KWIM MESSENGER CHAT APPLICATION

Major project report submitted in partial fulfilment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

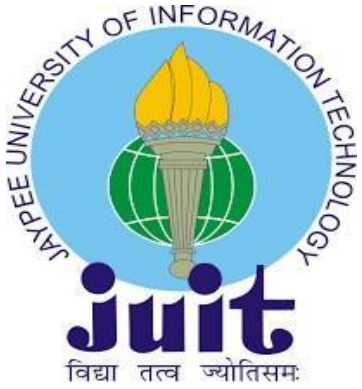
By

PARIKSHIT SHARMA (191278)

UNDER THE SUPERVISION OF

Dr. Deepak Gupta

to



Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled "KWIM Messenger Chat Application" in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering/Information Technology submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from July 2022 to May 2023 under the supervision of Dr. Deepak Gupta, Assistant Professor (SG), Department of Computer Science & Engineering. The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature)

Parikshit Sharma (191278)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)

Dr. Deepak Gupta

Assistant Professor (SG)

Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat

Dated:

CERTIFICATE

This is to certify that the work which is being presented in the Project report titled "**Kwim Messenger Chat Application**" in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering** and submitted to the Department of Computer Science & Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by "Parikshit Sharma (191278)" during the period from February 2023 to May 2023 under the supervision of Dr. Deepak Gupta, Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.

(Student Signature)
Parikshit Sharma, 191278

The above statement made is correct to the best of my knowledge.

(Supervisor Signature)

Dr Deepak Gupta

Assistant Professor (SG)

Computer Science & Engineering and Information Technology
Jaypee University of Information Technology, Waknaghat

Dated:

PLAGIARISM CERTIFICATE

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: .....

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: _____ **Department:** _____ **Enrolment No** _____

Contact No. _____ **E-mail.** _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

_____

_____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at .....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

<table>
<tr><th>Copy Received on</th><th>Excluded</th><th>Similarity Index (%)</th><th colspan="2">Generated Plagiarism Report Details (Title, Abstract & Chapters)</th></tr>
<tr><td rowspan="2"></td><td rowspan="4">• All Preliminary Pages
• Bibliography/Images/Quotes
• 14 Words String</td><td rowspan="2"></td><td>Word Counts</td><td></td></tr>
<tr><td rowspan="2">Report Generated on</td><td>Character Counts</td><td></td></tr>
<tr><td rowspan="2"></td><td>Submission ID</td><td>Total Pages Scanned</td><td></td></tr>
<tr><td></td><td>File Size</td><td></td></tr>
</table>

Checked by
Name & Signature **Librarian**

.....

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

ACKNOWLEDGEMENT

First, I express my gratitude to God who provided me with the courage and fortitude to complete the project.

I am grateful and wish my profound indebtedness to my Mentor Mr. Karan Soni, Team Lead, Flutter at enAct eServices, Rupnagar, Punjab and my Supervisor Dr Deepak Gupta, Assistant Professor (SG), Department of CSE Jaypee University of Information Technology, Wakhnaghat. With their Deep Knowledge & keen interest in their respective fields, my mentor and supervisor helped me to carry out this Project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this Project.

I would like to express my heartiest gratitude to Dr Deepak Gupta, Department of CSE, for his kind help to finish my Project.

I would also generously welcome each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a success. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patience of my parents.

Parikshit Sharma
191278

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Communication is the most basic need in today's world, expressing ones view and making others understand them is key for development of world. Social Media platform have enabled human beings to communicate from any corner of world to any person in the world. Social Media became the bridge for quick communication and interaction amongst people all over the world.

Social Media giants like Facebook, Twitter, and LinkedIn allow users to communicate with each other using Text Based Communication. Platforms like Instagram and Snapchat allow users to communicate through photos and videos, and use Visual Communication for the purpose. Platforms like Facebook, Instagram, and YouTube allow users to broadcast live video streams, enabling real-time communication and interaction with their followers. Using a One-to-Many communication model.

Using platforms like WhatsApp, WeChat, and Viber users are enabled to communicate with each other by sending voice messages, voice calls, and interact face-to-face via video calls.

Inspired from such Platform KWIM Messenger aims to enable user to communicate in their desired language, deliver messages without much delay, maintain the consistency, and thus also provides features like one-to-one chat, group chat, upload stories, and share it to selected people.

Chapter - 1

Introduction

1.1 Introduction

With the growing numbers of consumers of internet connectivity all across the globe. Around 97% or more of internet surfers explore the internet via mobile devices. More individuals in India are using the internet for employment, education, and leisure as a result of the COVID-19 epidemic, which has accelerated the increase of internet users in the country.

There are more than 70 languages which have more than 1 million active speakers all around the world out of which the top 10 languages are :- Hindi, English, Mandarin Chinese, Spanish, Arabic, Bengali, Portuguese, Russian, Japanese, Punjabi. Its very hard for indians alone to communicate among themselves due to te absence of common language. KWIM currently supports communication between 56 languages. It is an easy to use application in which a user has to only choose the language in which he wants to receive messages and then communicate with anyone across the globe. To enable users to send and receive messages swiftly, we set out to develop an app that would offer a seamless experience. We had to overcome a number of obstacles when designing the app, including creating a user-friendly interface, improving the program's speed, and guaranteeing the safety of information provided by users. We will go through the strategy we used to overcome these obstacles in this report, along with the fixes we put in place to build a productive hyperlocal delivery app. Along with insights into the lessons we learned along the way, we will also give an overview of the main features and functions of the app.

Overall, the process of creating the app was both resilient and gratifying. We anticipate that other companies trying to develop comparable apps to improve.

The app is simple to use, with a user-friendly interface and a range of features designed to make the communication experience as seamless as possible. Several stages of the development process were conceptualization, design, development, testing, and deployment. We encountered a number of challenges when developing the app, including enhancing its performance, guaranteeing data security, and connecting with third-party APIs and services. The positive aspect is that we were able to overcome all these challenges and develop this application through planning and efficient implementation.

The technology we are using to implement this application is Flutter. Google developed an open-source UI software development kit known as Flutter. It is used to create cross-platform software from a single codebase for Google Fuchsia, Linux, Windows, Mac OS, web, Android, and iOS devices. Flutter became available in May 2017 after being initially mentioned in 2015. The Dart programming language is utilized for developing Flutter apps, which makes use of many of its more sophisticated capabilities. Release versions of Flutter applications employ ahead-of-time (AOT) compilation across all platforms for greater performance, with the exception of the Web, where code is transpired to JavaScript. Because of this, it is the perfect option for web developers who wish to switch to building mobile apps without having to learn another programming language or framework. Hot reloading, which enables developers to view their modifications in real-time without needing to relaunch the programme, and a comprehensive set of built-in components which may be readily customised to match the particular demands of the application are just a couple of the helpful features offered by Flutter.

Flutter’s architecture comprises three layers that collaborate to power the framework.

- Framework Layer - built on Dart Language, is the component of Flutter that is most noticeable, and it gives developers access to a

broad variety of frameworks for creating and designing user interfaces. This layer can be further classified down into layers for rendering, widgets, and basic classes.

- Engine Layer - built on C/C++, provides a low-level implementation of the Flutter framework that incorporates essential APIs for graphics, accessibility, text layout, and plugin architecture. The code is executed and the user interface is rendered through communication with the framework layer.
- Embedder Layer - designed for a particular platform and is responsible for running the Flutter engine while offering it the resources it requires for developing the user experience. By establishing a platform-specific interface for the engine layer to communicate with the native platform APIs, it makes it possible for Flutter applications to function effectively on a variety of operating systems.

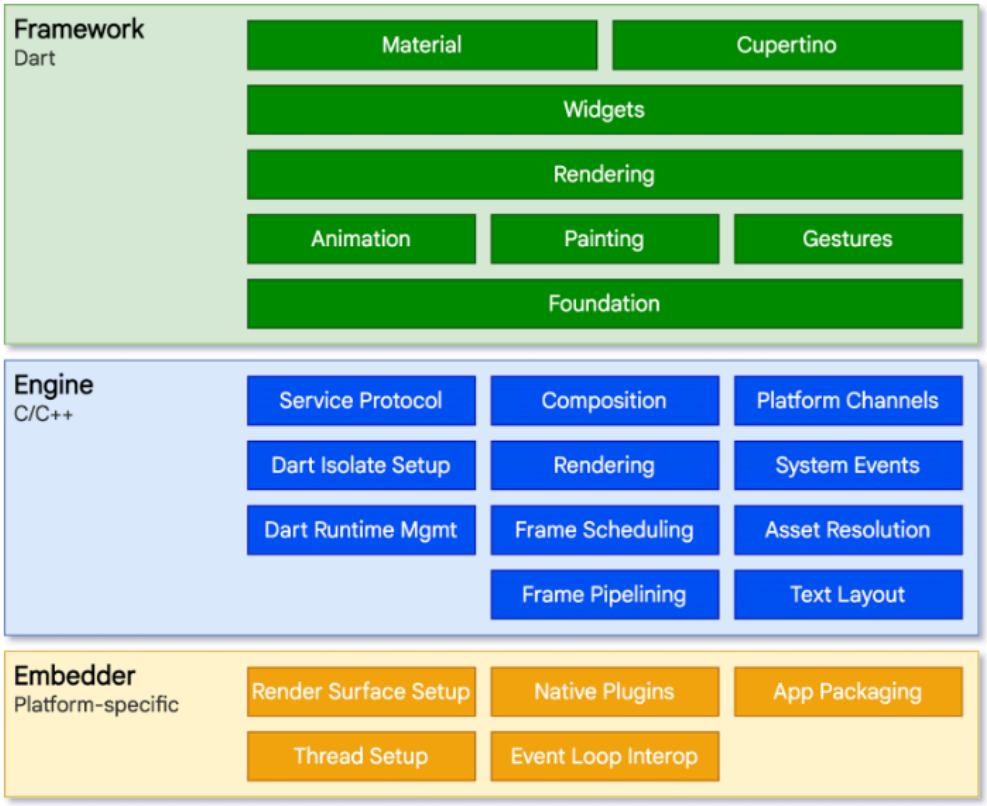


Fig. 1.1. Shows the architecture upon which flutter is built upon

Due to its simplicity, adaptability, and ability to rapidly develop and refresh applications, Flutter has grown to be an increasingly popular option for developing mobile apps. It has been used to create top-notch mobile applications by businesses like Google Ads, Alibaba, Kotak, Tencent and GooglePay. Developing software that is adaptable to several different kinds of hardware is known as cross-platform software creation. Microsoft Windows, Linux, macOS, or any combination of these operating systems may all be utilised by a cross-platform application. A cross-platform application is one that works exactly the same on any type of device, such as an internet browser or Adobe Flash. List of Websites that are built using Flutter (https://trends.builtwith.com/websitelist/Flutter)

1.2 Problem Statement

In today's world where war seems inevitable, Thinking about famous theoretical physicist Albert Einstein's word has inspired this work "Peace cannot be kept by force; it can only be achieved by understanding". Overcoming the complexity of communication between those who find themselves unable to understand different languages like me and many others. The biggest obstacle to communication between people who speak different languages makes it challenging to comprehend and communicate effectively means to which can result in misunderstandings, misinterpretations, and even conflict. linguistic obstacles can make it difficult to communicate thoughts, feelings, and intentions as well as hinder the accessibility of services and information. This challenge is especially significant in multicultural and multilingual environments in which individuals are likely to use a number of different languages and find it challenging to communicate effectively with one another. For anyone hoping to solve this issue, translation technologies, cultural sensitivity, and efficient communication techniques that put an emphasis on clarity, feelings for others, and dignity must be utilized.

People-to-people interaction is also important for improving international relations in a number of ways. It facilitates relationships, trade, and business while also helping to establish confidence and advance diplomacy.

1.3 Objectives

The main objectives of the KWIM Messenger chat application are :-

- Ease of Use: The application must be easy to use and navigate, with an intuitive interface that allows users to quickly access the features they need. It should be user-friendly and designed to cater to the needs of different types of users, including those with limited technical expertise.
- Security: The application must ensure the security and privacy of user data, including text messages, multimedia files, and other content. It should use encryption technology to protect user data from unauthorized access or interception by third parties.
- Reliability: The application must be reliable, with a high level of uptime and minimal downtime. It should be designed to minimize the occurrence of errors or glitches that can disrupt communication and negatively impact the user experience.
- Compatibility: The application must be compatible with different types of devices and operating systems, including smartphones, tablets, and computers. It should be designed to work seamlessly across different platforms, without compromising on the user experience.
- Integration with other services: To improve functionality and user experience, several chat programs connect with other services or platforms. For exchanging material or carrying out particular actions within the chat interface, integration with social networking platforms as well email services, or third-party applications are a few examples.

1.4 Methodology

In this section we will be discussing the methodology used in the project step by step. After selecting the framework and the platform, the first step toward building our project is to gather all the information and requirements for this project. In this step we identify user demands, determining necessary features, and establishing the project's scope are all part of this process. Then the next step is design and wireframing. After understanding the requirement of the application, objectives and other aspects and putting it together we work on its UI/UX designs. This step includes conceptualising and prototyping the application. This involves establishing how the user flows, developing the interface's functionality and the user experience, and producing an aesthetic that is compatible with the application and the requirements of the intended audience. In order to achieve this we have to follow certain rules:

1. **Identify target users** : The key to developing an effective application is recognising those who are your target users. To understand your users' wants, objectives, and pain spots, perform user study and collect feedback.

2. **Keeping it simple** : The layout must be uncomplicated, clear, and straightforward to use. We should avoid using complicated navigation from one component to another and cluttering our interfaces.

3. **Consistency** : In layout, uniformity is very crucial. In order to provide a uniform and consistent user experience we should use the exact same layout, fonts, colours, and icons throughout the entire application.

4. **Use the right Theme Colours** : The choice of shades is crucial in layout. Use colouring that is suitable for your brand and the intended use of the product. To make the user interface easy to comprehend and navigate, refrain from using excessive colours and make sure there is adequate contrast.

Flutter	Front-end framework
Express.js	API
Amazon CIAM services	Back-end authentication
MQTT Client	Create secure Sockets
sqflite	Local Database

Table 1.1 - Technologies stack for implementation of the application.

After creating the UI of our application and understanding the needs of the users our next step is to stack all the technology requirements. The choice of the technological stacking comes next. The main technology employed to create the mobile application for Android and iOS targeted users is Flutter, Node.js for API connectivity, and Amazon service for backend services, Firebase messaging, and many other are some more technologies that might be leveraged. Google's cloud-based Firebase platform offers an array of resources and applications for creating and deploying apps for the web and mobile devices. A wide range of backend capabilities provided by Firebase enable programmers to create apps with greater efficiency and speed. There are numerous services provided by Google’s firebase like: Authentication, Cloud messaging Programmers may 7 deliver alerts to users over many platforms, such as iOS and Android, and the web, using real-time cloud communication. After choosing the technologies we moved to the most important step of developing the code. The application's code is written during the development stage.

This involves building the application's front- end and back-end, integrating APIs, and testing it for faults and other problems. The application is tested throughout the testing and quality management stage to ensure it satisfies its functionality and operational criteria. Testing for units, validation of

integration, and user acceptability testing are all included in this. Following recommendations, documenting all of the efforts made, and making sure the source code is adaptable and manageable to accommodate subsequent upgrades and improvements are crucial through the whole process.

Fig 1.2 shows the graph representing the trend of Flutter in comparison with other technologies. Flutter being adaptive and flexible to use. Widgets state and their life cycle are some of the key features of Flutter which makes our development faster.

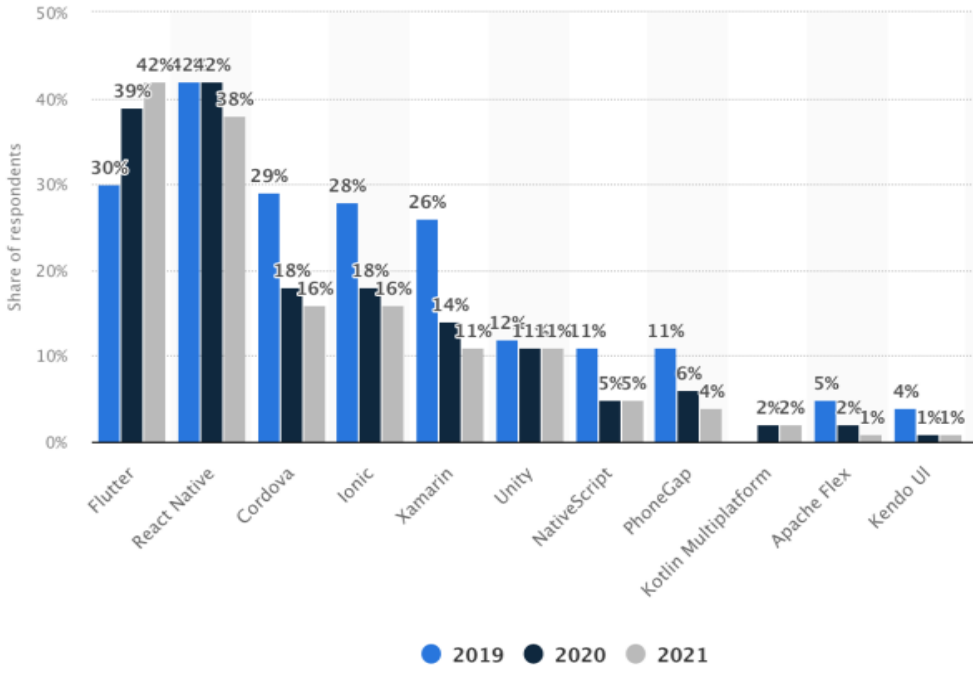


Fig. 1.2. The comparison between usage of different technologies [12]

Chapter-2
Literature Survey

Our everyday lives have grown entirely reliant on chat programs since they give us a quick and practical way to communicate. Recent years have seen a considerable breakthrough in chat application development, allowing for the development of programs that have become more trustworthy, efficient, and user-friendly. We are going to investigate the most recent innovations and trends in chat application development in this literature review.

Alam et al. (2017) developed and constructed an instant messaging app for the Android operating system as part of the research they conducted [7]. They used a client-server design, with the server running on a conventional desktop machine and the client running on the platform provided by Android. Their research centered on the use of multithreading and socket programming for efficient information processing and transport. The development and deployment of a MQTT-based instant messaging system have been recommended by Feng et al. (2018) [2]. For an efficient communication exchange, researchers used a publish-subscribe paradigm, in which the publisher publishes messages and the person who receives them subscribes to them. Their research concentrated on the MQTT protocol's usage in reliable as well as successful messaging.

An instant messaging infrastructure based on the MQTT protocol, which stands for MQ Telemetry Transport, has been proposed for design and implementation by Feng et al. (2018) [10]. For successful communication exchange, they used a publish-subscribe paradigm, in which messages are published by the sender and subscribed to by the recipient. The studies they conducted concentrated on the usage of the MQTT protocol as a means for dependable and effective messaging.

A secure chat software taking advantage of the Advanced Encryption Standard (AES) was suggested by Garg and Singh (2017) [4]. They used encrypted communication from end to end to ensure that only the recipient and the sender could view the messages during secure transmission between clients. Their research centered on developing secure chat applications using encryption methods.

For Universitas Negeri Padang, Muhaya et al. (2019) created a mobile chat application utilizing Firebase. For the transfer and synchronization of messages in real-time, they used Firebase Cloud Messaging [5]. Their research centered on using Firebase for effective and trustworthy message sharing. A messaging application based on React Native was created and built by Niu et al. in 2019. They used React Native to construct a multi-platform application, allowing it to operate on both the iOS as well as Android operating systems. Their research centered on the efficient and extensible development of applications using React Native.

For Android mobile devices, Okiro and Waweru (2018) presented an SMS-based conversation application. For the exchange of messages, they used SMS, which allowed the program to run in places with spotty or no internet access. The utilization of SMS messages for effective and reliable message exchange was the main topic of the research they conducted.

The creation and implementation issues of chatbots in a mobile app were investigated by Radev and Marinov in 2019 [8]. They used chatbots to automate the exchange of messages, allowing users to communicate with the app in their own language. Their research centered on the difficulties in designing and developing chatbots for mobile apps.

Elixir and Phoenix Framework were used for developing a mobile chat application by Ugrinovi and Vraar (2018). For efficient and extensible application development, they used the Phoenix framework which is responsible for allowing the application to manage large amounts of message

traffic. Their research centered around building scalable and effective applications using Elixir and the Phoenix Framework.

A web-based real-time chat interface server was designed and implemented, according to a study by Henriyan et al. (2016) [3]. To enable real-time communication between users, the authors stress the necessity for effective and scalable chat systems. They go through the chat server's architecture and individual parts, such as the server program, database, and web-based interface for users. The study emphasizes the application of the following: PHP, MySQL, & AJAX technologies for ensuring smooth data synchronization and transfer between clients and the server that handles the data.

Using the MQTT protocol, Hwang et al. (2016) report their study on the development and implementation of a dependable message transmission mechanism for Internet of Things applications [1]. The authors stress the need for dependable and effective delivery of messages in Internet of Things (IoT) contexts, where many devices and sensors collect and share data. They talk about the MQTT brokers, publishers, along with subscribers that make up the system's architecture. In order to guarantee accurate and timely message delivery, the article emphasizes the usage of Quality of Service (also known as QoS) levels made available by the MQTT protocol.

S. No.	Authors	Published By	Methodology	Advantages
1.	Hwang, Hyun Cheon, JiSu Park, and Jin Gon Shon. (2016)	IEEE Wireless Personal Communications	Designed and implemented a reliable message transmission system based on MQTT protocol in IoT.	Efficient Communication, Reliability and Quality of Service, Scalability, Low Bandwidth and Power Consumption, Wide Protocol Support and Interoperability, Real-time and Asynchronous Communication between devices connected via internet.
2.	Tang, Konglong, Yong Wang, Hao Liu, Yanxiu Sheng, Xi Wang, and Zhiqiang Wei. (2013)	International Conference on Information Science and Computer Applications (ICSA)	Designed and implemented a push notification system based on the MQTT protocol.	Granular Topic-Based Filtering which facilitates the delivery of intended notifications to the right recipients depending on their subscriptions through the push notification system.
3.	Henriyan, Diotra, Devie Pratama Subiyanti, Rizki Fauzian, Dian Anggraini,	International Conference on System	Design and implementation of web	The usefulness of a web-based chat interface server is that it may be improved by integrating it with other platforms or systems.

S. No.	Authors	Published By	Methodology	Advantages
	M. Vicky Ghani Aziz, and Ary Setijadi Prihatmanto. (2016)	Engineering and Technology (ICSET)	based real time chat interfacing server	
4.	Garg, A., & Singh, R. (2017)	International Journal of Innovative Research in Computer and Communication Engineering	Secure Chat Application Using Advanced Encryption Standard	A crucial component of a chat app is security. The chat application can encourage user confidence as well as trust in the safeguarding of their sensitive information by implementing AES encryption.
5.	M. F. Muhaya, A. Fauzi, and M. A. Kasim (2019)	Journal of Physics	Development of Mobile Chat Application Using Firebase for Universitas Negeri Padang	Scalable infrastructure is available on the cloud-based platform known as Firebase. By utilizing Firebase to create a mobile chat application, one can be certain that performance won't be affected as a lot of clients and message traffic increase.

S. No.	Authors	Published By	Methodology	Advantages
6.	Ali, Ammar H., and Ali M. Sagheer (2017)	International Journal of Computer Network and Information Security	Designed an android application for secure chatting.	Their research centered on the use of multithreading and socket programming for efficient information processing and transport.
7.	D. Radev and M. Marinov (2019)	International Journal of Open Information Technologies	Chatbots in a Mobile App: Design and Development Challenges	Enhanced User Experience and 24/7 Availability
8.	Boyd, Bryan, Joel Gauci, Michael P. Robertson, Nguyen Van Duy, Rahul Gupta, Vasfi Gucer, and Vladimir Kislicins (2014)	IBM Redbooks	Real-time Mobile Solutions with MQTT and IBM MessageSight.	High message throughput and support for several connected devices are features of IBM MessageSight as it offers scalable messaging features that enable real-time mobile applications to handle an expanding user base and rising message volume.

Table 2.1 - Earlier methods that were used to develop chat application using different technologies

Chapter - 3
System Development

3.1 Analysis

In this stage we gather and analyze, recognise, collect and evaluate the specifications and functionality for the application. Finding the application's key components, such as registration of new users in our application, signing-in the registered user, making them choose the desired language they want to communicate in allow them to communicate with any known person existing in their contacts. Show the message status to them either delivered, seen or processing. Allow them to share Different media (images, videos, audio, emoji, etc).

Based on the aforementioned specifications, we generated a system design that addresses the architecture, user interface, and database structure. Create a design paper outlining the technical specifications for the entire system. The system has to be built with an understanding of key platforms and technologies, such as Flutter, Firebase, MQTT Client, Sqflite, Floor, Workmanager, other pertinent resources and structures is the main emphasis of the implementation phase. Flutter Widgets along with some Third party dependencies. Amazon CIAM services for authentication services like delivering OTPs, Express.js for API services. Entire Data related to users messages is stored using Room Database after encrypting. Programming languages like JavaScript or TypeScript are used to carry out the system logic.

The Admin side database is created using popular backend-as-a-service (BaaS) platform Firebase. For mobile devices and online applications, Firebase supports a wide range of capabilities, which includes cloud-based storage, real-time data bases, authentication, and more. Using Firebase in Flutter apps has several advantages:

- **Easy integration :** straightforward and user-friendly integration: Firebase offers an integration for Flutter that is easy and straightforward

to use, with pre-built frameworks and extensions that make it simple to use its services. Developers can easily get started and rapidly incorporate Firebase into their Flutter apps thanks to Firebase's robust guidelines and assistance.

- **Scalability :** Firebase is highly adaptable, which enables it to manage enormous amounts of information and traffic without experiencing any performance concerns. Because Firebase offers automatic scaling, it will automatically increase its resources to withstand the load as flow and data volumes rise, ensuring quick and dependable effectiveness for your Flutter applications.

- **Real-time updates :** Data may be instantaneously updated throughout every device in real-time thanks to Firebase's real-time database abilities. Applications requiring real-time synchronisation or cooperation between numerous users, such chat programmes or collaborative editing tools, might benefit from this.

- **Analytics and testing :** To track and improve the efficiency of Flutter applications, Firebase offers strong analytics and benchmarking capabilities. Firebase Analytics gives developers in-depth knowledge on user behaviour, enabling them to improve user experience and monitor important KPIs. Developers may test their Flutter applications on a variety of devices using the testing tools provided by Firebase Test Lab, guaranteeing that they function properly on different platforms and setups.

- **Cloud storage :** Firebase offers services for storing and retrieving massive volumes of data, including audio, video, and picture files. Given its scalability, dependability, and security, Firebase storage is a solid option for applications that need to be able to transfer and upload large amounts of data. Incorporating a variety of strong services provided by Firebase into Flutter apps. Firebase is simple and offers advantages

including flexibility, real-time updates, strong authentication, cloud storage, and extensive statistics and testing features.

Simple integration, scalability, real-time updates, sturdy authentication, cloud storage, and effective data analysis and testing applications are just a few of the benefits of utilizing Firebase in Flutter applications.

3.2 Design

As a section of the development process, this phase entails developing a technology plan for the product's distribution application. This includes the architecture, user interface, and database schema. The framework of the system design includes descriptions of the backend services, APIs (Application Programming Interfaces), and user interface elements of the system as well as the manner in which they interact. The user interface architecture follows design standard practices in order to guarantee accessibility and simplicity. The database design specifies the data model, entity relationships, and data storage methods.

Designing of the project is distributed into certain levels. These levels are :

- Authentication
- Adding users in the admin database
- Allow them to chat by creating a topic for MQTT services

Fig 3.1 represents the ER model of user authentication. If the user is new to this application he needs to sign up and if an individual is already registered then he has to simply sign in. In both the cases the user can only validate himself with the mobile number [3]. The user gets an OTP on their registered mobile number. If the user is already signed in and not visiting the application first time then he is directly redirected to DashboardPage.

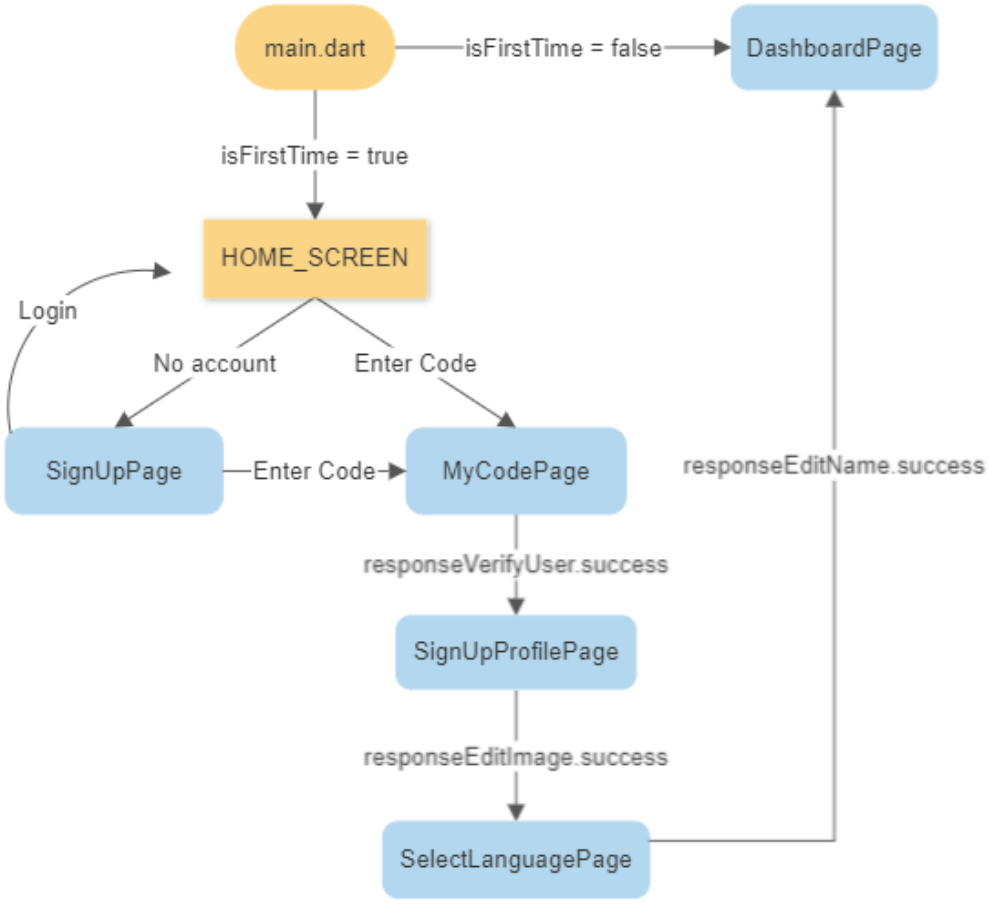


Fig. 3.1. Flow of application till user gets authenticated

Simplest way to remember whether a user is logged in or not is to store a token in local storage and each time application starts, it can be checked whether the token is null or not, if null then get him registered and store the generated token inside local storage. And to do so we used shared_preference for this, which is published by flutter developers on pub.dev, what it does is allows applications and activities to store preferences, similarly like maps in the form of ke value pairs, that persist even when user closes the application, alternatively Account Manager can be used to serve the same purpose, it stores OAuth token for all google Applications running on Android. After getting user authenticated and logged in next comes is Flow of Application. Fig 3.2 shows the flow of application from dashboard onwards.

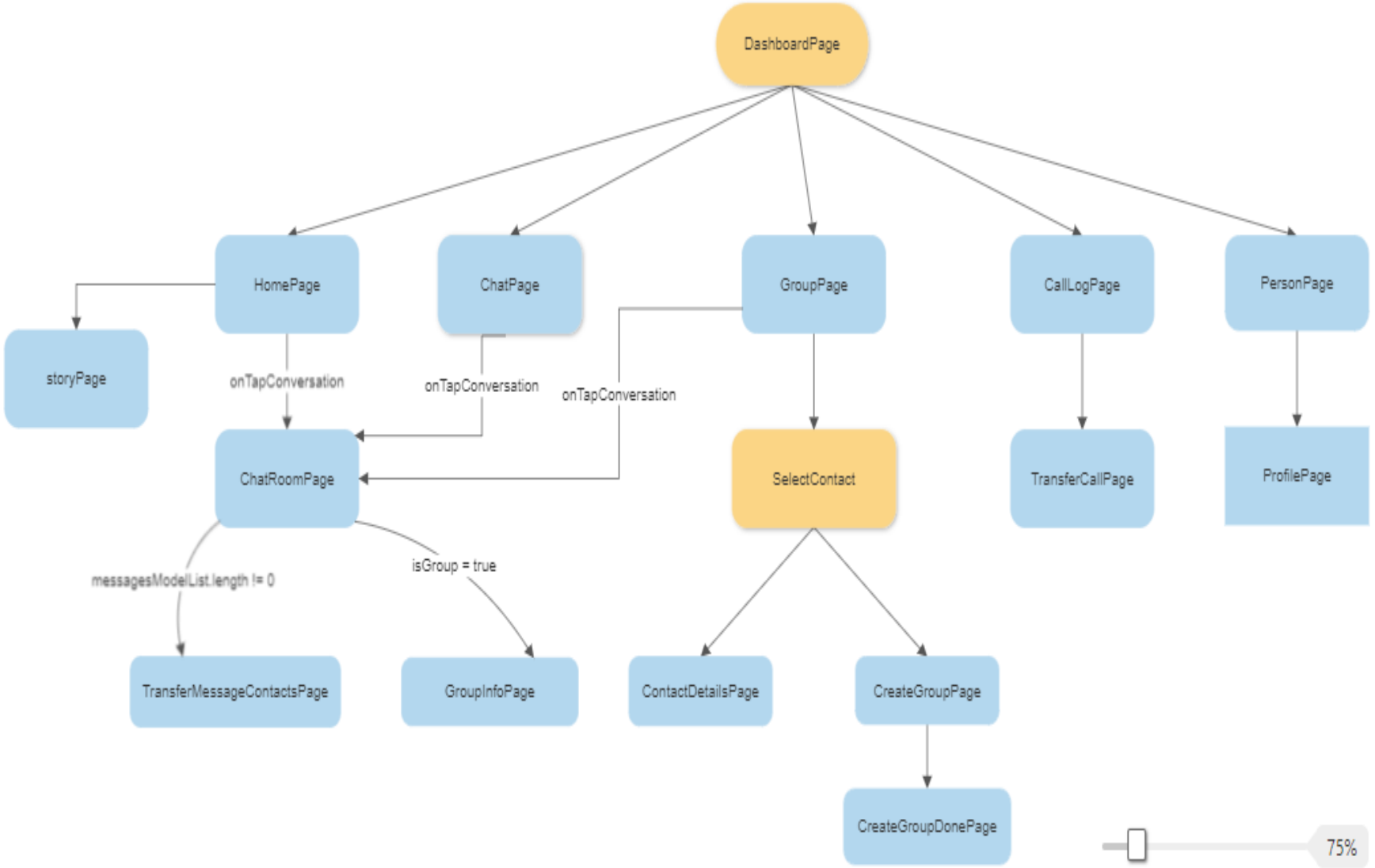


Fig. 3.2. Flow of application after user gets authenticated

3.3 Development

This section focuses on thorough discussion of the building process of our application. It also contains the tools and technologies we have used, as well as the challenges we came across may all be found during the creation portion of this report on a Flutter application. This contains the detailed information of the technologies used in this project.

- Android Studio
- Flutter
- Node.js
- Express.js
- Firebase
- MQTT Client

The usage of each technology in our project, from the initial design and development of our application to the addition of capabilities and the use of the aforementioned technologies in our application at different phases, will now be covered in a comprehensive way.

3.3.1 Android Studio

For developing applications for Different Platforms, an Integrated Development Environment (IDE) called Android Studio has been used. Additionally, this environment is used to develop Flutter, a popular framework for building multi-platform mobile apps. Numerous features and tools are available in the officially licensed Android IDE, Android Studio, which is intended to make the entire process of design and development simpler and more effective. The framework known as Flutter allows programmers to construct native mobile apps for the iOS and Android platforms by using a single source code base. With tools for creating a profile, evaluating it, and debugging, Android Studio provides a powerful development environment for creating Flutter apps. It also includes an Android device emulator so users can

test and preview campaigns on a variety of hardware and operating systems. Android Studio supports Flutter development using a plugin called "Flutter Tools". The IDE can now build and run Flutter projects as well as create, test, and troubleshoot Flutter apps thanks to the aid of this plugin. Other capabilities and functionalities added to the IDE include code emphasis and autocompletion.

Features of Android Studio for Flutter:

- This IDE offers a built-in Android emulator to test applications on multiple hardware and OS configurations.

- Developers may quickly see changes to their code without constantly recompiling the program thanks to dynamic refreshing and hot reloading.

- Developers may quickly see changes to their code thanks to dynamic refreshing without constantly recompiling the application.

- Code auto-completion and syntax-highlighting tools are offered for Flutter components and APIs. Tools for evaluating apps that evaluate their performance and point up places for improvement offer aid in developing native Android components that may be used in Flutter applications.

Tool plugin for Flutter. Through the Flutter Toolkit extension for Android Studio, which is also offered by Visual Studio Code, we may add on additional plugins, and Flutter development-specific features are introduced. This is done along with the Dart SDK which is also required.

- Flutter apps that have just been installed are commonly made accessible directly from the IDE.

- Resources for finding and fixing issues in Flutter programming.

- Programmers may execute and develop apps directly inside Android Studio thanks to the integration of the Flutter CLI.

- You may jump between Widgets specifications and the finished code for Flutter widgets and APIs.

- A welcoming setting for Android developers to work in an effective IDE with many tools and resources for creating apps. several cell phones with different Android and iOS versions that may be used to test and simulate apps.

- Assistance with Git and other effective version control methods for integrating native Android modules into Flutter apps.

Android Studio provides a powerful and efficient IDE for developing Android applications in addition to providing substantial help for Flutter growth and development. The IDE offers a variety of tools and capabilities that may expedite the development process and help programmers create high-quality, cross-platform mobile applications. Because of its robust capabilities and ease of use, Android Studio is a preferred choice among programmers for developing Flutter apps.

3.3.2 States and Widgets

Anything stored in the app's memory while it is operating may be very simply described as the state of the app. This contains all widgets, such as buttons, text fonts, icons, animations, etc., which facilitate the user interface (UI) of the programme. Everything is a widget in flutter, Widgets in Flutter can be majorly classified into stateless and stateful widget.

- Stateless widget - Stateless widgets are those whose state cannot be altered after construction. And to do so the previous when is to be destroyed and new one is to put in its place inside widget tree.

- Stateful widget - Stateful Widgets are those that can have their state changed even after construction. These states are variable and subject to various changes during the course of their existence. But they are expensive to manage than stateless widget.
  Stateful widgets are nothing but a class with a constructor and some member functions that can be overridden. These include createState(), initState(), didChangeDependencies(), build(), didUpdateWidget(), setState(), deactivate() and dispose(), with each having its own relevance.

- Inherited Widgets - there is a third class of widgets too.

3.3.3 Node.js

A JavaScript runtime called Node.js was constructed on top of the V8 JavaScript engine. It's perfect for creating server-side apps since it lets programmers run JavaScript code externally from a web browser. Node.js is recognized for its capacity to manage several concurrent connections without creating delays for other apps because of its event-driven, autonomous I/O architecture. As a result, it is incredibly adaptable and great for creating real-time applications. Node.js can run on low-cost hardware and handle a lot of data without suffering any type of lag since it is designed to be lightweight as well as speedy. Numerous library components and modules are created and maintained by a sizable and passionate programming community and may be rapidly integrated to Node.js applications to provide new features and functions.

Both relational and non-relational databases are supported by Node.js, making it simple to choose the optimal database for your application depending on

your individual requirements and specifications. With the help of the integrated package manager npm (Node Package Manager), developers can easily install, manage, and update independent libraries and modules. Because Node.js is a publicly accessible application, anybody may examine, modify, and contribute to its codebase. Developers can easily use and comprehend the platform because to its thorough documentation, and those who are already familiar with JavaScript will find it quite easy to pick up.

Node.js is utilised in this project to develop the whole back-end of the project. Node.js has the following advantages over other technologies:

- **JavaScript Runtime:** Using the Node framework, programmers may run JavaScript-based programmes without the need for a web browser. JavaScript may therefore be used by programmers to build server-side apps, command-line tools, and other types of software.
- **Node.js's Event-driven Architecture**, autonomous I/O strategy allows it to handle several concurrent connections without slowing other requests. Because of this, Node.js is incredibly versatile and perfect for building real-time applications, including chat, gaming, and other types of apps that require data to be handled right away.
- **Multiple Database Support**: Node.js supports both relational and non-relational databases. This makes it straightforward to choose the best database for your application based on your particular needs and requirements.
- **Easy to Understand:** Node.js is relatively easy to understand and implement for programmers who are already familiar with JavaScript. This suggests that programmers won't need to learn a completely new programming language or environment in order to start creating server-side apps utilising Node.js right away.
- **The V8 JavaScript engine**, which has been heavily optimised for performance and is small, was built upon when Node.js was created. As a result, Node.js can work rapidly and use fewer system resources than competing server-side platforms. Another aspect of Node.js that

helps programmers to write code that executes quickly is asynchronous programming.

- **Wide-ranging Modules**: A sizable and dedicated community of collaborators to Node.js has produced and is maintaining a wide variety of modules and packages. These extensions may easily be added to Node.js apps to provide further functionality and features.

Node.js enables javascript asynchronous programming and to do so it is crucially dependent on event loop, which is an ongoing loop that keeps track of occurring events and runs callback int its response.

3.3.4 Express.js

Express.js is a rapid, lightweight web framework built on Node.js that comes with a variety of helpful features for building APIs and online applications. It is built on Node.js and provides a straightforward API, simplifying the process of developing online and mobile apps. Express.js's server-side features may be used to do a range of tasks, such as handling incoming requests, authorizing users, and managing errors. Middleware services can be connected in a pipeline to create complex request-response processes.

Express.js is widely used to build application programming interfaces (APIs), which allow communication between various program components. An API enables the creation of complex software platforms by providing a standard interface for software to exchange data and services. Express.js provides a simple and intuitive API that makes building APIs easy. By designing routes that correspond to the different HTTP methods, developers may construct functions that manage GET, POST, PUT, and DELETE requests. Express.js offers a number of intermediate activities, such as going through received JSON data, authorizing users, and handling failures. Express.js may also be used to create a RESTful API, an architectural structure for creating APIs that follows a set of guidelines. RESTful APIs update assets (like data objects) while providing HTTP methods-based responses in a recognized format (like

JSON). Express.js API development requires careful consideration of security and scalability issues.

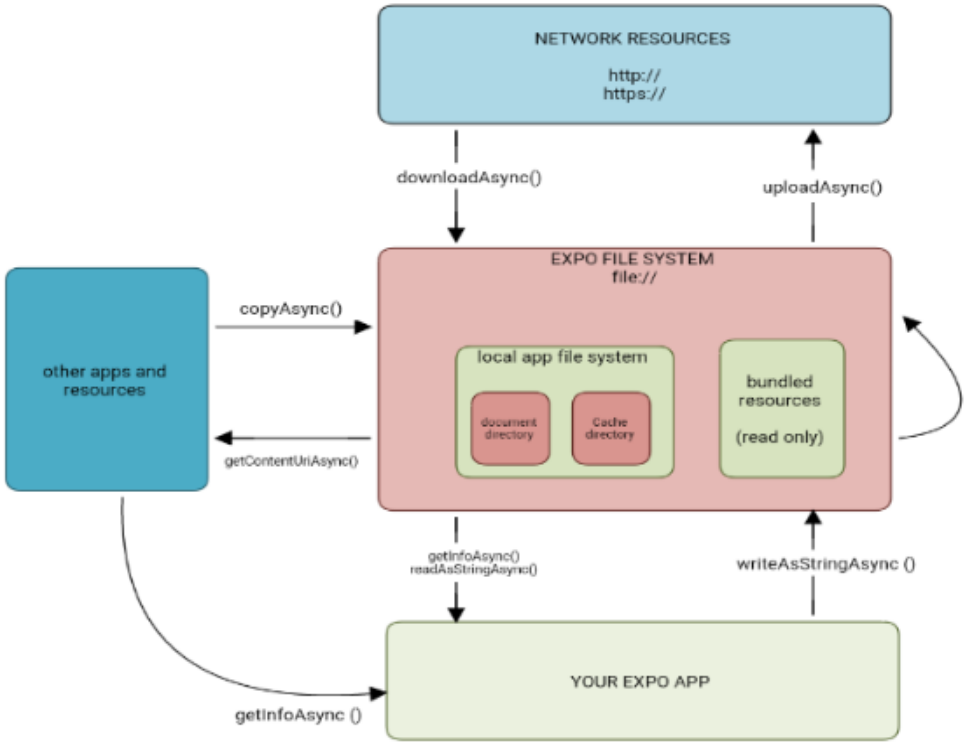


Fig. 3.3. Flow of requests and response from client to server

With Express.js's implementation of SSL/TLS encryption, data in transit may be made more secure. Rate restriction and other attack-prevention techniques are also encouraged. The Advantages of utilizing Express.js to create APIs for a Flutter applications include:

- Scalability: Express.js is incredibly scalable and has no latency while handling several connections at once. It may be installed on a network of servers or a cloud-based platform like AWS or Google Cloud.

- Security: SSL/TLS encryption is supported by Express.js, making it possible to safeguard data during the time it is being transported. It also promotes rate limiting and other attack-prevention techniques.

- A sizable community Significant and well-known developer community: A big and vibrant developer community for Express.js creates and maintains a number of modules and libraries that are easily integrated into Express.js programs to provide new capabilities. The platform also includes outstanding documentation that is easy for developers to understand and utilize.

3.3.5 Google Firebase

A platform called Firebase, developed by Google, provides a variety of tools for building and growing mobile and online apps. since of the variety of tools and services it offers, it is a popular choice for developers utilizing React Native since it may facilitate development and save time to market. The characteristics offered by the firebase in-built portion are listed in Figure 3.4

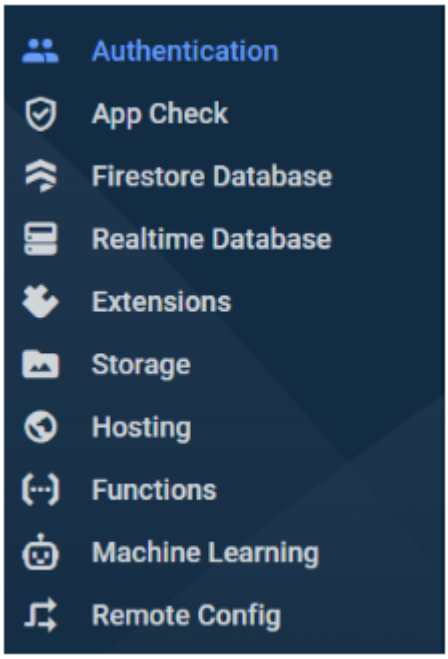


Fig. 3.4. List of firebase features

Key details regarding Firebase for Flutter applications is provided below:

- To store and sync data instantly, a possible solution is to use Firebase Realtime Database, a cloud-hosted database. It provides a NoSQL database, allowing for easy-to-manage information in a flexible, scalable way[3, 4]. The Firebase Realtime Database may be used to construct applications that respond instantly to data changes, enhancing their interactivity and user attractiveness.

- Using Google Authentication is a quick and simple way to add authentication to our application. Social media, phone, email, passwords, and other methods of authentication are supported[6]. With Firebase Identification, you can develop secure apps that request user authentication and authorization.In this project, the sole type of authentication we're focusing on is phone number verification. OTP is generated and sent to the provided cellphone number. For additional security, this OTP has a configurable life duration.

- Large-format user-generated content, such as pictures and videos, may be conveniently saved and served using Firebase cloud-based storage. It enables both local and distant storage, enabling material access from any place on the planet. Firebase Cloud Storage integrates with additional Firebase services like Oauth and Dynamic Database to provide a comprehensive storage service for your Flutter application.

- FCM (Firebase cloud messaging) is another capability offered by Google Firebase. The stable and scalable Firebase cloud-based messaging platform enables message delivery to clients on Android, iOS, and the web. It makes it simple to provide timely and pertinent communications to individuals by providing a number of features, such as targeting, preparation, and statistical analysis. As a result of its ease of use, scalability, and compatibility with other Google services, it is a well-liked choice among developers who utilize Flutter

FCM is well utilized in this application to generate notifications from admin end to all the users.

3.3.6 Workmanager

Flutter's WorkManager module for processing background tasks facilitates the execution of tasks that are left for a later time, asynchronous processes. To enable the creation of cross-platform apps, it was first launched as a component of the Android Jetpack and then incorporated into Flutter. Three different task kinds are available in WorkManager: one-off, recurring, and chained. Chained tasks are a series of activities that are carried out one after the other, whereas one-time tasks are ones that are carried out just once. Periodic tasks are those that are carried out at regular intervals.

Numerous capabilities offered by WorkManager assure the dependability and effectiveness of background work. To guarantee that activities are only run when device circumstances are ideal, developers can establish task restrictions such as connection availability or device charging status, or check whether required storage is available or not. To avoid overloading the device and to save battery life, WorkManager also provides task throttling. In order to guarantee that the tasks are rescheduled and carried out as planned, it also handles edge circumstances like device reboots and application upgrades and automatically retries unsuccessful tasks.

3.3.7 MQTT Client

IoT (Internet of Things) applications frequently employ the popular lightweight messaging protocol MQTT (Message Queuing Telemetry Transport) [1,2,9]. Different MQTT client libraries are available in Flutter that can be used to connect to MQTT brokers. For establishing connections with MQTT brokers, subscribing to topics, and publishing messages, these libraries offer simple-to-use APIs. The'mqtt_client' library is one of the well-liked

MQTT client libraries in Flutter, and offers a quick and efficient method to interact with MQTT brokers. With the help of this library, programmers quickly add MQTT capability to their Flutter apps and set up two-way communication between such apps and IoT gadgets. Flutter's MQTT client library is a useful tool for engineers like us, Since it enables us to create Internet of Things (IoT) applications that can track and interact with connected objects.

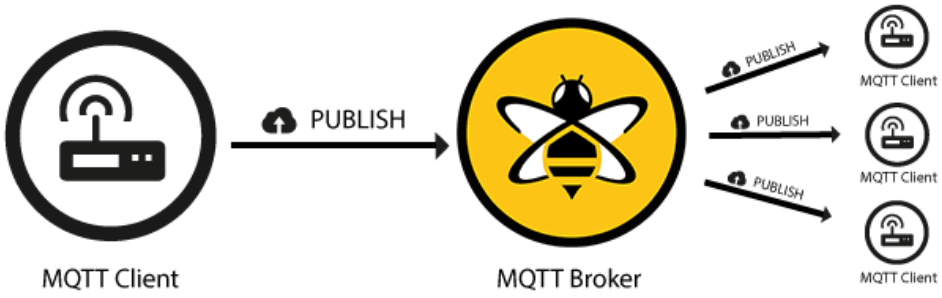


Fig. 3.5. MQTT client and MQTT broker [11]

Fig 3.5 shows how MQTT Broker notifies all the clients subscribed to a topic whenever any of the Mqtt clients publish something

The MqttClient class from the **mqtt_client** package must be used in order to connect to the MQTT broker when creating a MQTT chat application in Flutter. It is therefore required to subscribe to the chosen subjects because this will allow for easier message-exchange. The **publishMessage** function may be used to send messages, and the MqttClient class has a callback function for receiving messages that runs whenever a new message is received. When publishing messages, it's critical to take the quality of service (QoS) level into account because it affects how dependable the messages will be delivered. The chat application can also get an extra degree of protection by integrating a user authentication system.

Chapter - 4

Experiments and Result Analysis

Firstly we initialised our Flutter project inside a desired folder, There are several modules and pages in this project starting from the Welcome Screen we have and many, numerous screens or "views" that collectively make up your application's user interface are typical. Each screen serves as an individual component of your mobile application and may have various features and functionalities. Table 4.1 shows the list of screens given below.

S. No.	Screen Title	Screen Purpose
1.	Splash Screen	To introduce user to the application within 2 seconds of delay.
2.	Home Screen	If the user is not logged in then allow him to either SignIn or if user has not created a account yet then to SignUp
3.	Sign Up Page	Similar functionality as to SignIn with a bit of difference
4.	My Code Page	Prompts user to enter the OTP/code received, which on validation if was for signUp then the tokens generated are updated in the database
5.	Sign Up Profile Page	If User has SignUp he is walked through 2 more pages where he is asked to fill profile details like image, name to be displayed along with a status.
6.	Selected Language Page	After updating the profile details user is asked for desired language in which user wants to receive message in.
7.	Dashboard Page	This is the main screen where user can reach and navigate to start chatting with desired person, or update his profile, view call logs, view status of others, etc
8.	Home Page	From where user can redirect to status

		or see the chats by redirecting to Chats Room Page
9.	Chat Room Page	Depending upon the type of chat whether it is a group or not, chats are listed.
10.	Group page	Where user create a group or select a new contact to chat with.
11.	Call Log Page	Here user is redirected to different page where he can see all the call logs wich are fetched from storage
12.	Person Page	This is the screen where user can update his profile or change his language preference.

Table 4.1 - Different screens and their purpose.

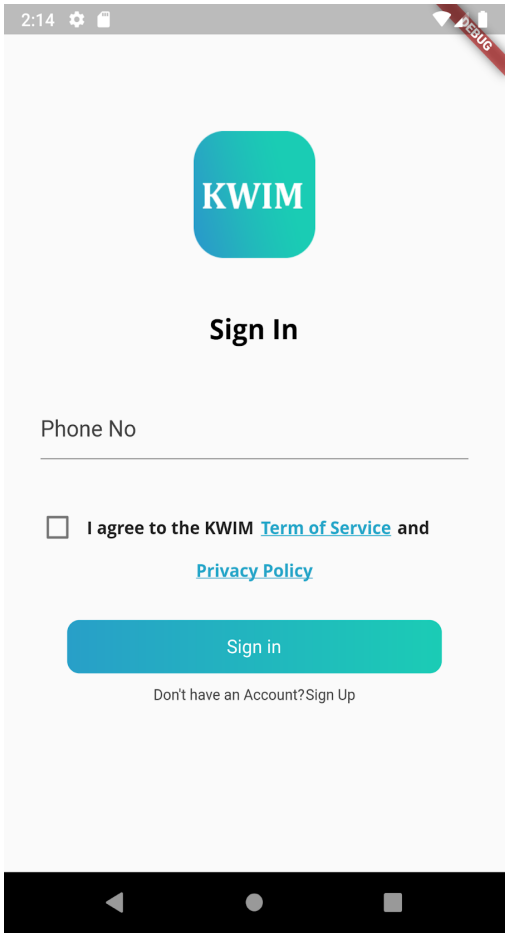


Fig. 4.1. Sign in page and route to sign up

Fig 4.1 shows how the application's Login and Registration screen will look like. Functionality of the login screen is ,it gives the user an option to register or an existing user to sign-in to their respective accounts. The Function of Register screen is, if the user is not registered (new user), he has to enter a mobile number to get himself registered. This is generating a One Time Password sending through Amazon CIAM and sending this to the user's phone to authenticate the user. The same step is being applied to both, first time registering people as well as already registered accounts.

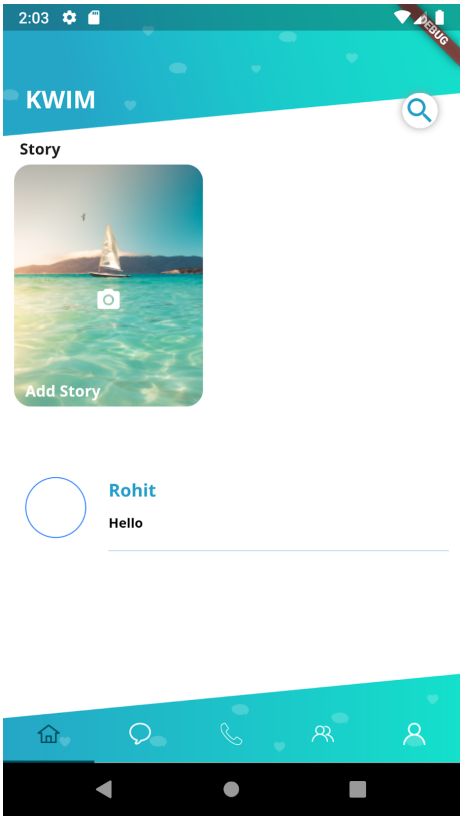


Fig. 4.2. Dashboard screen

Fig 4.2 shows The Dashboard Screen and a "TabBar View" that appears at the bottom of the screen that allows user of the application to navigate and browse through different functions, as listed in Table 4.1. This gives us an animation for a specific view. This "Tabbar View" will become visible when a user scrolls up with an animation of fadeIn. It also offers simplest way to navigate

that we have utilize to improve our application's efficiency, usability, and responsiveness.

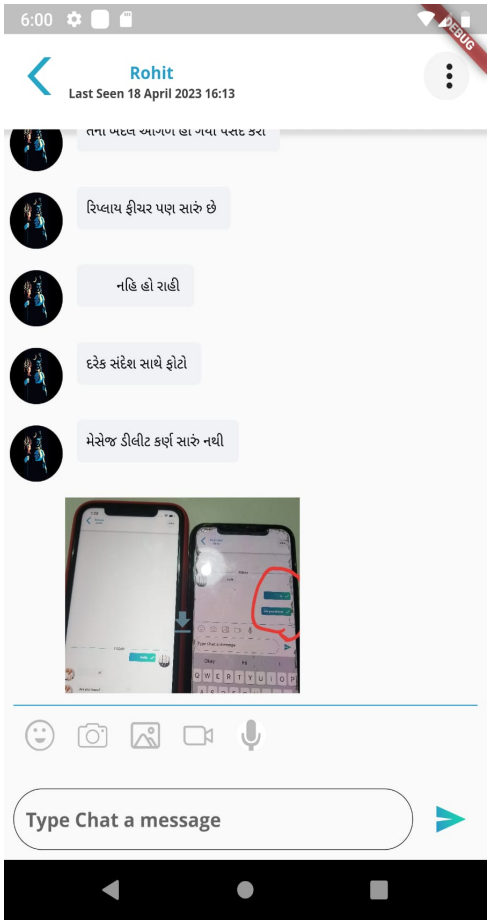


Fig. 4.3. Chats after translation

Fig 4.3 shows the text after it gets translate, and the features offered like showing the last seen, status of the message either seen, delivered or still in network somewhere, message not sent, etc.

Fig 4.4 shows the Person Page which allows user to edit the information regarding him like photo, name, status, choose different language, etc. And also ensures smooth logout

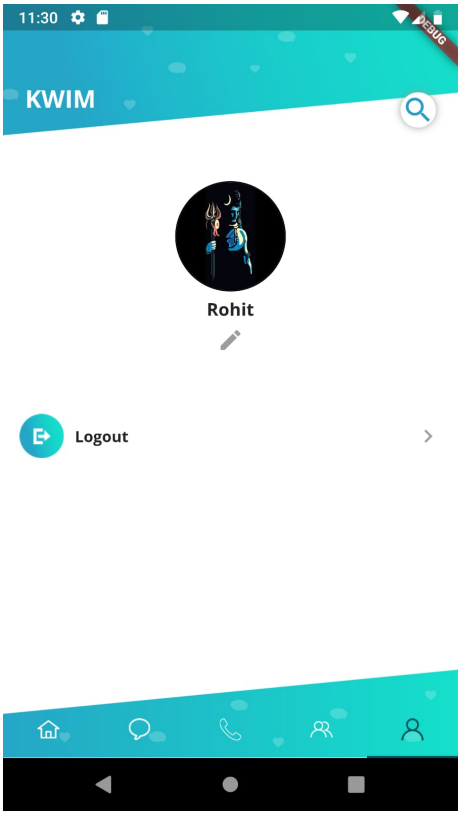


Fig. 4.4. Edit profile page

Figure 4.5 represents the file structure of te report, there we followed MVC architecture. The API folder consists of all the api calls that have been used, we used bloc state manager for managing different states like language, country, user details, etc. Then the Controller folder consists of all the related to controller part of architecture like establishing a means to communicate between models and views. Then the entire database which was stored using Room database is contained inside the database directory file path. Then the models structure is kept classified. The UI of the screen is contained inside the pages folder. The constant, colors, theme, widgets, etc and rest of the the constants are contained in the base directory of lib.

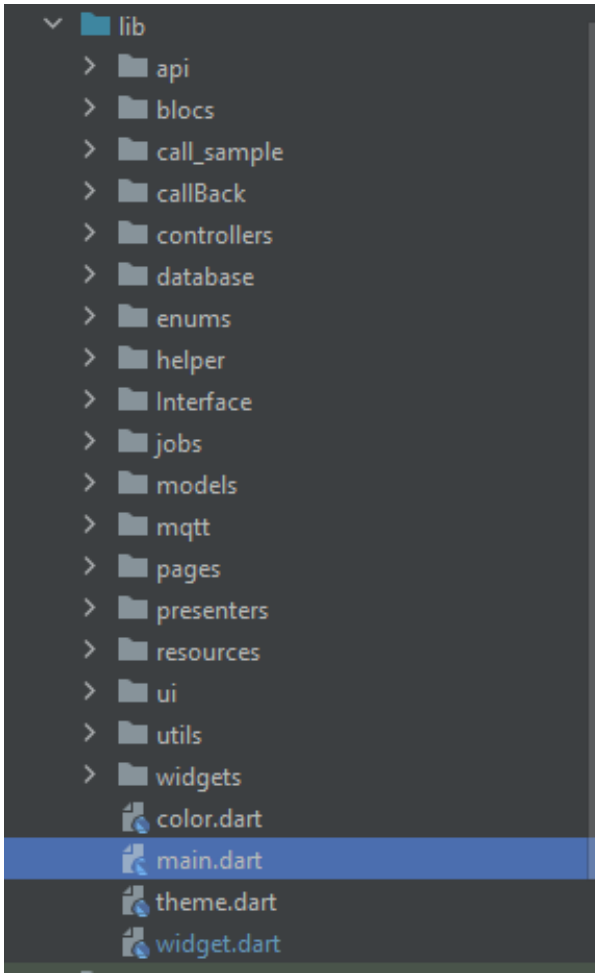


Fig. 4.5. Structure of files

```
static Future<String> getToken() async {
  final SharedPreferences prefs = await SharedPreferences.getInstance();

  return prefs.getString(PREF_TOKEN) ?? null;
}
```

Fig. 4.6. To check whether the user is logging in first time or not

Figure 4.6 shows how we check whether the user is logging in first time or not. To do so we utilized shared preferences if token exists then we return the token else we just return null.

Figure 4.7 shows the Blocs structure that we used one is the repository one which contains all the api calls repository, then there is CountryBloc and messageBloc class which implements BlocBase, messageBloc consist a method responsible to add a message.

Fig. 4.7. Bloc structure used

Figure 4.8 shows different dependencies that we used shared_preferneces, device_preview, pin_code_fields, pinput, fluttertoast, path, rxdart, mqtt_client, provider, contact_services, permission_handler, sqflite, path_provider, flt_worker, event_bus, badges, syncronized, date_format, duration, random_string, firebase_messaging, firebase_core, firebase_auth, flutter_local_notification, clipboard_manager, stop_watch_timer, record_mp3, flutter_sound, speech_to_text, flutter_webrtc, image_picker, janus_client, awesome_notifications, share, flutter_ringtone_player, giffy_dialog, android_alarm_manager, etc

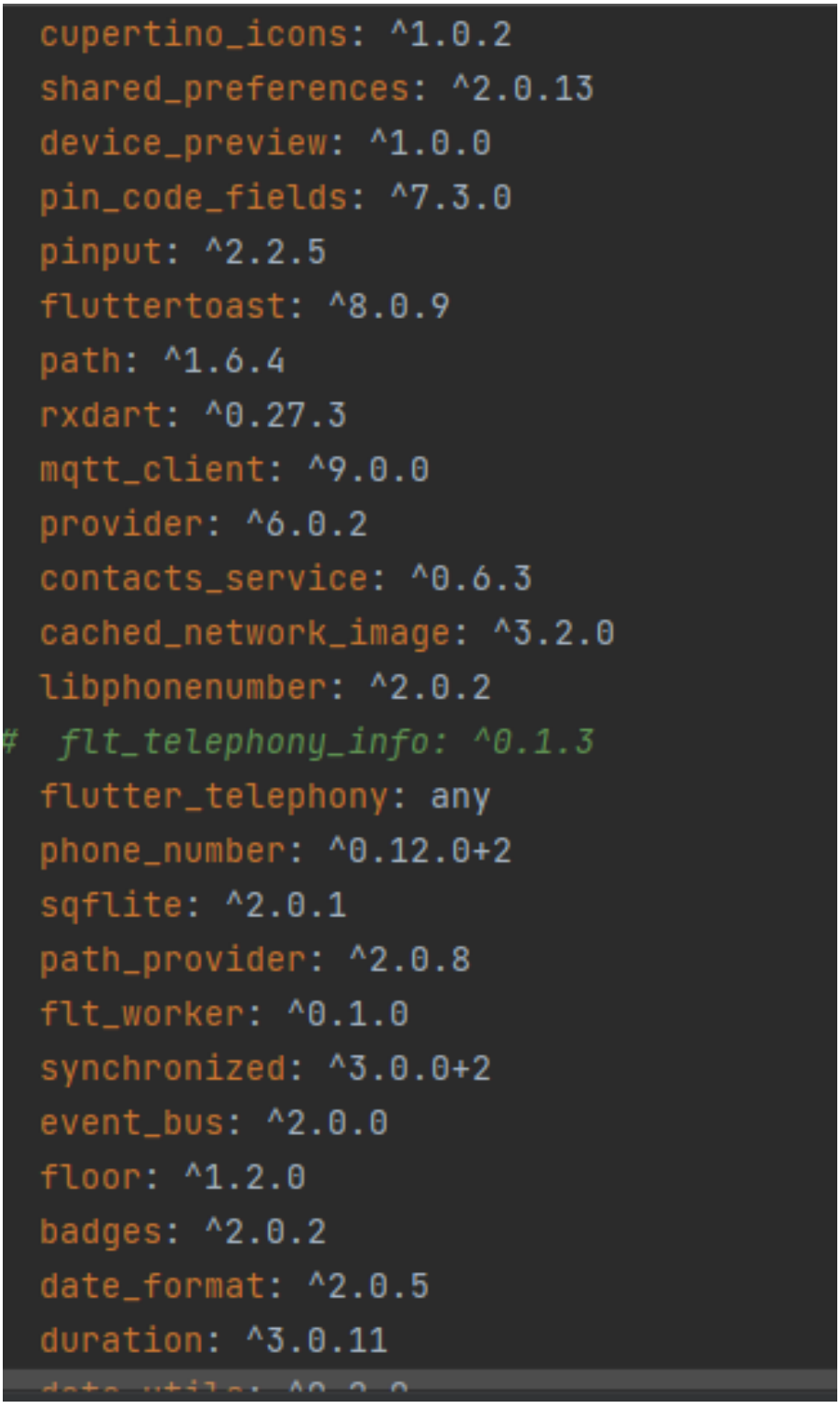


Fig. 4.8. Dependencies added inside pubspec.yaml file

Figure 4.9 shows how we redirected from page splashscreen to dasboard page, we used Navigator.pushReplacement that can move between several screens

or parts in a Flutter project by using the pushReplacement technique offered by the Flutter framework.

A stack of screens called the "navigation stack" may be managed using a set of methods provided by the Navigator class, which is a component of the Flutter framework. Users may advance and rewind through the app's displays by using the navigation stack, which keeps track of the sequence in which the app's screens are shown.

By replacing the existing screen with a fresh one, the push replacement technique is used to bring a new screen into the navigation stack. When you wish to move to another screen while deleting the previous screen from the stack, this is helpful.

```
void gotoNextPage() async {
  //////print("Go To next page is called..");
  var isFirstTime = await LocalStorageService.getToken();
  //print("isFirstTime" + isFirstTime.toString());
  if (isFirstTime != null) {
    Navigator.pushReplacement(
      context,
      MaterialPageRoute(builder: (context) => DashboardPage()),
    );
  } else {
//      Navigator.pushReplacement(
//          context,
//          MaterialPageRoute(builder: (context) => DashboardPage()));
    Navigator.pushReplacementNamed(context, HOME_SCREEN);
  }
}
```

Fig. 4.9. Navigating from splash screen to dashboard

Figure 4.10 shows the permission that are being requested from android users, to do so we carried out changes in AndroidManifest.xml inside Android/app/src/main. We added uses-permission android:name tags inside manifest tag. Some permission that we requested are READ_CONTACTS, WRITE_CONTACTS, CAMERA, BLUETOOTH, RECORD_AUDIO,

FOREGROUND_SERVICE, WAKE_LOCK, ACCESS_NETWORK_STATE, WRITE_EXTERNAL_STORAGE, GET_TASKS, etc

```
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.WRITE_CONTACTS" />
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.GET_ACCOUNTS"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.MODIFY_PHONE_STATE"
```

Fig. 4.10. Permission requested from android users

To do the same for ios devices (to request for permission that are being required by application) we need to carry out changes both in Info.plist file as well as Podfile. Figure 4.11 shows changes that are required to be carried out in ios/Runner/Info.plist and Figure 4.12 shows changes that are required to be carried out in ios/Podfile. In info.plist file we have to add different key value pairs like this =>

<key>NSSpeechRecognitionUsageDescription</key>

<string>This app will use your Speech Recognition</string>

Similarly we have to do some changes in Podfile to uncomment the lines after the description one and change its value to 1. While doing this one has to be careful with the indentation too.

dart: PermissionGroup.camera

'PERMISSION_CAMERA=1',

```
<key>LSRequiresIPhoneOS</key>
<true/>
<key>NSAppleMusicUsageDescription</key>
<string>Play using while you are hiking</string>
<key>NSCameraUsageDescription</key>
<string>Need camera access for uploading images</string>
<key>NSContactsUsageDescription</key>
<string>Required to show list of contacts</string>
```

Fig. 4.11. Request for permissions in the info.plist file

```
config.build_settings['GCC_PREPROCESSOR_DEFINITIONS'] ||= [
  '$(inherited)',

  ## dart: PermissionGroup.calendar
  # 'PERMISSION_EVENTS=1',

  ## dart: PermissionGroup.reminders
  # 'PERMISSION_REMINDERS=1',

  ## dart: PermissionGroup.contacts
   'PERMISSION_CONTACTS=1',

  ## dart: PermissionGroup.camera
   'PERMISSION_CAMERA=1',

  ## dart: PermissionGroup.microphone
  # 'PERMISSION_MICROPHONE=1',

  ## dart: PermissionGroup.speech
  # 'PERMISSION_SPEECH_RECOGNIZER=1',
```

Fig. 4.12. Changes that have to be done in podfile

Chapter - 5
Conclusion

5.1 Conclusion

The best degree of full-stack application development involves addressing several issues that aren't noticeable in applications that are identical to this one. As it would be offering one to one chat, a group chat option, Uploading status, and many other services offered by current top programs.

The exact point of differentiation is chats get translated into the desired language chosen by the user, for an example lets say someone types "**Hello, my name is Mikhail Gorbachev**" he would send it like "**привет, меня зовут михаил горбачев**" in his language. Now for a person who like the way of thinking of this russian man and wants to interact more with him, he can communicate to him in his own language "**હેલો, મારું નામ મિખાઇલ ગોર્બાચેવ છે**" this the message that a Gujrati men talking to him will appear. And Now this can be either one to one communication or a group chat. No more language Barrier while communicating.

This application was integrated with the backend and we are able to get and fetch the responses from there, every response, rejection and error was handled in such a way that it didn't slow down the application. In this report, we've addressed the main components and functionality of this Messenger Application.

5.2 Future Scope

This project has the potential to be scaled to become more intricate, and we can give the program more features. The following is a list of the functions that may be added to our application to improve its effectiveness and user experience:

- Currently we are working on Calling services like video call and voice call, etc. it will soon be incorporated in te project
  .
- There are some bug fixes that are to be done like recording not successful when done for more than 1 minute, UI layouts of chats to be converted to be more user friendly.

- More languages can be supported in future.

- A feature of reply on messages is also being incorporated.

- More login options, currently users of this application can only sign-in using a phone number, but in future we will integrate email login with phone number, so that the user can have more options to sign-in. Only if thats feasible with the current application.

- Apart from this optimisation can always be done. To improve the resource utilization of users device.

REFERENCES

[1] Hwang, Hyun Cheon, JiSu Park, and Jin Gon Shon. "Design and implementation of a reliable message transmission system based on MQTT protocol in IoT." Wireless Personal Communications 91 (2016): 1765-1777.

[2] Tang, Konglong, Yong Wang, Hao Liu, Yanxiu Sheng, Xi Wang, and Zhiqiang Wei. "Design and implementation of push notification system based on the MQTT protocol." In 2013 International Conference on Information Science and Computer Applications (ISCA 2013), pp. 116-119. Atlantis Press, 2013.

[3] Henriyan, Diotra, Devie Pratama Subiyanti, Rizki Fauzian, Dian Anggraini, M. Vicky Ghani Aziz, and Ary Setijadi Prihatmanto. "Design and implementation of web based real time chat interfacing server." In 2016 6th International Conference on System Engineering and Technology (ICSET), pp. 83-87. IEEE, 2016.

[4] Garg, A., & Singh, R. “Secure Chat Application Using Advanced Encryption Standard.” In 2017 International Journal of Innovative Research in Computer and Communication Engineering, 5(7), 3279-3287.

[5] M. F. Muhaya, A. Fauzi, and M. A. Kasim, "Development of Mobile Chat Application Using Firebase for Universitas Negeri Padang," Journal of Physics: Conference Series, vol. 1317, no. 1, p. 012033, 2019.

[6] Martinez, Emilio Rodriguez. React: Cross-platform Application Development with React Native: Build 4 Real-world Apps with React Native. Packt Publishing Ltd, 2018.

[7] Ali, Ammar H., and Ali M. Sagheer. "Design of an android application for secure chatting." International Journal of Computer Network and Information Security 9, no. 2 (2017): 29.

[8] D. Radev and M. Marinov, "Chatbots in a Mobile App: Design and Development Challenges," International Journal of Open Information Technologies, vol. 7, no. 11, pp. 42-47, 2019.

[9] Boyd, Bryan, Joel Gauci, Michael P. Robertson, Nguyen Van Duy, Rahul Gupta, Vasfi Gucer, and Vladimir Kislicins. *Building Real-time Mobile Solutions with MQTT and IBM MessageSight*. IBM Redbooks, 2014.

[10] Y. Feng, X. Li, Z. Li, and Y. Chen, "Design and implementation of instant messaging system based on MQTT protocol," Journal of Physics: Conference Series, vol. 1019, no. 1, p. 012008, 2018.

[11] The HiveMQ Team, “MQTT Publish, Subscribe & Unsubscribe - MQTT Essentials: Part 4”, HIVEMQ, https://www.hivemq.com/blog/mqtt-essentials-part-4-mqtt-publish-subscribe-unsubscribe/ (accessed May 7, 2023).

[12] Varun Bhagat, “Best Hybrid App Development Frameworks 2023”, PixelCrayons, https://www.pixelcrayons.com/blog/hybrid-mobile-app-frameworks/ (accessed May 8, 2023).