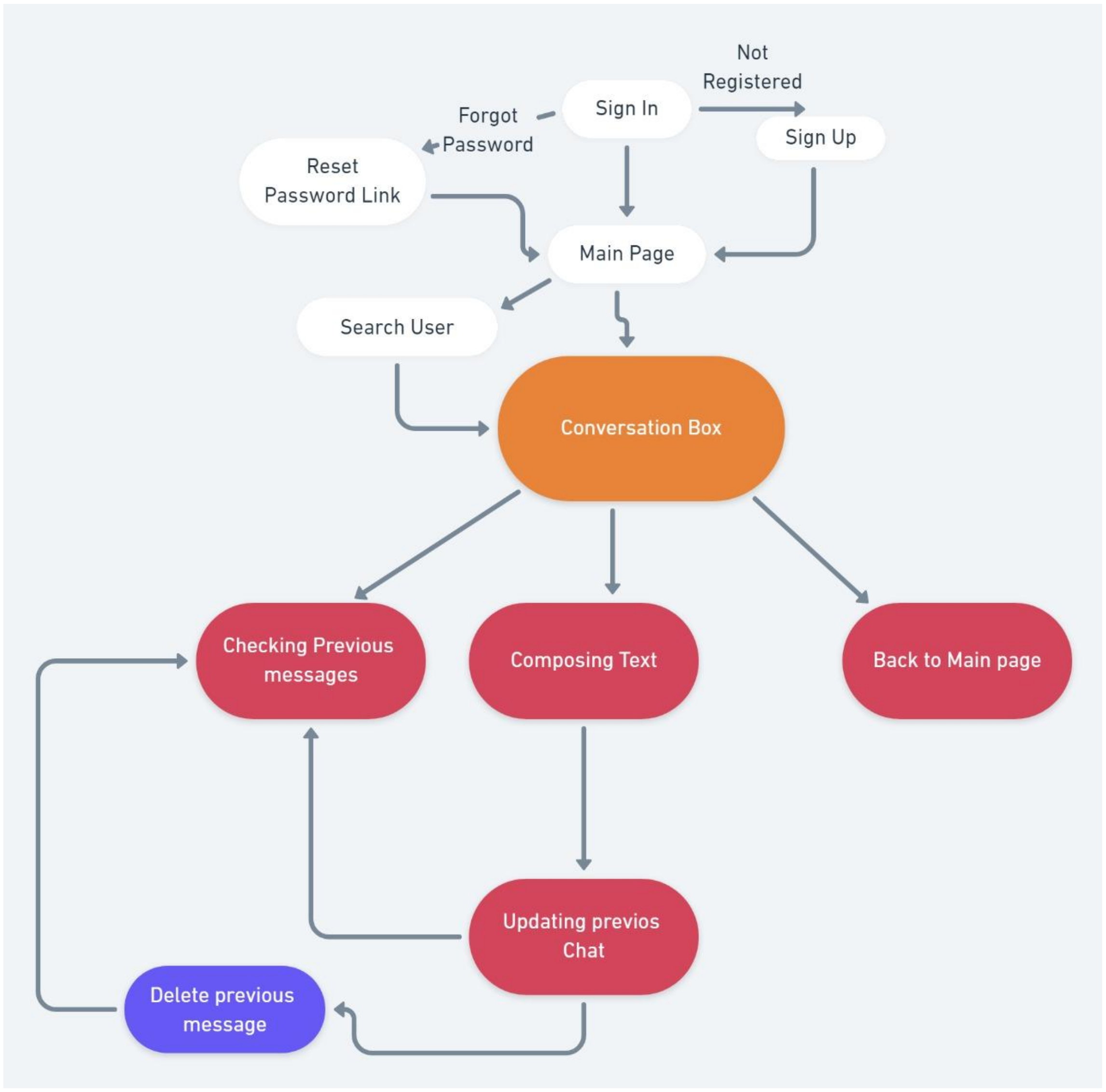


- **Logical View**

Logical view for a single module contains the submodules that compromise the entire functionality there for a user in one use case. Since the use case in my case is to add a message, the listed ones are the submodules for that.

User can write a text message to another user through the conversation screen on his device. Users can also see previous conversation with that user just above the current message in chronological order.

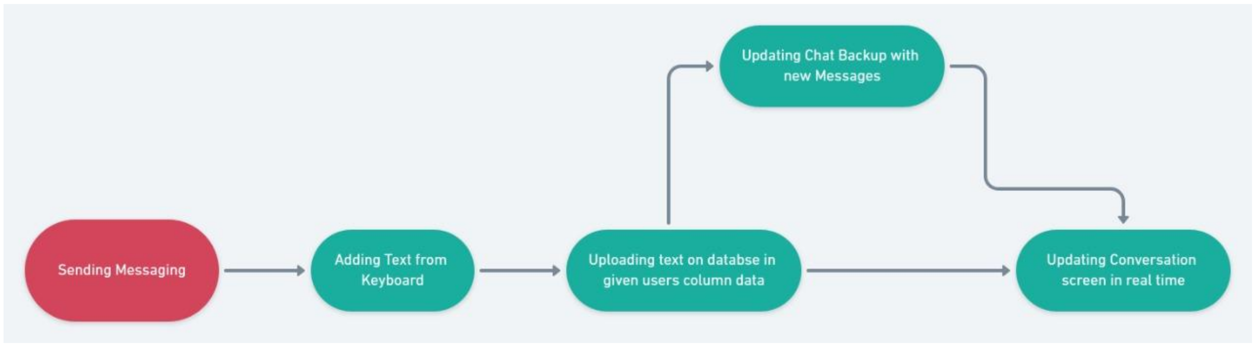
There will be a back button to take the user to the main screen to find another user to chat with. All the conversation will be backed up on the database and can be retrieved after installing the app again.



- **Process View**

The Process View contains sequential steps or processes taken to achieve a task as claimed by the application. In the above labelled diagram, the arrows represent the ordering expected to be taken by the user to accomplish the 'messaging'" scenario.

Users will write text from the local keyboard which will be updated on the database in a data column for a given user. Chat backup will also be update in real time and Conversation screen will also update right after composing the message.

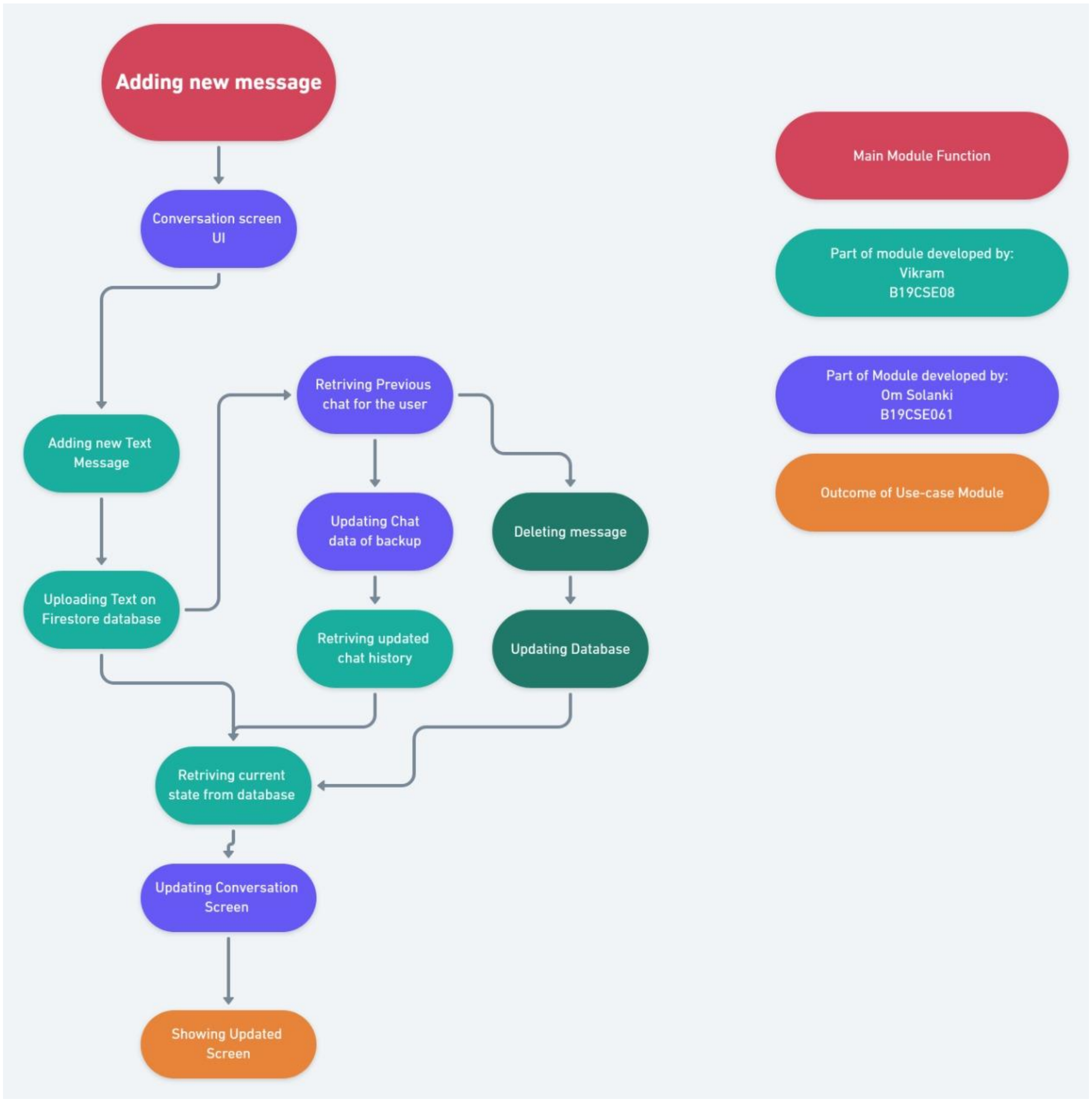

Sending Messaging
Adding Text from Keyboard
Uploading text on databse in given users column data
Updating Chat Backup with new Messages
Updating Conversation screen in real time

- **Development View**

Development view describes which sub-modules are developed by which team mate. In here, all the blocks in mint and purple contain the various submodules to design and code, where Vikram handled sub modules in mint containing message uploading and current state updation on every upload whereas Om Solanki handled the sub-modules in purple which contains updating back and UI of conversation screen and displaying its conversation screen.

Beside this processes involved are explained in process view and timeline is

explained in gantt Chart uploaded in the SRS document.


Adding new message
Conversation screen UI
Adding new Text Message
Uploading Text on Firestore database
Retriving Previous chat for the user
Updating Chat data of backup
Retriving updated chat history
Deleting message
Updating Database
Retriving current state from database
Updating Conversation Screen
Showing Updated Screen
Main Module Function
Part of module developed by: Vikram B19CSE08
Part of Module developed by: Om Solanki B19CSE061
Outcome of Use-case Module

- **Use Case/User Scenario**

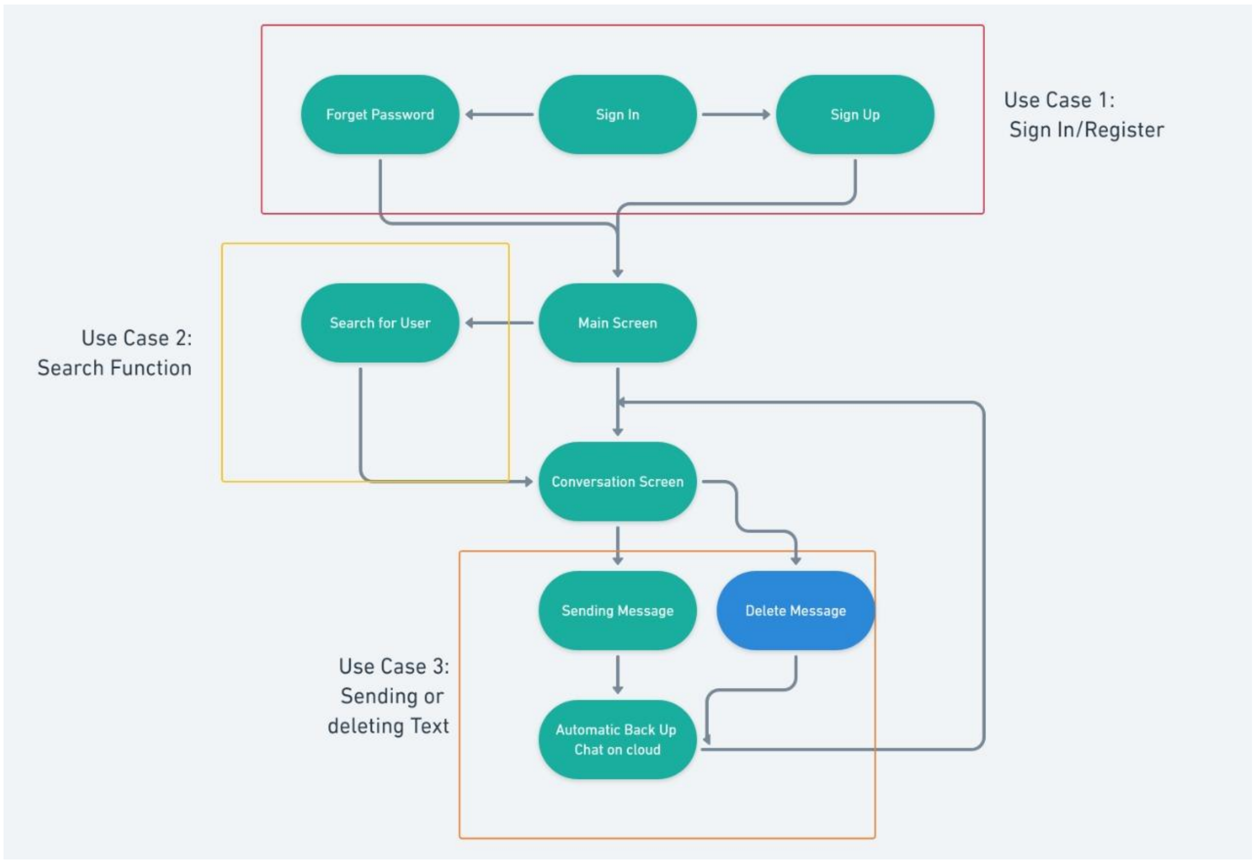
Use Cases are as:

Sign In/ Register (with forgot password)   Search function   Sending text

Registration/Sign In: Users need to have an email account to sign-up in the application. If a user has already registered he can login with his correct credentials. After logging in users can have access to their previous chats. In case if a user failed to provide the correct credentials, he can opt for the forgot password option.

Searching among other users: User can also search among another user among his contact list. He/she can use the provided search bar for this action.

Sending Message: User can see all the previous conversation made with that specific user and can continue to chat on this screen only. In the conversation screen, the user can write his message he wants to convey.



Quality Attributes

- **Reusability** Chat components (Sign In , Sign Up, Forgot Password etc) were designed separately so that they can be used for other products in future by making small or no change.
- **Recoverability** In case of an interruption or a failure, the app can recover the data directly affected and re-establish the desired state. The data like chats and user details can be easily recovered from the database server in case of system failure. The measures for the attribute are the auto recovery ratio and mean breakdown time, which is high for the application.
- **User interface aesthetics** The app provides a user interface which is pleasing and satisfying for the user. Even a novice person can easily operate the app. The qualities for the user interface for this app are:
    1. Simplicity
    2. Consistency
    3. Intuitiveness
    4. Prevention
- **Installability** The app can be successfully installed and/or uninstalled in a specific environment effectively and efficiently.