National Institute of Technology Karnataka Surathkal

Course: Software Engineering Lab (CS303)
Topic: Final Report for Project Hermes

Team Members:
Ashutosh Anand (191CS111)
Alan Tony (191CS207)
Lakshmi Aashish Prateek Janaswamy (191CS225)
Sudarshan Sundarrajan (191CS255)

Table of Contents

1: Introduction

1.1: Project Overview

Hermes is a chat application that is built around the idea of connecting individuals and professionals on one platform that provides both a way of communication as well as a structured and productive environment for professionals.

The goal of this project is to help connect many people to create an online virtual community to share ideas, work or for casual purposes. It is a collaboration platform where individuals or different teams within an organization can discuss, and share ideas. It offers a means to create a productive environment for professionals, and at the same time, it can also be utilized as an application for a casual conversation among individuals/groups.

1.2: Scope of the application

The Software product, as mentioned in the overview, is a chat application that shall essentially provide services such as an environment that a team/group of individuals/employees/peers can utilise to have formal or informal conversations. Given that any chat application is a very time critical product, the platform is designed to be accommodating to real time messaging and be intuitive to use.

1.3: Features offered by Hermes

Hermes offers a plethora of features that enables it to be a productive and casual platform for a broad user base. The list of features is as follows:

1. Channels
2. Messaging Features
3. Roles & privileges within groups
4. Task management & Event scheduling

2: Software Requirements

2.1: System Features

2.1.1: User Dashboard

- This is the home page that is reached after successful authentication.
- Users can access their personal chats as well as their groups through the user dashboard/home page

2.1.2: Messaging Service

- Users will be able to communicate with each other via the messaging service.
- Messaging can be done between two individual accounts or in a group.

2.1.3: Roles

- Roles are a means to identify a subset of people in a group.
- Each user in a group can have multiple roles.
- Each role is associated with a set of permissions.
  Example: HR role has the permission to add users and create channels in a group while the Engineer role has permission to only view and write messages in the channel.
- Roles in a group can be manipulated by users with admin-like roles

2.1.4: Channels

- Each group in our application will have multiple subsections, which can have different purposes. These subsections are termed as a group's "channels".
  Example: Assume a group created for a company can have "channels" for HR, Engineering team, Quarterly meets, Yearly meets and so on.
- A channel is used as a mode of communication between multiple users of a group.
- The permissions a user has over a channel can be decided by the role assigned to them.

2.1.5: Events Scheduler & Calendar

- Each group will have access to its own event scheduler to improve productivity and schedule events. All events that are scheduled can be viewed on the calendar.
- An event can be marked on the calendar with a start date, end date, and a description. The roles that are concerned with the event are also set while creating the event.
- Notifications will alert users for an upcoming event.
- An event can also be rescheduled by the person who created it. Subsequently, a notification is sent to all the other attendees of the event regarding the rescheduling of the event.
- Objectives/descriptions of the event can also be specified when creating an event.

2.2: Non-functional requirements

2.2.1 Performance Requirements

- Hermes aims to enable users to exchange messages at real-time with low latency.
- The application itself should be lightweight and given that the initial version is targeted at mobile users, it should also consume lesser memory.

2.2.2 Security Requirements

- No user is allowed to use Hermes without registering themselves to the service. The user will get access to the services upon successful registration and/or authentication.
- Once users are registered, every time they wish to use the application, they will be prompted to login with their credentials.
- Users will not be permitted to use the same credentials to create new accounts on Hermes
- Apart from authentication, sensitive user information must be stored in a secure manner and should be hidden from other users.

2.2.3 Portability

Hermes should be developed with cross-platform functionality. The initial version of Hermes is targeted at mobile users (iOS 9 and above, Android 8 and above).

2.2.4 Scalability

The application should be scalable in terms of functionality, user base, and database.

2.2.5 Interface Requirements

2.2.5.1 User Interface

- Users should be able to intuitively use Hermes with minimal assistance.
- The user interface should be user-friendly and easy to navigate.

2.2.5.2 Hardware Interfaces

- A functional virtual keyboard or any external keyboard connected to the device for text chat
- A working Internet connection

2.2.6 OS Permissions

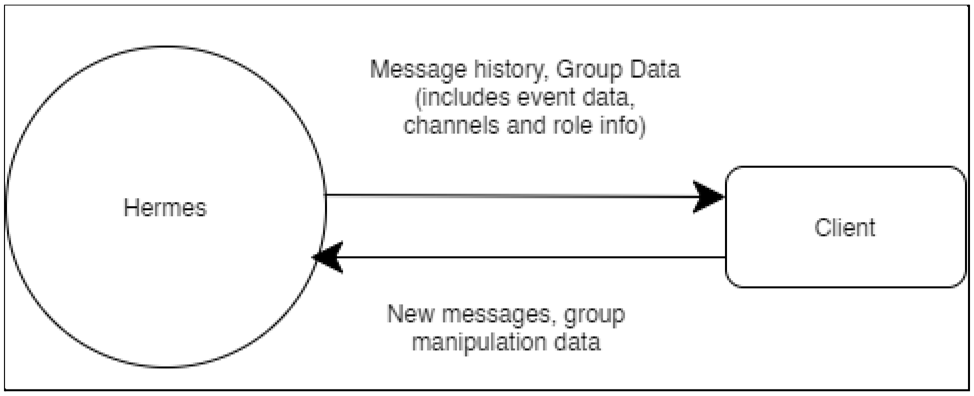
- Contacts
- Microphone
- Notifications

3: Design

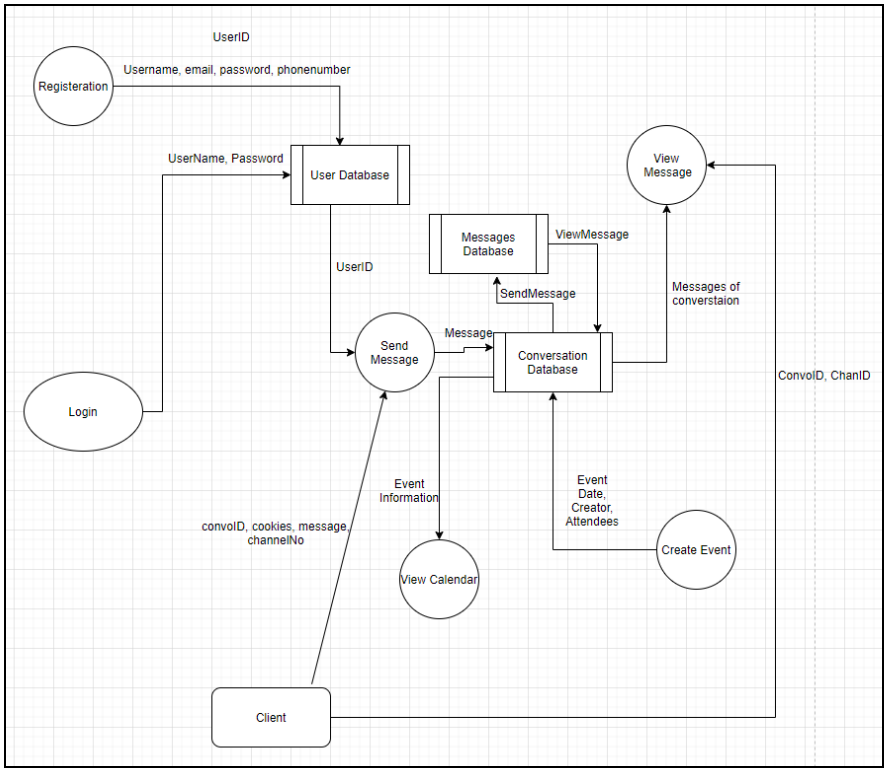
3.1: Data Flow Diagrams

The diagrams below represent the flow of data in the Hermes project.

3.1.1: DFD Level 0 (Context Diagram)

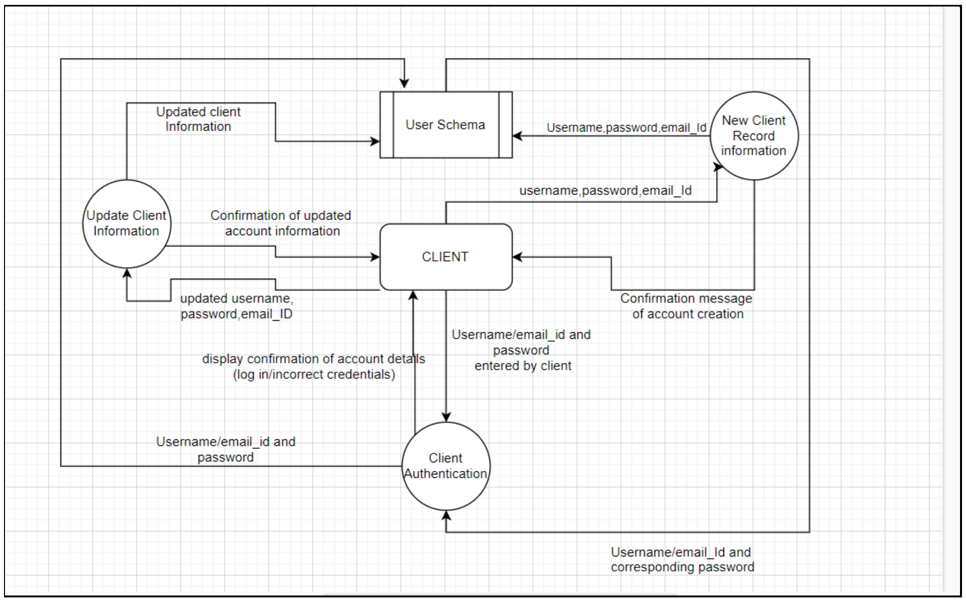

Message history, Group Data (includes event data, channels and role info)
Hermes
Client
New messages, group manipulation data

3.1.2: DFD Level 1

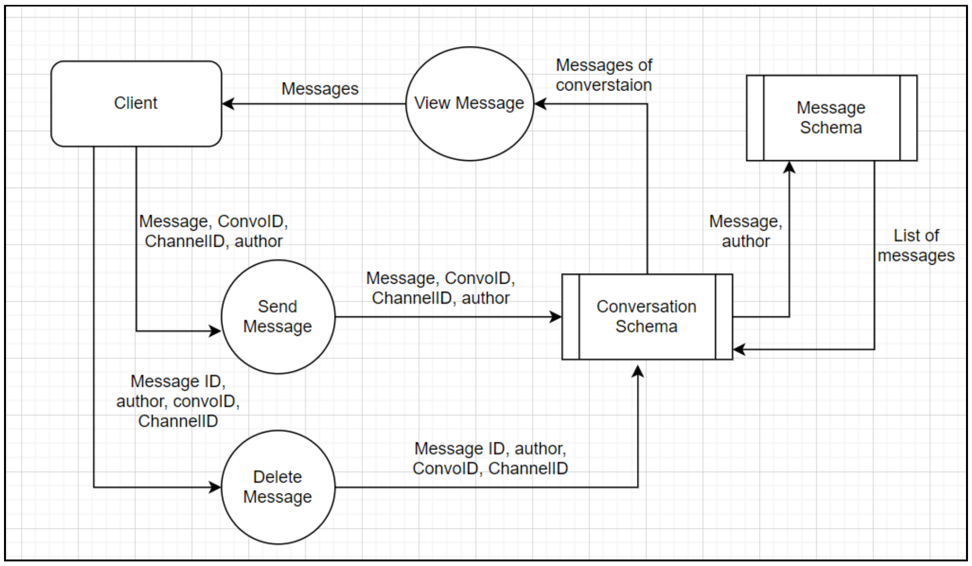


Level 1 DFD for Hermes

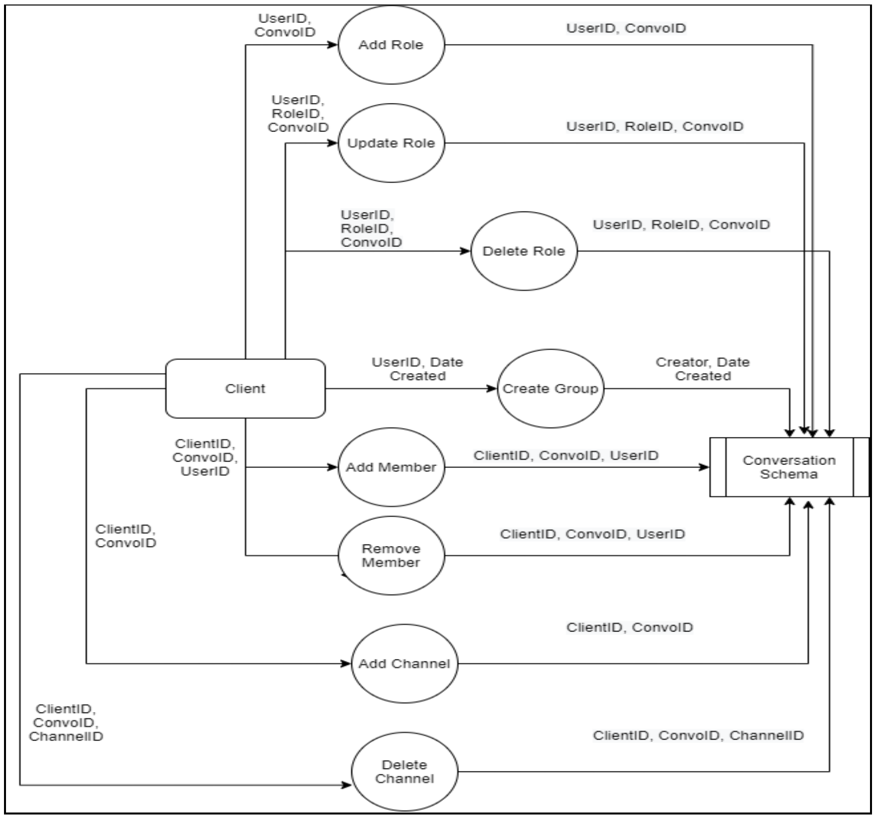
3.1.3 DFD Level 2


User Schema
New Client Record information
Username,password,email_Id
username,password,email_Id
Updated client Information
Update Client Information
Confirmation of updated account information
CLIENT
updated username, password,email_ID
Confirmation message of account creation
Username/email_id and password entered by client
display confirmation of account details (log in/incorrect credentials)
Username/email_id and password
Client Authentication
Username/email_Id and corresponding password

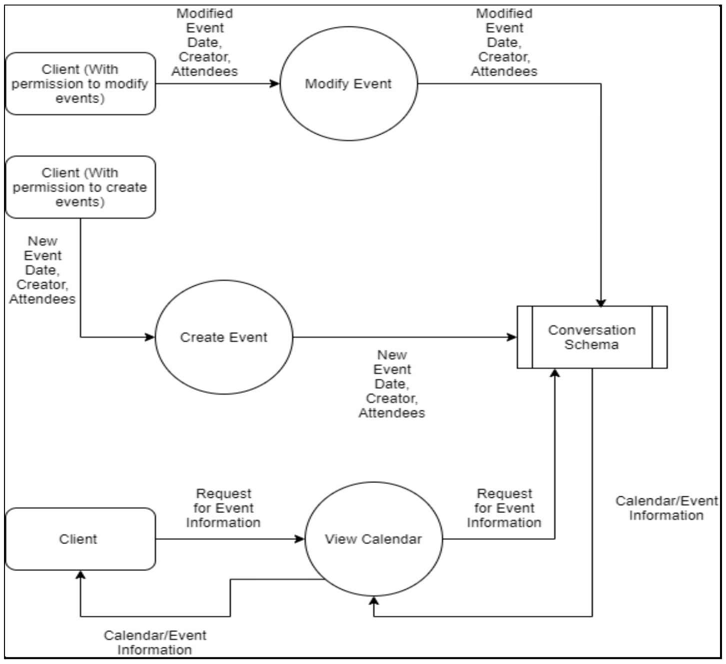
1. *Login/Authentication Module*


Client
Messages
View Message
Messages of converstaion
Message Schema
Message, ConvoID, ChannelID, author
Message, author
List of messages
Send Message
Message, ConvoID, ChannelID, author
Conversation Schema
Message ID, author, convoID, ChannelID
Delete Message
Message ID, author, ConvoID, ChannelID

2. *Message Management Module*



3. <u>Group Management Module</u>

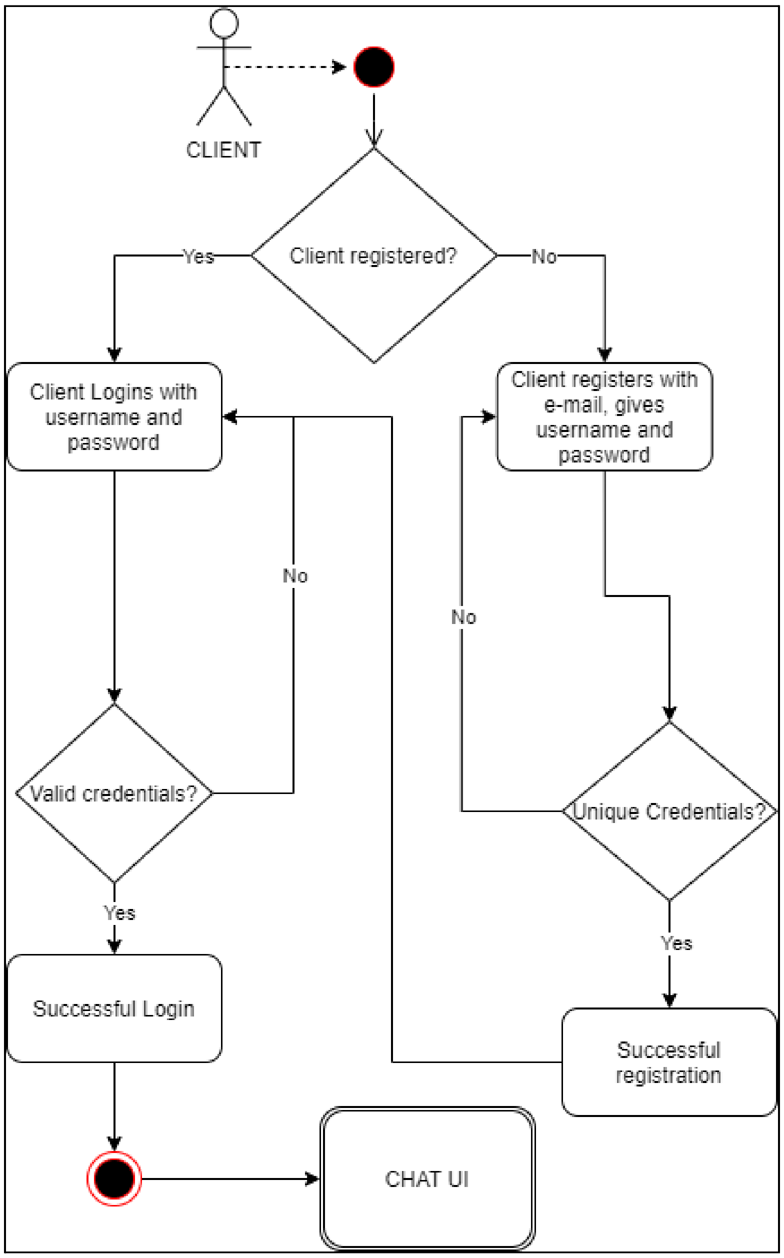

Client (With permission to modify events)
Modified Event Date, Creator, Attendees
Modify Event
Modified Event Date, Creator, Attendees
Client (With permission to create events)
New Event Date, Creator, Attendees
Create Event
New Event Date, Creator, Attendees
Conversation Schema
Client
Request for Event Information
View Calendar
Request for Event Information
Calendar/Event Information
Calendar/Event Information

4. <u>Event Management Module</u>

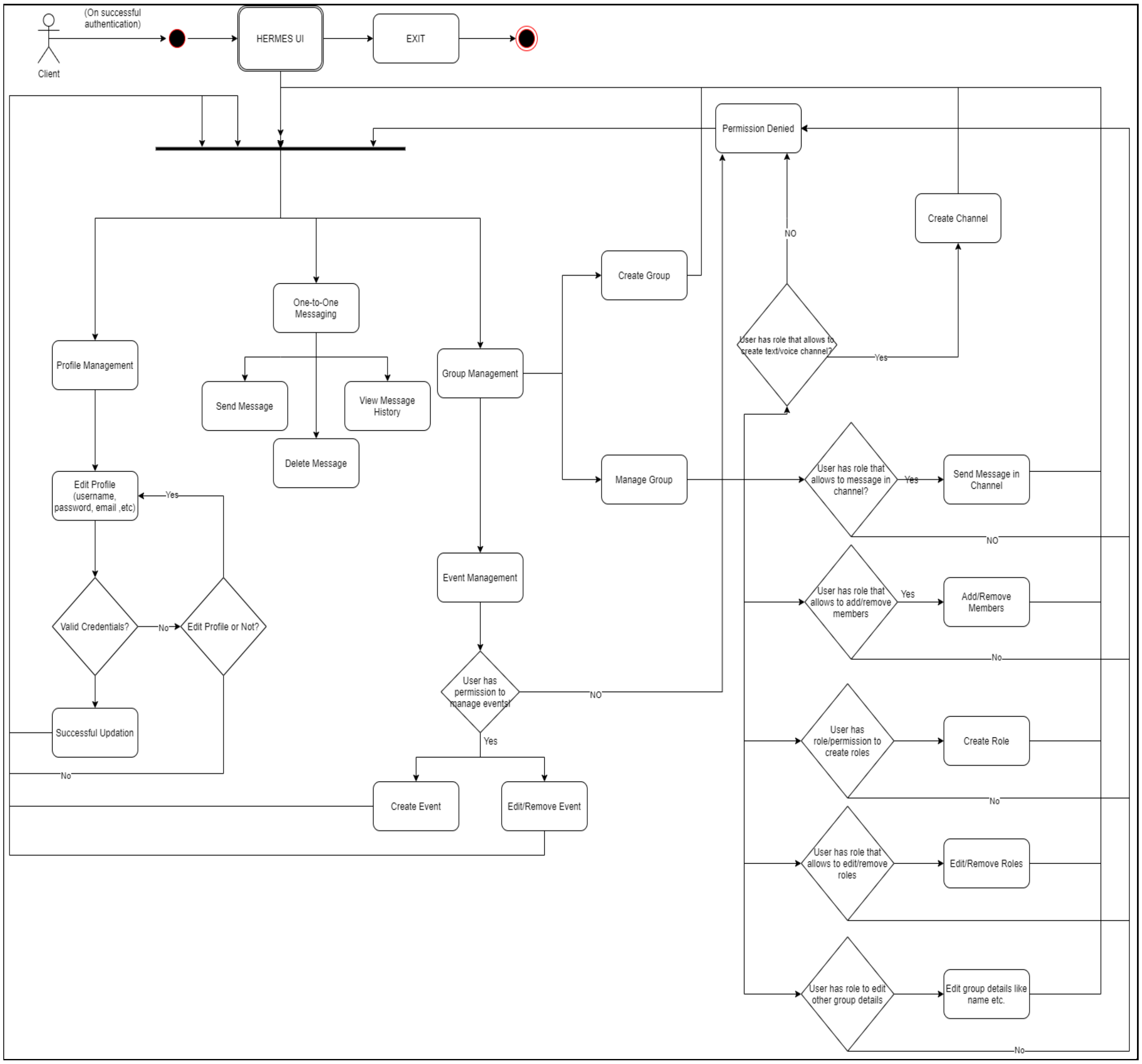
3.2 Activity Diagram

The following diagrams explain the control flow of Hermes. To make the flow clearer, the activity diagram was split into two, one for the login module and the other for the rest of Hermes.

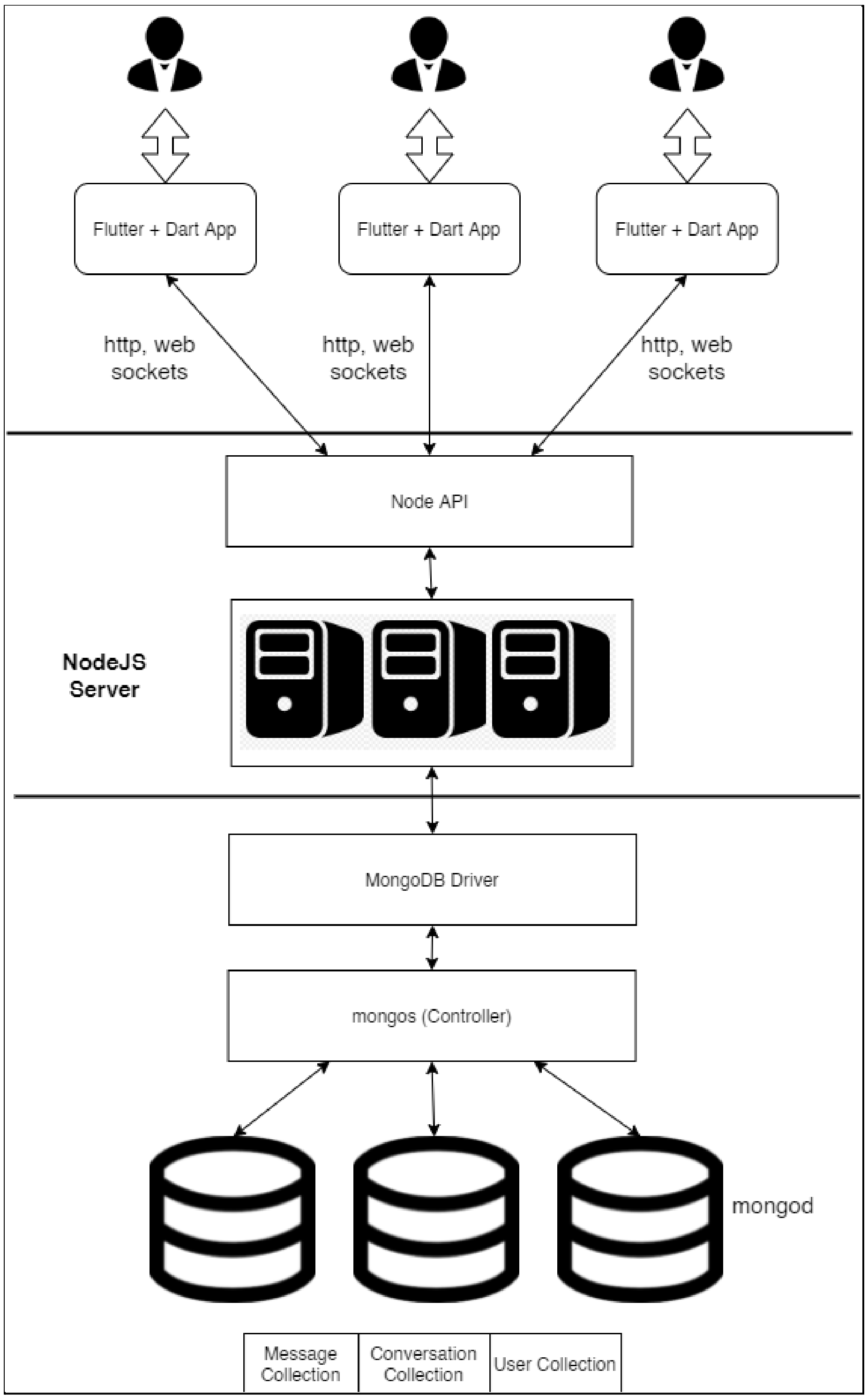
3.2.1 Login/Registration Module



3.2.2 Rest of Hermes


Client
(On successful authentication)
HERMES UI
EXIT
Permission Denied
Create Channel
Create Group
NO
User has role that allows to create text/voice channel?
Yes
One-to-One Messaging
Profile Management
Group Management
Send Message
View Message History
Delete Message
Manage Group
User has role that allows to message in channel?
Send Message in Channel
Edit Profile (username, password, email ,etc)
NO
Event Management
User has role that allows to add/remove members
Add/Remove Members
Valid Credentials?
No
Edit Profile or Not?
User has permission to manage events?
Successful Updation
User has role/permission to create roles
Create Role
Create Event
Edit/Remove Event
User has role that allows to edit/remove roles
Edit/Remove Roles
User has role to edit other group details
Edit group details like name etc.

3.3 System Architecture



4: Implementation

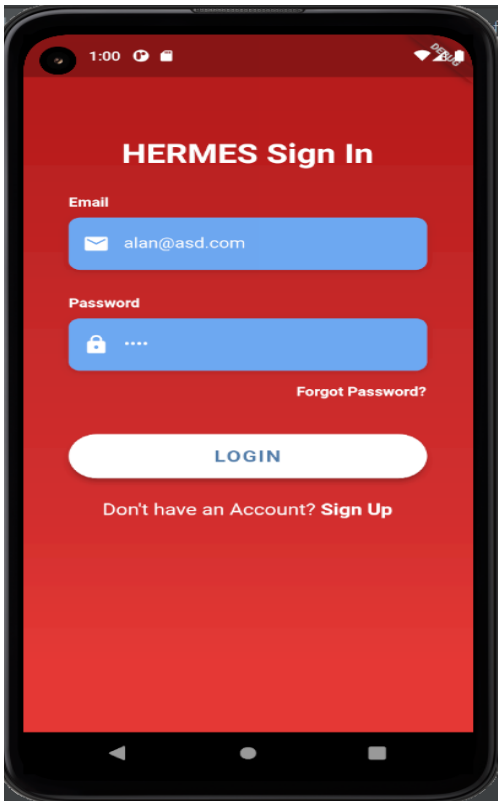
This project was implemented using the following tech stack:

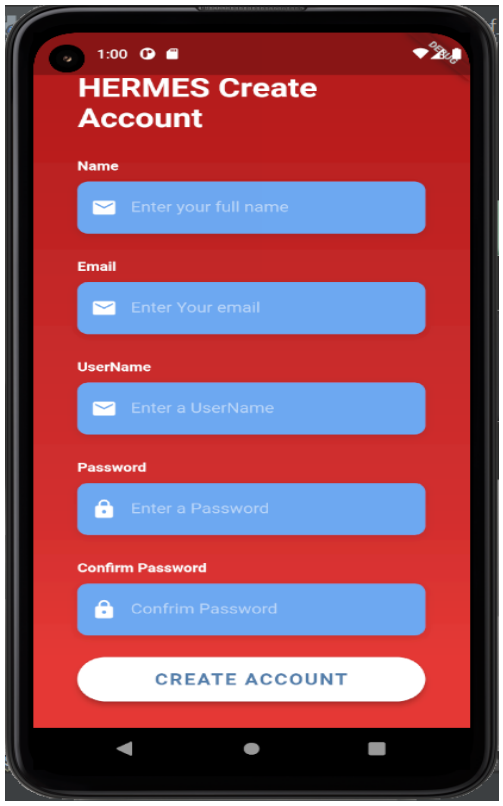
- **Frontend:** Flutter SDK (Dart is the language used)
- **Backend:** Node.js + Express.js (and socket.io for real-time messaging)
- **Database:** MongoDB

4.1 Frontend Implementation

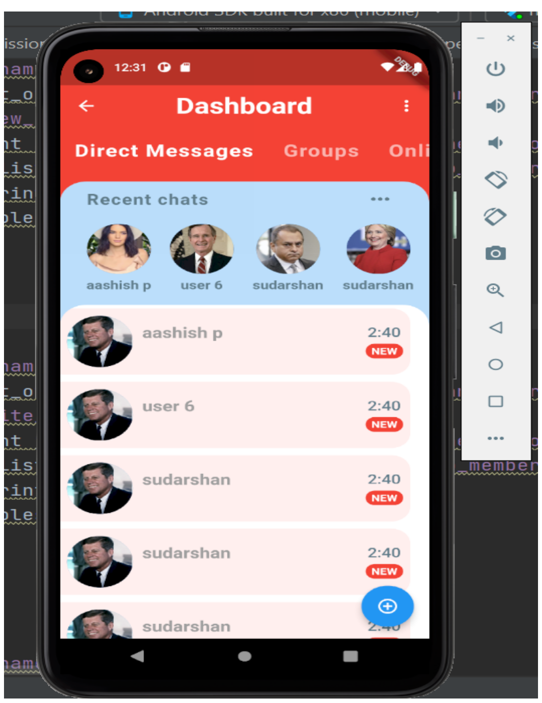
As mentioned above, Flutter was used to implement the UI elements for Hermes. The UI screenshots for the various modules are attached below.

4.1.1 Login Page and Registration Page





4.1.2 User Dashboard and Direct Messaging



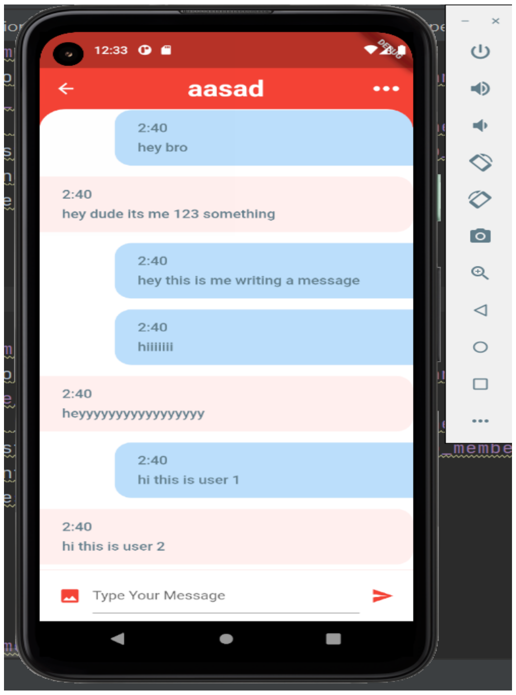
aashish p 2:40 NEW

user 6 2:40 NEW

sudarshan 2:40 NEW

sudarshan 2:40 NEW

sudarshan 2:40



2:40
hey bro

2:40
hey dude its me 123 something

2:40
hey this is me writing a message

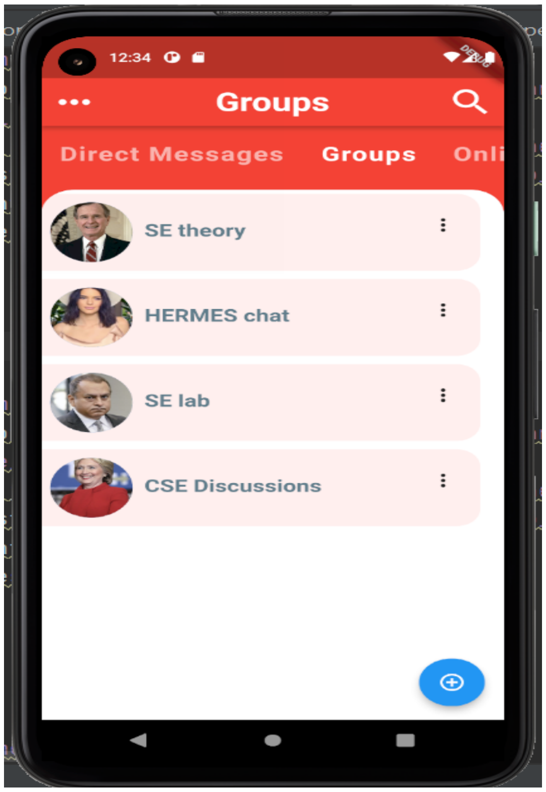
2:40
hiiiiiii

2:40
heyyyyyyyyyyyyyyyyy

2:40
hi this is user 1

2:40
hi this is user 2

4.1.3 Groups List



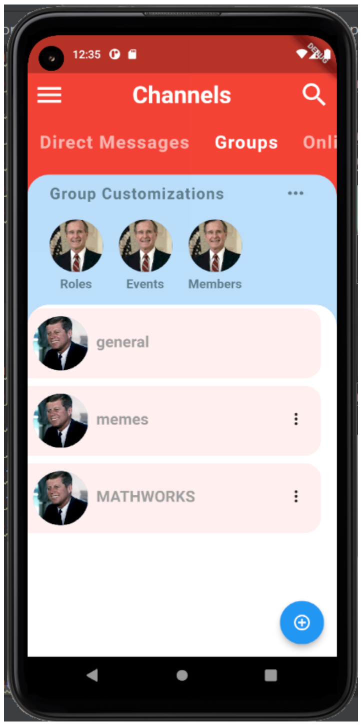
SE theory

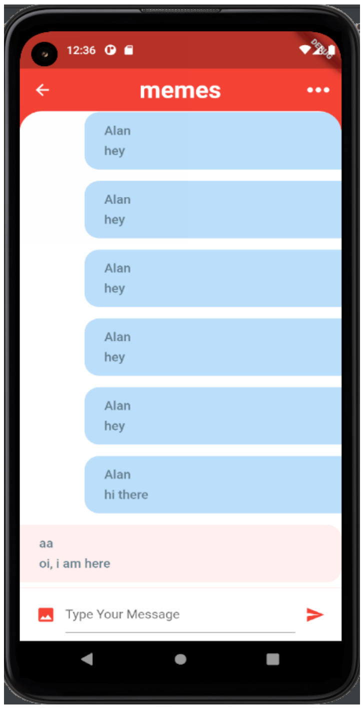
HERMES chat

SE lab

CSE Discussions

4.1.4 Channels, Channel Chat Window and Members in a Group





Alan
hey

Alan
hey

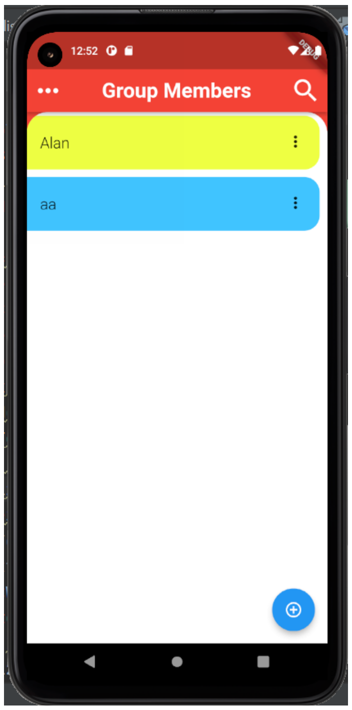
Alan
hey

Alan
hey

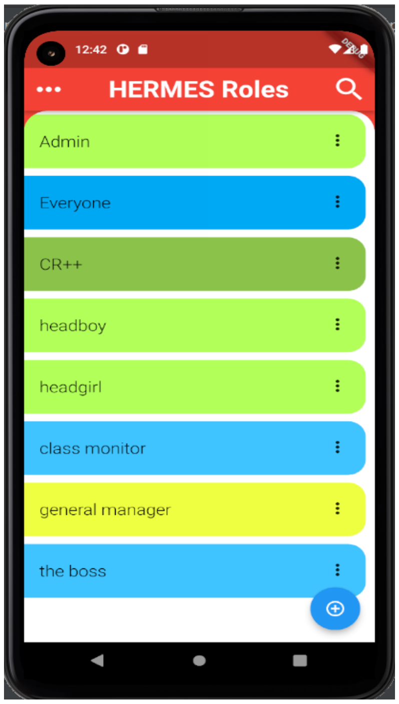
Alan
hey

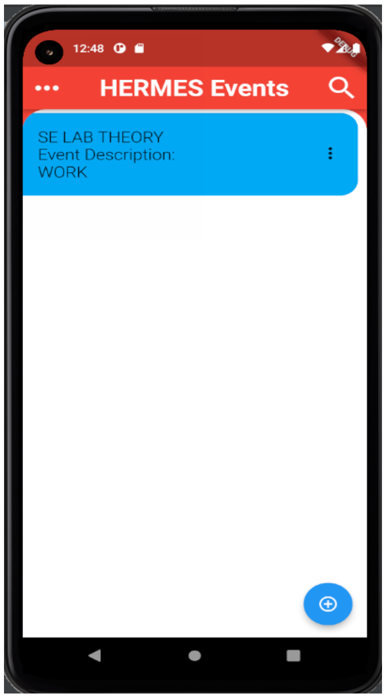
Alan
hi there

aa
oi, i am here



4.1.5 Roles and Events in a Group





SE LAB THEORY
Event Description:
WORK

5: Testing

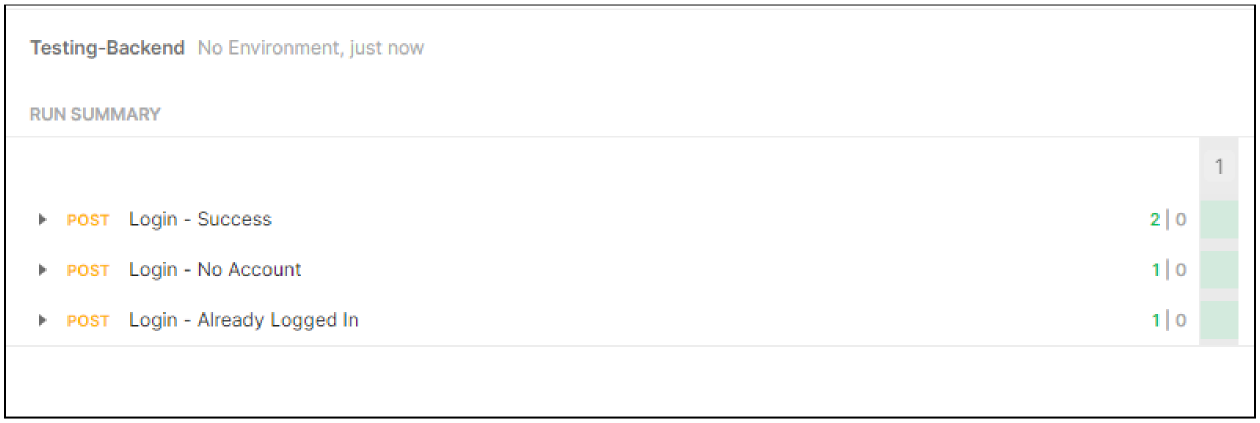
For Hermes, unit and integration testing was done using Postman for the backend and the frontend was manually tested for edge cases.

5.1 Backend Testing

Using the Postman application, we were able to automate our backend testing for both unit testing as well as integration testing. We based our pass/fail criteria for the test cases using a success flag along with the status code returned by the response.

The following images show the result of a unit test conducted for login as an example and also the results from the integration testing.

5.1.1 Unit Test of Login



Testing-Backend No Environment, just now

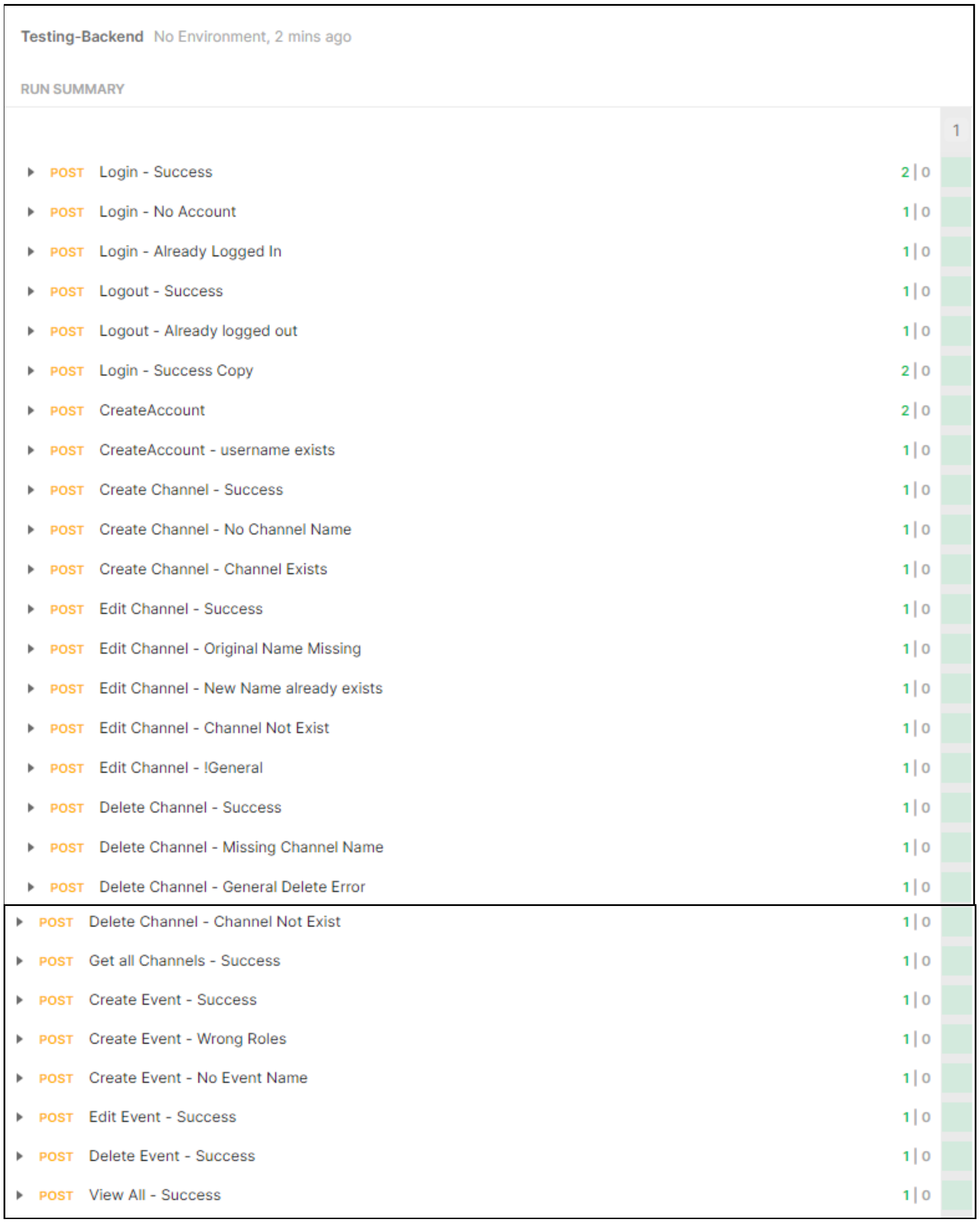
RUN SUMMARY

			1
POST	Login - Success	2 | 0	🟩
POST	Login - No Account	1 | 0	🟩
POST	Login - Already Logged In	1 | 0	🟩

The light green bars on the side signify successful completion of the unit test for login. We had three test cases:

1. Successful Login
2. Login with an invalid account
3. Logging in when the user is already logged in

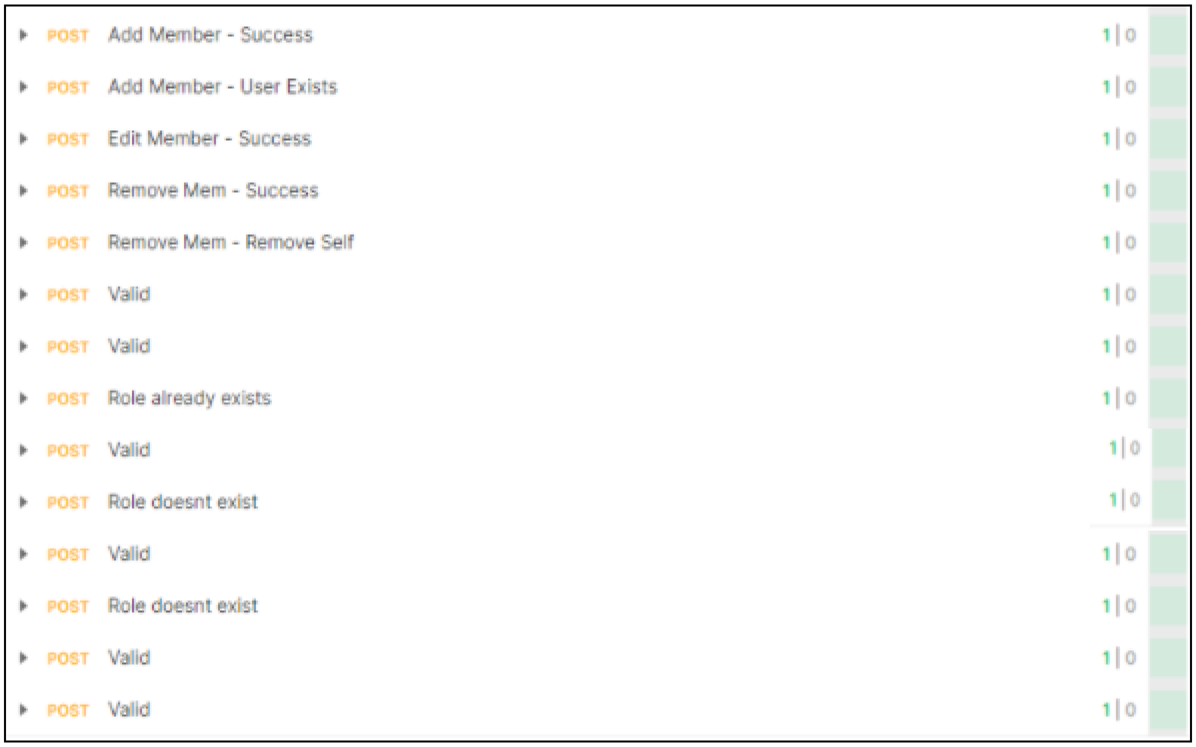
5.1.2 Integration Testing



Testing-Backend No Environment, 2 mins ago

RUN SUMMARY

			1
POST	Login - Success	2 | 0	
POST	Login - No Account	1 | 0	
POST	Login - Already Logged In	1 | 0	
POST	Logout - Success	1 | 0	
POST	Logout - Already logged out	1 | 0	
POST	Login - Success Copy	2 | 0	
POST	CreateAccount	2 | 0	
POST	CreateAccount - username exists	1 | 0	
POST	Create Channel - Success	1 | 0	
POST	Create Channel - No Channel Name	1 | 0	
POST	Create Channel - Channel Exists	1 | 0	
POST	Edit Channel - Success	1 | 0	
POST	Edit Channel - Original Name Missing	1 | 0	
POST	Edit Channel - New Name already exists	1 | 0	
POST	Edit Channel - Channel Not Exist	1 | 0	
POST	Edit Channel - !General	1 | 0	
POST	Delete Channel - Success	1 | 0	
POST	Delete Channel - Missing Channel Name	1 | 0	
POST	Delete Channel - General Delete Error	1 | 0	
POST	Delete Channel - Channel Not Exist	1 | 0	
POST	Get all Channels - Success	1 | 0	
POST	Create Event - Success	1 | 0	
POST	Create Event - Wrong Roles	1 | 0	
POST	Create Event - No Event Name	1 | 0	
POST	Edit Event - Success	1 | 0	
POST	Delete Event - Success	1 | 0	
POST	View All - Success	1 | 0	

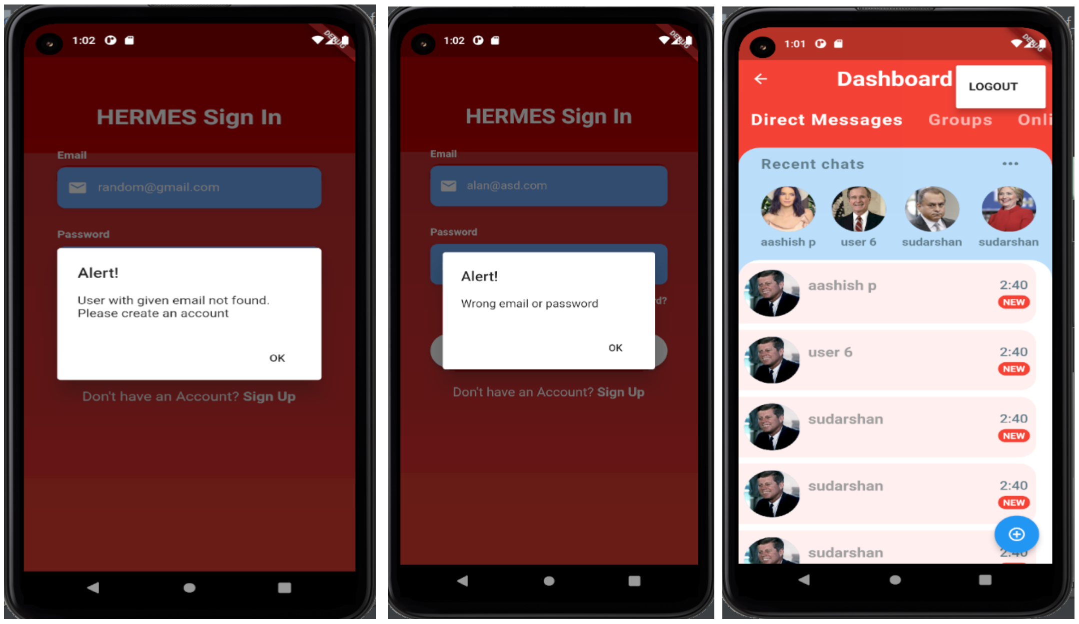


```
POST Add Member - Success        1 | 0
POST Add Member - User Exists    1 | 0
POST Edit Member - Success       1 | 0
POST Remove Mem - Success        1 | 0
POST Remove Mem - Remove Self    1 | 0
POST Valid                       1 | 0
POST Valid                       1 | 0
POST Role already exists         1 | 0
POST Valid                       1 | 0
POST Role doesnt exist           1 | 0
POST Valid                       1 | 0
POST Role doesnt exist           1 | 0
POST Valid                       1 | 0
POST Valid                       1 | 0
```

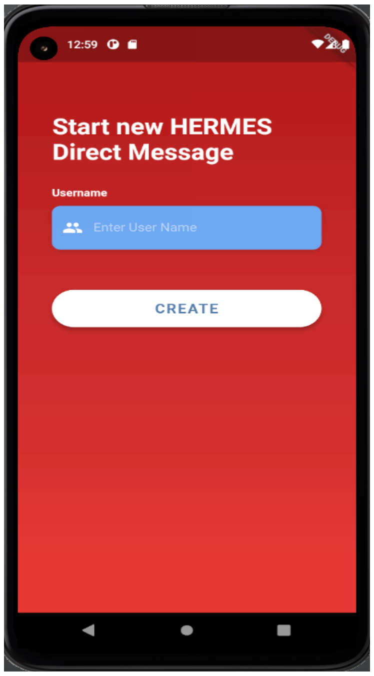
5.2 Frontend Testing

All features of the frontend were manually tested for various scenarios:

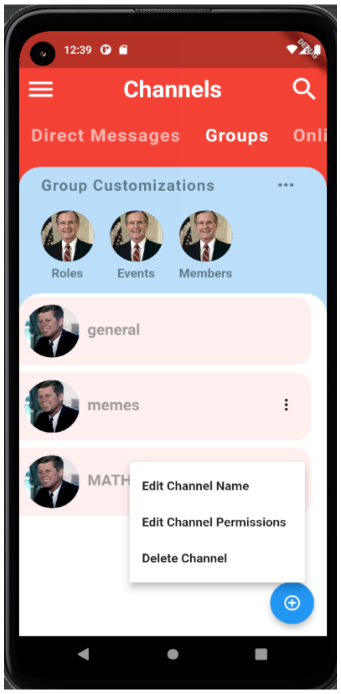
5.2.1 User Service: Login with invalid username and password, logout

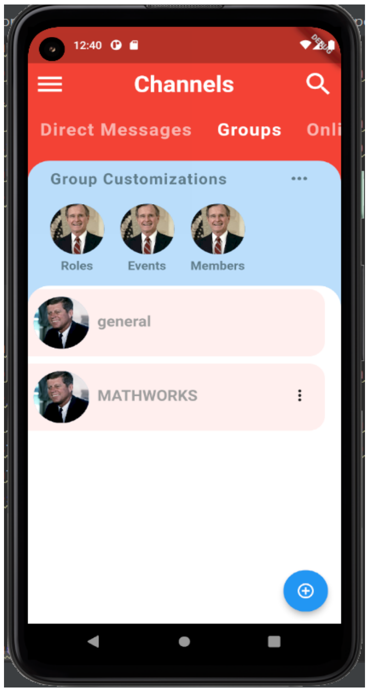


5.2.2 Create DM

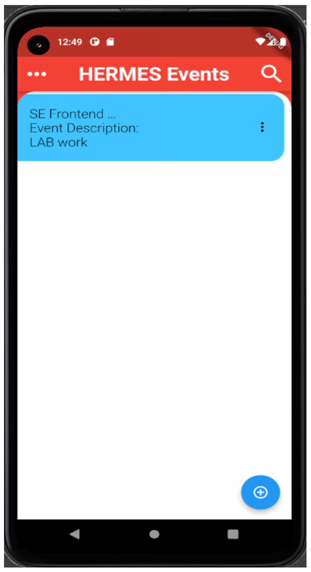


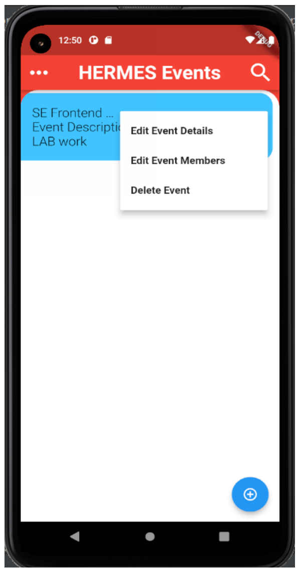
5.2.3 Group channel options and channel list after deleting channel

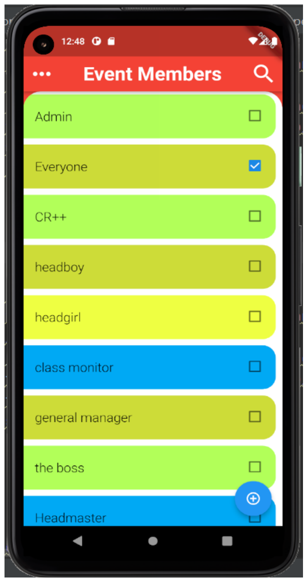




5.2.4 Group event details, event options, and edit event roles

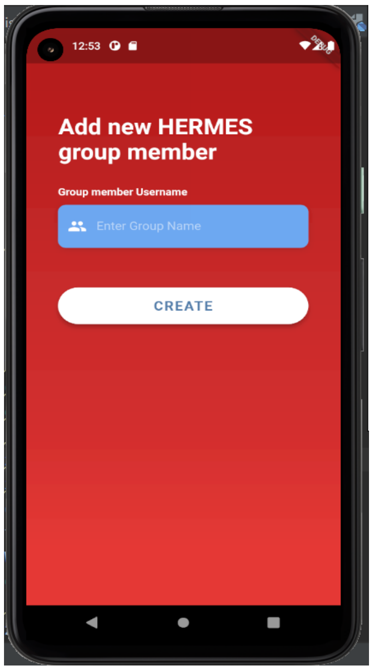


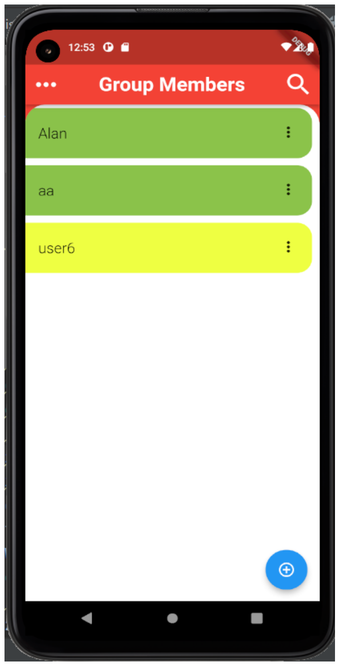


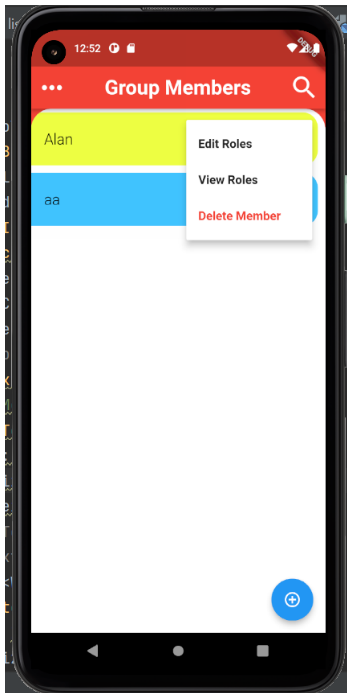


5.2.5 Group Members

5.2.5.1 Add group member and member options

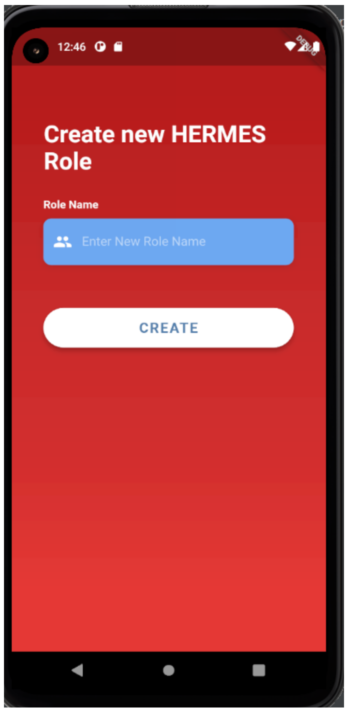


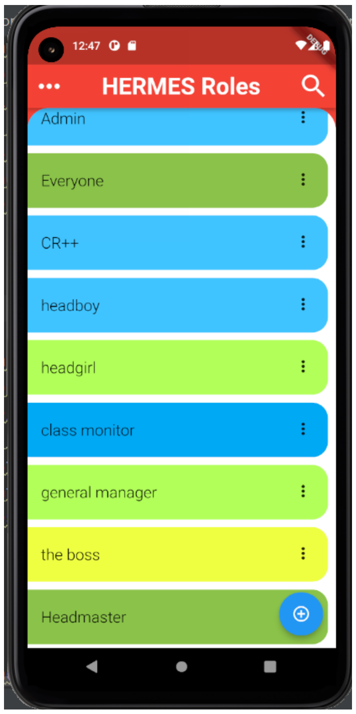




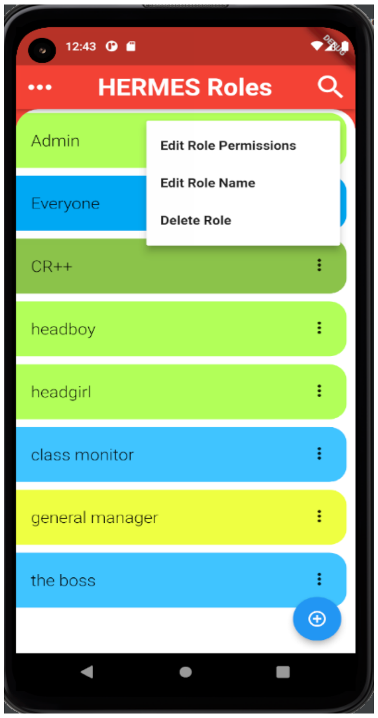
5.2.6 Roles

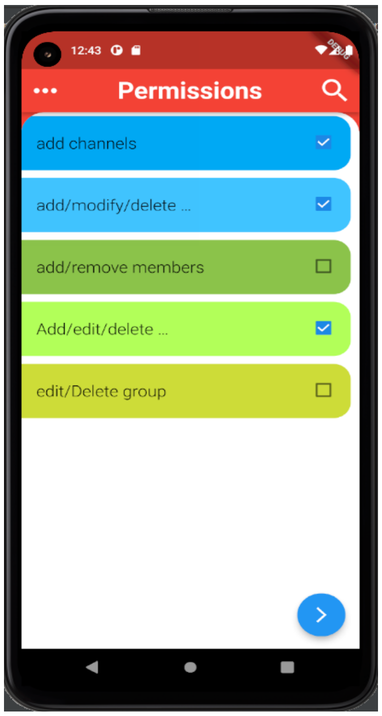
5.2.6.1 Create role

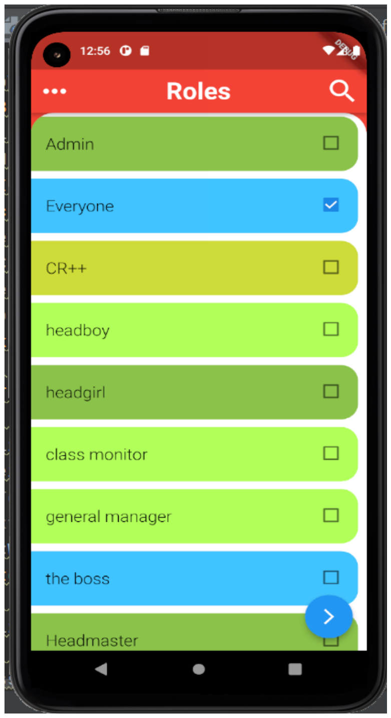




5.2.6.2 Edit Roles







6: Maintenance

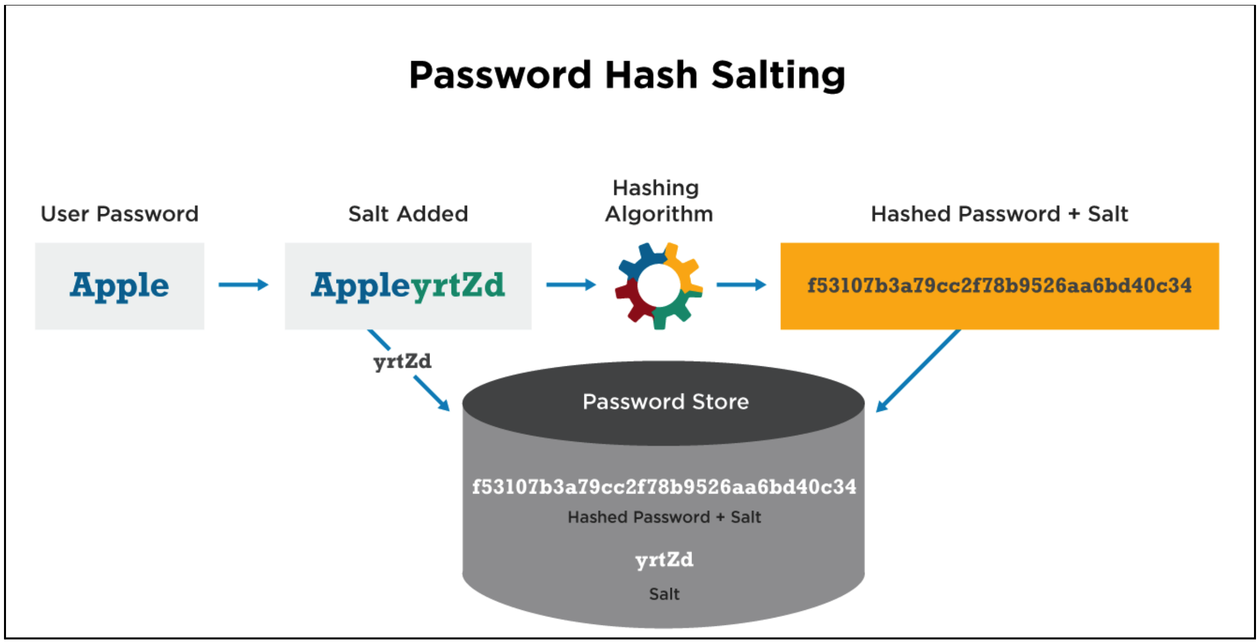
6.1: Security

Hermes employs various security measures to ensure user privacy and security. The measures are as follows.

6.1.1 Hashed and Salted Passwords

To prevent possibilities of leaking passwords from the database, we stored the salted and hashed password. Salting is the concatenation of a random string of characters to the passwords. The resulting string is hashed. Hashed passwords are hard to crack because they are scrambled versions of themselves.

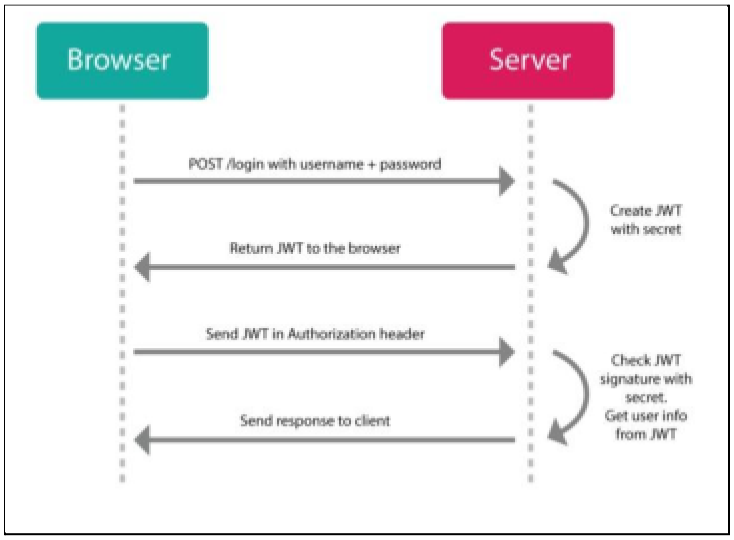
Large salts also protect against certain methods of attack on hashes, including rainbow tables orlogs of hashed passwords previously broken. Both hashing and salting can be repeated more than once to increase the difficulty in breaking the security.



6.1.2 JWT Token Authentication

JWT Tokens to authorize users to make the browsing along the application seamless while maintaining high security and privacy for the user. JWT tokens are a compact and self-contained way for securely transmitting information between parties as a JSON object. The user credentials are embedded into a string encrypted with a private key at the server side. The token will be safe with the user. Each JWT token can have a time of expiry, after which the token becomes invalid. This way, the client need not pass credentials and be authenticated for each request made to the server, but rather send just the token sent to the client on logging in.

JWT token also prevents session high-jacking where the user's credentials are logged, or cached in an intermediate server and exploited. Also no one can pretend to be the client after the original client has logged in since only the client has the token.


Browser
Server
POST /login with username + password
Create JWT with secret
Return JWT to the browser
Send JWT in Authorization header
Check JWT signature with secret. Get user info from JWT
Send response to client

6.1.3 HTTPS Connection

To maintain secure connection and communication between client-server for text messaging and authorization we use HTTP secured connection. It prevents the Man-in-the-Middle (MitM) attacks, in which hackers intercept the network traffic and steal confidential data like passwords, Aadhar numbers, birthdays, etc.

We generated a self-signed SSL certificate and passed it to the user. After the TLS handshake, the network traffic exchanged between the server and clients are encrypted by public-private key encryption.

6.1.4 Permission Management

Each group in the application has various roles that its members can define. Roles with a fixed set of permissions help managing thousands of users in a conversation, thus along with the authorizing middleware we also provide the Permission middleware.

6.2: Portability

After deploying the software to the cloud we hope to maintain a Node web server and a Mongo DataBase server. The client side software is to be installed as an apk from the client side. DNS servers are to be maintained and secured too.

After deploying we continue to maintain the servers by continuously monitoring traffic and testing various edge cases if they ever arise. Any new code is to be heavily tested on a test server (for testing) before actually pushing the code to production.

******