

Real-Time Chat Application

A major project report submitted in partial fulfillment of the requirement
for the award of a degree of

Bachelor of Technology

in

Computer Science & Engineering / Information Technology

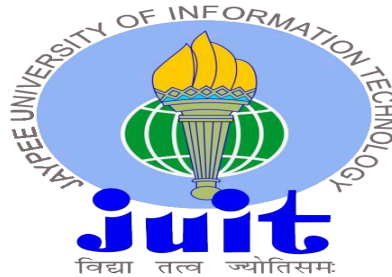
Submitted by

Rakshit (201539)

Yassharth Mani Tripathi (201502)

Under the guidance & supervision of

Mr. Praveen Modi



**Department of Computer Science & Engineering and
Information Technology**

**Jaypee University of Information Technology, Waknaghat,
Solan - 173234 (India)**

CERTIFICATE

This is to certify that the work which is being presented in the project report titled '**Real-time Chat Application**' in partial fulfillment of the requirements for the award of the degree of **B.Tech in Computer Science And Engineering / Information Technology** and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by **Rakshit (201539) and Yassharth Mani Tripathi (201502)** during the period from August 2023 to May 2024 under the supervision of **Mr. Praveen Modi** (Associate Professor (Grade II), Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat)

I also authenticate that I have carried out the above mentioned project work under the proficiency stream Cloud Computing.

Submitted by:

Rakshit (201539)

Yassharth Mani Tripathi (201502)

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

Supervised by:

Mr. Praveen Modi

Associate Professor (Grade II)

Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this report entitled '**Real Time Chat Application**' in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science & 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 August 2023 to May 2024 under the supervision of **Mr. Praveen Modi** (Assistant Professor (Grade II), Department of Computer Science & Engineering and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature with Date)

Student Name: Rakshit

Roll No.: 201539

(Student Signature with Date)

Student Name: Yassharth Mani Tripathi

Roll No.: 201502

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

(Supervisor Signature with Date)

Supervisor Name: Mr. Praveen Modi

Designation: Assistant Professor (Grade II)

Department: Computer Science & Engineering and Information Technology

Dated:

ACKNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to almighty God for his divine blessing that made it possible to complete the project work successfully.

I am really grateful and wish my profound indebtedness to Supervisor Mr. Praveen Modi Jaypee University of Information Technology, Wagnaghat deep Knowledge to carry out this project. Their 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 Mr. Praveen Modi, for their kind help in finishing 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 win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, who have developed their convenient help and facilitated my undertaking.

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

Rakshit (201539)

Yaasharth Mani Tripathi (201502)

TABLE OF CONTENT

TOPIC	Page no.
CERTIFICATE	i
CANDIDATE'S DECLARATION	ii
ACKNOWLEDGEMENT	iii
TABLE OF CONTENT	iv
LIST OF FIGURES	vi
LIST OF ABBREVIATIONS	vii
ABSTRACT	viii
CHAPTER-1 INTRODUCTION	1
1.1 Introduction	1
1.2 Problem Statement	2
1.3 Objective	3
1.4 Significance and Motivation	3
1.5 Organization of Project Report	4
CHAPTER-2 LITERATURE SURVEY	6
2.1 Overview of Relevant Literature	6
2.2 Key Gaps in the Literature	13
CHAPTER-3 SYSTEM DEVELOPMENT	15
3.1 Requirements and Analysis	15
3.2 Project Design and Architecture	25
3.3 Implementation	27
3.4 Key Challenges	47
CHAPTER-4 TESTING	49
4.1 Testing Strategy	49
4.2 Test Cases and Outcomes	51
CHAPTER-5 RESULT AND EVALUATION	53

CHAPTER-6 CONCLUSION AND FUTURE SCOPE	63
6.1 Conclusion	63
6.2 Future Scope	64
REFERENCES	65
PLAGIARISM CERTIFICATE	66

LIST OF FIGURES

Fig 3.1 User Interaction with App

Fig 3.2 App Flow Diagram

Fig 3.3 Data Serialization with Protocol Buffers

Fig 3.4 Bi-directional Communication

Fig 3.5 AWS Architecture

Fig 3.6 VPC Architecture

Fig 3.7 Load Balancer

Fig 3.8 Key Pairs

Fig 3.9 Security Groups

Fig 3.10 EC2 Instance

Fig 3.11 Route 53

Fig 3.12 Target Group

Fig 3.13 SSL Certificate

Fig 4.1 Evans CLI

Fig 5.1 Register Page

Fig 5.2 User Credentials

Fig 5.3 Login Page

Fig 5.4 Settings Page

Fig 5.5 Edit Profile Page

Fig 5.6 All Chats Window

Fig 5.7 Chat Window

Fig 5.8 Chat Window

Fig 5.9 Group Window

Fig 5.10 Preference Window

LIST OF ABBREVIATIONS

APP - Android Parsing Package

APK - Android Passing Kit

IP - I-Phone Application

API - Application Programming Interface

HTTP - Hypertext Type Protocol

SDK - Software Development Kit

UI - User Interface

UX - User Experience

IDE - Integrated Development Environment

RPC - Remote Procedure Call

ABSTRACT

The Real-Time Chat Application is a comprehensive solution aimed at addressing communication challenges by harnessing technology's power. The primary focus is on providing users with a seamless and efficient chat experience. The application ensures real-time communication, enabling users to exchange messages, multimedia content, and engage in discussions effortlessly.

Constructed in Golang, the application's backend leverages the language's efficiency and concurrency characteristics to manage several connections at once and provide a smooth chat experience. A strong and scalable architecture is ensured by the inclusion of gRPC, which makes communication between the server and clients more efficient using bidirectional streaming.

To build a visually beautiful and cross-platform user experience, Flutter is used on the frontend. Golang and Flutter work together to create a dynamic and responsive chat application that works across a range of platforms and devices.

This chat program offers a dependable and effective way to communicate, demonstrating the strength of contemporary technology. The application provides proof of Golang's capabilities, whether it is used for commercial or personal communication.

CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION

In this modern generation, generations have revolutionized various factors of our lives. An excellent breakthrough in this area is the advent of an actual-time chat application, evolved using Flutter, Golang, and gRPC. This application serves as a powerful tool that streamlines many assignments, complements communicate among colleagues, pals, and circles of relatives, and provides efficient verbal exchange.

A current Flutter-based chat software leveraging Go (Golang) and gRPC generation serves as a unified platform for streamlining interpersonal communications among diverse person agencies, along with peers, classmates, coworkers, friends, students, and parents. By consolidating these disparate parties onto a singular digital interface, this innovative solution simplifies and accelerates numerous approaches, thereby improving the productivity and performance of people worried.

The primary intention is to foster immediate facts sharing and communicate. It allow customers to easily announce important bulletins, trade messages, multimedia content material, and behavior real-time conversations. The software helps the formation of various channels, allowing customers to create committed areas for one of a kind subjects or companies, enhancing the general communication revel in.

In addition, this real-time messaging platform gives a whole lot of features tailored to fulfill the unique requirements of diverse person companies. For specialists making use of it, there are features like steady report switch, textual content trade, and media sharing for work-related purposes.

The app gives advantages past just offering real-time updates for dad and mom, instructors, and group of workers. It additionally enables easy and short communication among these groups and their partners or spouses, permitting them to be greater concerned and knowledgeable at some stage in the educational procedure. This more desirable verbal exchange can result in a deeper connection and collaboration amongst all events concerned.

1.2 PROBLEM STATEMENT

Despite the promises of many real-time chat applications, the data of their users gets leaked or sold off in the black market. To prevent data loss, the application is going to be open source means its source code is going to be freely available to all the developers out there through GitHub, so many developers can collaborate, and enhance it and its features.

The application is going to be built using new technologies Flutter, GoLang, and gRPC. Building the application using Flutter will make the chat application “**write once run everywhere**” which means Flutter helps to build cross-platform applications, the code would make the application run on iOS, Windows, Linux, and Android with only one code base.

Through gRPC, our real-time chat application is going to implement Bidirectional streaming of messages, enhancing the interactive and dynamic nature of the communication platform. This bidirectional streaming enables the continuous flow of messages and media between users, enabling updates and responses without re-refreshing the application again and again.

On the Golang backend, gRPC will efficiently handle the streaming of messages. The bidirectional nature of the communication protocol allows the server to push updates to clients as soon as new messages are available. This two-way communication ensures that users are always in sync with the latest information, promoting a truly real-time experience.

Golang (or Go) ensures a fast compilation process and efficient resource utilization means optimizing the garbage and runtime collector. It has a strong standard library that would make it possible to work on HTTP/2 technologies. Go also supports cross-compilation, allowing developers to build binaries for different operating systems and architectures from a single codebase across various environments.

The application will take effective security measures, making it open source is one while using technologies like gRPC and GoLang would make it more secure and will ensure end-to-end encryption.

1.3 OBJECTIVE

The objectives of developing a Real-time chat application are:

1. Enhance communication and the flow of information: Offering an integrated space where individuals can effortlessly communicate with one another, exchanging everyday updates and significant news, facilitating prompt and efficient dissemination of information among various groups, including families, friends, students, coworkers, and parents.
2. Enhanced User Interaction and Experience: To enrich user interaction and experience by implementing various features. These include user registration and authentication for secure access, support for multiple chat rooms to facilitate diverse conversations, and the provision of user presence and status indicators. Additionally, the application will store message history, support file sharing, and implement notifications to keep users engaged and informed.
3. Foster engagement and collaboration: Facilitating interactive features such as forming groups, sharing media, exchanging texts to encourage collaboration among friends, effective exchange of information and collaborative learning experiences between students and instructors, as well as the meaningful involvement of parents in their child's academic development, are essential for a productive educational environment.
4. Streamline Data Analysis and Management: A comprehensive data management system is essential to efficiently store, organize, and access important documents, data, images, and related materials.

1.4 SIGNIFICANCE AND MOTIVATION OF THE PROJECT WORK

The development of a real-time chat utility the usage of Flutter, Go, and gRPC holds sizable relevance and motivation in the current panorama of digital communicate. Firstly, Flutter, with its move-platform abilities, guarantees a continuing user enjoy throughout diverse

devices, fostering inclusivity and accessibility. Its reactive framework allows the introduction of visually appealing and dynamic interfaces, contributing to an attractive chat interface.

The incorporation of Go, a programming language known for its efficiency and concurrency features, enhances the application's performance. Go's lightweight nature ensures quick response times, making the chat experience swift and responsive even under heavy user loads. Additionally, Go's simplicity facilitates rapid development, aligning with the agile demands of modern software engineering.

The use of gRPC (gRPC Remote Procedure Calls), a high-performance RPC (Remote Procedure Call) framework, further elevates the application's efficiency. With gRPC, communication between the client and server is streamlined, offering advantages like bidirectional streaming and automatic code generation, reducing development time and complexity.

This chat application addresses the growing need for instantaneous and reliable communication in today's fast-paced world. Whether for personal connections or professional collaborations, the real-time nature of the application enhances user connectivity and productivity. The combination of Flutter, Go, and gRPC thus reflects a strategic and contemporary approach to developing a responsive, scalable, and efficient chat platform that aligns with the evolving demands of modern communication technologies.

1.5 ORGANIZATION OF PROJECT REPORT

The project report adheres to a structured format and consists of six chapters:

Chapter 1: Introduction - Outlines the research topic, including an overview, research objectives, significance, and scope.

Chapter 2: Literature Review - Thoroughly examines existing literature related to the research subject by analyzing previous studies, theoretical frameworks, and gap identification.

Chapter 3: System Development - Describes the research design, data collection techniques, data processing methods, analysis, and evaluation strategies used in the study.

Chapter 4: Testing - Presents and analyzes the results of conducted experiments, including detailed descriptions of experimental setups, data acquisition processes, and findings derived from various signal processing and feature engineering approaches.

Chapter 5: Results - Reports the outcome of the project work.

Chapter 6: Conclusion and Future Work - Summarizes the research findings, evaluates their relevance, highlights the study's contributions, addresses limitations, and proposes areas for future research.

CHAPTER 2: LITERATURE SURVEY

2.1 OVERVIEW OF RELEVANT LITERATURE

[1] Flutter vs React Native, Cross-Platform Mobile Application, Thesis March 2018-WenhauWu.

Both React Native and Flutter are broadly used frameworks for growing move-platform cellular packages. When selecting between them, concerns such as task necessities, developer information, and performance expectations come into play. In this assessment, we have a look at the essential capabilities and variations among React Native and Flutter that will help you make an informed selection.

React Native, developed by using Facebook, is a JavaScript-primarily based framework that permits builders to build cellular programs for each iOS and Android the usage of a unmarried codebase. By leveraging the strength of React, a JavaScript library for building user interfaces, React Native creates seamless studies across diverse gadgets. With a vast array of pre-built additives, an active network, and tremendous third-celebration libraries, React Native is a desired preference among builders.

On the other hand, Flutter, an open-source UI software improvement kit (SDK) from Google, allows designers to create attractive go-platform programs using the Dart programming language. Unlike React Native, Flutter generates its very own UI additives, ensuring consistent capability across different systems. Additionally, Flutter offers a warm-reload function that enables real-time updates, streamlining the improvement manner.

A crucial component when selecting a framework is improvement pace. Here, React Native shines, because it permits builders to reuse code throughout platforms, reducing development times. Thanks to the abundance of pre-built components and libraries, React Native simplifies speedy prototyping and development stages. However, writing personal code for UI factors can also take longer with Flutter initially. Nevertheless, Flutter's Hot Reload characteristic extensively improves productiveness, allowing instantaneous generation and debugging.

In phrases of performance, Flutter has a moderate aspect over React Native. Since Flutter renders its very own UI additives, it offers local-like performance on both iOS and Android structures. On the contrary, React Native makes use of bridging to connect with local elements, which can lead to slower overall performance. Although React Native has

advanced in this place over the years, In phrases of pics-in-depth packages, Flutter tends to perform more consistently and smoothly compared to React Native. While both structures have their strengths and weaknesses, React Native has greater widespread records and a larger developer base, supplying access to an abundance of assets, libraries, and community-generated solutions. These robust surroundings make it less complicated for builders to discover answers to various development challenges. However, regardless of its noticeably more recent repute, Flutter has seen an extensive increase and won recognition amongst builders, with a faithful fanbase and developing repository of applications and plugins available.

[2] Clean approach to Flutter Development through the Clean Flutter architecture package, IEEE 2019, Shady Boukhary, Eduardo Colemanares.

In this paper, Shady Boukhary and Eduardo Colemanares present an effective methodology for developing Flutter apps using the principles of clean architecture. By separating concerns into distinct layers, they aim to enhance maintainability, testability, and overall efficiency in large-scale projects.

To start, the authors acknowledge common issues encountered during Flutter app development, including difficulties in managing code quality and speed of development as applications grow in size and complexity. To overcome these challenges, they recommend utilizing the Flutter Clean architecture package, which divides the application into three primary sections: presentation, and data layers.

The presentation layer acts as an intermediary between users and applications, displaying information through visually appealing widgets, views, and controllers while handling user inputs. In contrast, the domain layer comprises essential components such as business logic, entities, use cases, and interfaces, which define the rules and procedures governing the application's functionality. Notably, this layer features an inverted dependency structure, allowing each layer to operate independently, making it easier to adapt the application to different platforms or frameworks without disrupting its overall operation. Finally, the data layer manages data retrieval and persistence by utilizing repositories, data sources, and models to interact with external data sources like databases, web services, or local storage, thereby streamlining complex data management tasks and enabling the domain layer to focus solely on business logic.

Boukhary and Colemanares stress the significance of dependency inversion and injection within these layers to achieve optimal coupling. They demonstrate how implementing this strategy leads to more manageable, maintainable, and extensible software systems.

In summary, the authors' work presents a practical guide for leveraging clean architecture in Flutter app development, emphasizing the benefits of modularity, testability, and maintainability achieved by structuring applications according to well-defined layers.

In essence, implementing the Flutter Clean Architecture framework allows developers to effortlessly simulate or replace dependent modules during unit testing, enabling thorough examination of individual layers within an application. This article explores various design patterns and software libraries that aid in building clean architectures for Flutter projects. These include pre-made components and resources from packages like 'flutter_clean_architecture' and 'flutter_bloc'. Moreover, the text provides practical instances and code fragments demonstrating the proper implementation of Flutter Clean Architecture. These illustrations cover organizing projects, separating responsibilities, and setting up straightforward interactions through occasions and callbacks.

Employing this methodology has several advantages. It improves code arrangement, makes maintenance simpler, and lowers the complexity associated with scaling applications. Its modular nature permits team members to concentrate on distinct parts without interfering with each other, thereby streamlining cooperation and minimizing development hurdles. Finally, clean architecture fosters testability, making it easier and more extensive to validate individual layers of an app through unit tests.

[3] End user's perception of Flutter mobile apps, Malmo University Nov 2019-Dahl, Ola.

Google's Flutter is an open-source software development kit that allows programmers to build cross-platform mobile applications using a shared coding base. Understanding how users perceive these apps is important as Flutter becomes more widely used. By analyzing user perspectives, we can gain insightful information about their experiences, preferences, and overall satisfaction with Flutter-based apps. The user interface (UI) and user experience (UX) play a significant role in shaping end-users' opinions. Flutter provides a diverse selection of personalized widgets and guarantees a slick and reactive UI across several platforms, which users generally find attractive and coherent. Moreover, Flutter's focus on delivering a local-like experience positively impacts end-user sentiment. Performance is

another key element that influences end-user perception. Flutter apps are developed employing the Dart programming language, which converts into native code, resulting in quicker load times and lower latency. Users highly prize the seamless and responsive behavior of Flutter programs, contributing to higher levels of customer contentment. In addition, the capability to design powerful applications suitable for numerous devices and operating systems improves end-user approval.

Furthermore, the accessibility of third-party modules and libraries is crucial for developers who work with Flutter. These add-ons facilitate the development process by offering pre-existing functionality. Users benefit from having easy access to a large collection of readily available packages since it speeds up the integration of new characteristics and capacities into applications. This adaptability favourably affects end users' attitudes toward Flutter apps. Last but not least, Flutter's cross-platform functionality is a considerable strength, making it possible for creators to produce apps compatible with both Android and iOS platforms via a single source codebase.

Users enjoy using applications created with Flutter since it allows them to use the software across various devices without worrying about compatibility concerns. Flutter's ability to work on several platforms positively impacts how consumers perceive it because it gives them ease of access and adaptability. Frequent upgrades and community assistance are vital elements that influence an individual's opinion of Flutter. Developers give their best efforts to ensure that the platform gets better over time by continuously improving and fixing bugs thanks to the large and dynamic developer group around Flutter. This dedication helps build trust and reliance among customers. Reviews and evaluations made by people significantly shape their impression of Flutter programs. Benefits like high performance, good UX/UI, and multi-platform capabilities draw more individuals when favorable assessments focus on these qualities. On the other hand, unfavorable critiques bring attention to locations where development may be required. It is imperative to gather and analyze customer comments to pinpoint possible problems and improve the end-user experience so that the general consumer view of Flutter applications might become more optimistic.

[4] GRPC: A Communication Cooperation Mechanism in Distributed Systems

GRPC is a communication cooperation mechanism in distributed systems. In this document, we present an innovative approach to remote procedure call (RPC) systems known as Group Remote Procedure Call (GRPC). Existing RPC models have limitations, such as unreliable

messaging, lack of transparency, and restricted scalability. To address these issues, we propose integrating the process group concept into RPC frameworks, enabling them to accommodate collective communication. We outline three distinct GRPC mechanisms—lookup, function-convergence, and update—for managing group membership, executing group functions, and updating shared data. Our proposed system exhibits improved dependability, openness, and parallelism when compared to traditional RPC designs. We describe the structure's architecture using four key elements: conventional GRPC invocation procedures; handling GRPC responses; controlling GRPC groups; and ensuring atomicity and sequence of occurrence for message distribution. Additionally, we evaluate the efficiency of our prototypes through benchmark tests comparing GRPC's functionality with that of other RPC systems. This research makes significant contributions to the area of interprocess communication by introducing novel approaches to developing efficient group communication protocols in complex sys

[5] Comparing REST, SOAP, Socket and gRPC in computation offloading of mobile applications: an energy cost analysis

This paper examines the energy cost analysis of various communication protocols and architectural styles while computing algorithms of different complexity levels and input sizes on mobile devices. The authors aim to assess the energy efficiency of distinct communication methods during computational tasks by conducting experiments that address specific research questions.

To investigate this topic, the study employed four primary communication protocols - REST, SOAP, Socket, and gRPC. These were used to execute simple sorting algorithms with diverse complexity and input dimensions. The authors then evaluated the energy expenditure associated with each communication approach by comparing the amounts of energy consumed based on the quantity of computations carried out and the volume of data transferred. To gain insight into situations wherein one technique may excel over another, mainly when appearing calculations regionally, they altered those variables.

Throughout their investigation, the authors accrued quantifiable records regarding the test's execution and supplied complete findings in Table II. Each line of the desk represents an person experimental situation, whilst columns 1-5 depict relevant quantities referring to the total quantity of statistics processed and computational work required. The final columns

display the corresponding electricity consumption figures for nearby processing thru SOAP, REST, Socket, or gRPC, respectively, representing the gadgets of size as energy.

The results of the take a look at show that gRPC is the maximum power-green protocol for remote execution, observed by means of Socket, REST, and SOAP. The look at also highlights the significance of thinking about the complexity of the set of rules and the input length when choosing the verbal exchange protocol for far off execution. The researchers conclude that the choice of communication protocol and architectural style can have a significant impact on the energy consumption of mobile devices and that developers should carefully consider these factors when designing mobile applications.

Furthermore, the investigation uncovered the optimal communication strategies for distant processing. Our findings indicate that remote execution was the least expensive option across various scenarios, including smaller data sets and linear algorithms. However, when dealing with intricate procedures and bigger datasets, computational offloading might conserve up to ten times more energy compared to performing computations locally.

By determining the most cost-effective communication approaches, we may assist software engineers in selecting appropriate technologies and maximizing energy savings through computation offloading. In forthcoming projects, we aim to repeat these tests employing diverse algorithms, input varieties, and gadgets. Additionally, we intend to apply actual apps to transfer our algorithms to assess under what circumstances offloading is practical and which communication protocols or architectures are best suited for this purpose.

[6] Comparison of Memory Usage between REST API in Javascript and Golang

This research paper compares the memory usage efficiency of RESTful APIs built using JavaScript and Go (Golang). The authors begin by emphasizing the significance of the internet and web applications in today's businesses, before delving into the experimental methodology used to evaluate the memory consumption of these APIs.

To conduct the comparison, the researchers leveraged a tool called 'pprof' to measure the memory usage of both technologies. They designed and implemented identical RESTful APIs in JavaScript and Go, and utilized a dataset from Google Cloud to test their memory usage under various loads. Specifically, they focused on the GET protocol, which retrieves data with a response status code of 200 OK.

The findings of the experiment revealed that Go is more memory-efficient than JavaScript when building RESTful APIs. The researchers located that Go consumes extensively less

reminiscence than JavaScript, particularly when dealing with large datasets. To further elucidate those observations, the article includes visible representations of the reminiscence utilization styles exhibited through the 2 technologies through tables and graphs.

The examine additionally discusses the database layout and REST API architecture used at some point of the test. The authors set up a MySQL database with described field systems for every desk, organizing them into two tables and two fields per dataset. The data stored within these tables were retrieved via the REST API, which was implemented in either JavaScript or Go. In this investigation, the HTTP method employed the GET request.

In conclusion, the study sheds light on the significance of optimizing memory usage in web applications through an examination of REST API performance with respect to JavaScript and Go (Golang).

By comparing these platforms' memory resource utilization, researchers encourage software program builders to adopt Go while constructing RESTful interfaces requiring big reminiscence use due to its validated superiority in handling reminiscence-intensive duties. Ultimately, this scholarly paper gives precious insights into the reminiscence consumption styles of REST APIs, making it a beneficial aid for experts and students inquisitive about internet application improvement.

[7] Designing website vaccine reserving machine the usage of Golang programming language and framework react JS

Golang programming language changed into used to construct the backend of the vaccine reserving machine internet site. Golang, also referred to as Go, is an open-source programming language evolved by Google. It changed into selected for its benefits in phrases of pace, reliability, scalability, and ease.

The backend of the application is liable for dealing with facts control, authentication, and different backend functionalities. It guarantees the safety and integrity of the gadget. Golang is a statically typed language that produces binary code, making it green and speedy.

In the context of the vaccine reserving device, Golang is used to handle duties together with coping with vaccine schedules, vaccine stock, and registrant records. The backend machine guarantees that the vaccination manner runs smoothly and securely.

Golang's simplicity and scalability make it appropriate for constructing strong and excessive-overall performance backend structures. It allows for concurrent programming,

which means that a couple of obligations can be accomplished simultaneously, enhancing the system's performance and responsiveness.

Overall, Golang turned into selected because the backend programming language for the vaccine booking machine website due to its pace, reliability, scalability, and simplicity. It performs an important function in dealing with and securing the records and functionalities of the software.

2.2 KEY GAPS IN THE LITERATURE

Some of the potential gaps are addressed here

1. Diversity in features

It appears that only a certain feature set was implemented. Need to implement more and more new features.

2. Consistent Performance Metrics:

The chat application requires consistent performance metrics to assess its efficiency. Benchmarks and metrics should be established to measure chat responsiveness, reliability, and user satisfaction consistently.

3. Robustness

With varying network conditions and device specifications, the app needs to function seamlessly under diverse network and device environments.

4. Adaptation across various geographical areas

The application should be designed to adapt to different geographical regions, keeping in mind language preferences and currency preferences. This would ensure user friendly and culturally sensitive chat experience.

5. Real-time processing

The app should do real-time processing of messages, that is it should ensure messages are processed and delivered in real-time to enhance user engagement.

6. Collaboration

The app would be made open-source and developers are welcome to collaborate and contribute to the project.

7. Scalability

The app should be scalable means it should scale up and down according to the traffic it experiences.

8. User-Friendly interface

A User-friendly interface would be prioritized to enhance user experience.

CHAPTER 3: SYSTEM DEVELOPMENT

3.1 REQUIREMENTS AND ANALYSIS

Flutter

Google's Flutter is an innovative software development kit (SDK) designed to streamline the creation of cross-platform applications. With its capacity to generate high-caliber native experiences on various devices through a singular coding base, Flutter has garnered substantial attention within the developer community. At the heart of Flutter lies a distinctive rendering mechanism that sets it apart from other frameworks. Rather than relying on platform-specific widgets, Flutter employs its proprietary collection of customizable widgets to craft striking and uniform graphical interfaces. This method outcomes in Flutter apps showing a local appearance and experience, thereby selling an uninterrupted give-up-user revel throughout diverse structures.

A prominent gain of Flutter is its stay preview capability, allowing developers to witness instantaneous updates to their code without the need for rebuilding. This expedites the development procedure and fosters short experimentation and refinement, making Flutter especially suitable for prototyping and successive application development. Furthermore, Flutter's remarkable overall performance attributes are excellent. Its use of an excessive-overall performance rendering engine called Skia allows seamless animation and lightning-fast rendering, consequently resulting in a snappy and tasty consumer revel. Additionally, Flutter harnesses the electricity of the photograph processing unit (GPU), decreasing computational load and maximizing performance. Notably, Flutter boasts an in-depth library of prefabricated widgets and equipment, overlaying an extensive spectrum of user interface elements, such as buttons, text fields, lists, and navigation. Moreover, the robust plugin architecture of Flutter provides admission to a significant array of third-celebration packages and extensions, extensively broadening the scope of the framework.

Flutter's multi-platform functionality enables developers to write down code that can be executed on numerous devices and running systems, thereby lowering the quantity of effort

and time required during the development technique. By adopting this approach, organizations and builders can effortlessly get entry to a wider variety of customers due to the shortage of want to hold distinct codebases for each platform. Additionally, Flutter gives an uncomplicated method of integrating with different technologies through its easy APIs, permitting builders to harness tool capabilities which include cameras, vicinity statistics, and sensors. Moreover, Flutter boasts seamless integration with Firebase, a cell development platform offered using Google, which offers entry to precious backend services together with authentication, cloud storage, and actual-time databases.

The sturdy developer guide community surrounding Flutter is another substantial contributor to its reputation. A lively and collaborative community presents ample sources, tutorials, and example tasks, making it less complicated for builders to locate solutions to their questions or a percentage of their know-how. As an end result, Flutter has installed itself as a prominent framework for building go-platform applications. Its personal interface is both expressive and efficient, while its excessive overall performance and warm reload function permit developers to speedy construct visually attractive and reactive apps. Furthermore, Flutter's potential to run on a couple of structures, a widespread series of widgets, and easy integration with external technologies have made it a proper alternative for groups and people trying to increase pinnacle-notch apps for diverse systems. Notably, Flutter remains below continuous development, with frequent updates and network involvement assisting to solidify its function as a leading tool for current software improvement.

Flutter Widgets

In Flutter app development, widgets serve as the essential components used to build user interfaces. A widget represents a seen element at the display that customers can engage with, such as buttons, textual content, pix, containers, and other format elements. Developers can make use of diverse widgets supplied via Flutter to create visually appealing and functional consumer interfaces for their applications. The core widgets in Flutter include Text, Images, Icons, Buttons, Rows, Columns, and bins, which can be mixed to create more complicated UI designs. Additionally, Flutter offers a range of widgets following particular layout guidelines, including Material Design and Cupertino widgets, which adhere to Google's Material Design principles and Apple's Human Interface Guidelines, respectively. Furthermore, Flutter gives

widgets for layout management, including Centre, Padding, SizedBox, and Expanded, to help developers set up widgets successfully on the display screen.

In the realm of software development, widgets serve as consumer interface components for inputting records into packages, including TextField, Checkbox, Radio, and Slider. These factors facilitate navigation between monitors or pages inside an app through navigation bars, drawers, and tab bars. To resource developers in constructing sturdy and maintainable apps, the Block (BLoC) library gives a comprehensive implementation of the BLoC layout pattern, complete with aid for asynchronous operations, error dealing, and navigation. Additionally, the BLoC library functions as equipment for debugging and trying out these patterns. Code technology, or using pre-made templates and standards to mechanically write code, is another essential tool inside the Flutter atmosphere. With the help of this approach, the Flutter Bloc module can create BLoC boilerplate code extra fast and correctly than might otherwise be viable. Another famous technique for dealing with the kingdom in web development known as Redux has been tailored for use in Flutter through the Flutter Redux bundle. This lets in for the advent of a single, centralized place wherein the whole thing of the utility's state can be controlled, making it less difficult to keep the music off and modify it when necessary. Reactive programming, a computer technology paradigm concerned with how packages reply to changes in their inputs over the years, is becoming more and more essential in modern software development. Libraries like RxDart offer sets of reactive programming gear that work well with the BLoC pattern, permitting green management of complex asynchronous statistics flow.

One of the maximum complex principles in Flutter app improvement is nation management, which entails tracking and modifying an application's country in response to personal interactions. As the complexity of an application grows, managing its nation can end up more and tougher. Flutter offers numerous kingdom control tactics, such as the use of `setState()`, InheritedWidgets, Providers, BLoCs, Redux, and others. Each approach has its professionals and cons regarding code complexity, overall performance, maintainability, and scalability. Choosing the right approach depends on the unique wishes of the software.

Another challenge in Flutter improvement is dealing with platform-unique incompatibilities. Even though Flutter targets to offer regular consumer enjoyment throughout all systems, the

capability and layout of widgets may also vary between them. To ensure that their packages paint properly on all platforms, developers ought to be privy to these variations.

In addition to kingdom control and platform-associated problems, other difficulties in Flutter programming include navigating via the app, implementing animations and gestures, making use of external APIs and facts resources, and operating with external records assets. These regions require deep expertise in Flutter's API and the capability to apply pleasant practices correctly.

Dart

Dart is a contemporary programming language gaining popularity because of its particular aggregate of overall performance, usability, and flexibility. Developed using Google, Dart offers an intuitive syntax and seamless integration with different programming languages, streamlining the improvement manner. The language's statically typed device ensures correct mistake detection and minimizes errors within the direction of the coding stage. Furthermore, Dart's balance between static typing and kind inference permits maximum useful productivity without compromising on code outstanding.

Dart's fantastic overall performance skills are owed to its gift-day digital device, dubbed the Dart VM, which executes code sooner or later of runtime thru really-in-time (JIT) compilation. Additionally, Dart can be compiled in advance of time (AOT) into nearby gadget code, resulting in exceedingly rapid execution speeds.

Dart's versatility extends past its suitability in several software program improvement niches. For example, it may be hired for growing cell packages with the usage of the widely followed Flutter framework, imparting an accessible substitute for move-platform app improvement. Flutter leverages Dart's reactive programming version and widget-based overall structure to craft visually attractive and responsive cell apps like-minded with each iOS and Android running system.

Moreover, Dart also can be featured as the precise alternative for decreasing again-end improvement through frameworks inclusive of Aqueduct or Angel. These frameworks capitalize on Dart's asynchronous programming talents to construct immoderate-performing net applications and representational country transfers RESTful APIs. Dart's assistance for

concurrent processing and comprehensive series of libraries and equipment render it nicely appropriate for constructing server-aspect applications.

The Dart programming language boasts a sturdy set of included capabilities and gear that cater to diverse additives of software software development. This consists of features for coping with input/output operations, network conversation, and encryption, among others. Additionally, Dart's package manage gadget, called Pub, gives builders clean admission to a fantastic repository of third-party libraries and packages, for this reason enhancing their productivity.

The language itself prioritizes code reuse and modularity, allowing builders to create reusable modules and libraries. This technique encourages maintainable and scalable coding practices even reducing unnecessary duplication of attempts.

Moreover, the Dart community actively contributes to the language's ecosystem by presenting precious assets, documentation, and network-created applications. This collaborative environment fosters teamwork, understanding sharing, and innovation amongst developers.

Overall, Dart is a versatile and effective programming language that offers a streamlined improvement revel, fantastic average performance talents, and cross-platform compatibility. Its intuitive syntax, rigorous type system, massive library offerings, and sturdy help from the developer network make it a popular desire for constructing complex and scalable programs. With frameworks like Flutter leading the manner, Dart showcases the wonderful capability for future growth and adoption throughout the tech corporation.

gRPC

gRPC, which stands for Remote Procedure Call, is as same as the super-smart messenger that helps different computer programs talk to each other efficiently. Imagine you have a bunch of friends scattered all over the city, and you need to share messages with them quickly. gRPC is like the speedster courier service that makes sure your messages reach their destinations in a flash. What makes gRPC so cool is its use of the fancy-sounding HTTP/2 protocol. This protocol is like a magic wand that allows multiple messages to be sent and received at the

same time. It's like being able to chat with multiple friends simultaneously, saving a lot of time and making the conversation more efficient.

Now, when it comes to putting messages in a format that computers can understand, gRPC uses something called Protocol Buffers. It's like having a secret code that only your friends can decipher. These coded messages are super quick to send and receive because they're not bogged down with unnecessary details.

The best part? gRPC is a friendly multilingual speaker. It doesn't matter which programming language your friends speak; it can understand and talk to them all. Whether your friend codes in Java, Python, Go, C++, or another language, gRPC has a common language that everyone can understand.

And just like people have different ways of having conversations—some prefer a back-and-forth chat, while others like to spill everything at once—gRPC supports various talking styles. Whether it's a simple one-question-one-answer chat, a continuous stream of updates, or a combination of both, gRPC is up for the task.

GoLang

Golang, or Go, is like that reliable friend who's always got your back in the programming world. Born out of the creative minds at Google—thanks to Robert Griesemer, Rob Pike, and Ken Thompson—this language made its debut in 2009, and it's been turning heads ever since.

What sets Go apart is its commitment to simplicity. The creators wanted a language that's easy to learn, straightforward and doesn't make you feel like you're drowning in code. And guess what? They nailed it. Go's syntax is clean and readable, making it a great pick for both newbies and seasoned developers looking for a breath of fresh air.

Now, let's talk about concurrency—Go's party trick. It does this cool thing with goroutines, these lightweight threads that handle concurrent tasks like a champ. It's like having a bunch of friends who can juggle tasks effortlessly without stepping on each other's toes. Plus, Go throws in channels to help these goroutines communicate seamlessly, turning what could be a coordination nightmare into a piece of cake.

But Go isn't just a one-trick pony. Its standard library is like a treasure trove of goodies. Whether you're into networking, cryptography, or web development, Go's got your back. The best part? You won't find yourself drowning in third-party dependencies because Go encourages self-sufficiency.

The community around Go is something special too. It's not just a language; it's a vibe. Collaboration is the name of the game, and the Go Package Index is the hub where developers share their code adventures.

Big names like Google, Uber, and Dropbox are all aboard the Go train. Why? Because Go isn't just about looking good on paper; it's about delivering the goods in terms of speed and efficiency. Perfect for building software that can handle the big leagues, especially in the cloud-native world.

Analysis

1. **Requirements Gathering:** In this segment, the crew engages in various sports to gather and record important data approximately the application's capability, features, and consumer interface. This includes keeping meetings, carrying out interviews, and distributing surveys among stakeholders inclusive of builders. The primary goal is to acquire a clear knowledge of what the software should do and the way it has to behave so that it will meet the desires of its users.
2. **System Design:** Once the requirements are acquired, the team will proceed to broaden a comprehensive machine layout that info the general structure and business enterprise of the utility. This includes mapping out the architectural framework, figuring out the diverse components involved, and crafting visible representations showcasing how customers will engage with the software. Specifically, this procedure involves designing the database queries, specifying the personal elements, and features, and conceptualizing the layout and look of the consumer interface.
3. **Database Development:** In light of the desired wishes, our team shall craft the most desirable database schema to facilitate the control of relevant records factors such as scholar details, attendance information, grades, and administrative statistics. This

design could be based on seamless information accessibility, retrieval, and interconnectivity between numerous components.

4. **Application Development:** The improvement team will utilize the Flutter framework to create the chat application. During the software program improvement manner, they may build out every module and capability that was formerly designed for the machine. These include enforcing features like sharing files and multimedia content, beginning voice or video calls, customizing profiles with avatar pix, and organizing group conversations.
5. **Integration and Testing:** Once the application modules are advanced, we can integrate them seamlessly to ensure easy interplay and overall performance among numerous elements. This integration section will contain comprehensive testing to stumble on and clear up any troubles that may arise from records exchange, gadget compatibility, or functionalities. Furthermore, rigorous checking out methods, inclusive of unit testing and person attractiveness checking out, could be undertaken to affirm that the application conforms to its meant specifications and affords the preferred stage of performance.
6. **Deployment and Deployment:** After successful testing, the application will be prepared for deployment. This entails packaging the software for use on Android devices by generating installation files and releasing them onto platforms like the Google Play Store or alternative mobile marketplaces. Simultaneously, the necessary infrastructure, including servers or cloud services, must be set up to support the application's operation and maintain its accompanying databases. In essence, this stage involves preparing the application for widespread availability and user accessibility after ensuring its functionality and stability during testing.
7. **Maintenance and Updates:** Once an application has been launched, it's essential to prioritize ongoing maintenance and updates to guarantee seamless functionality. This involves tracking the app's performance, promptly resolving any technical glitches or complaints from users, and deploying updates to incorporate fresh functionalities or respond to user input. To maintain optimal operation, routine maintenance tasks like data backups, security upgrades, and performance enhancements are also crucial. By

focusing on these efforts, developers can safeguard against potential problems and guarantee a high-quality user experience.

It is imperative to maintain close collaborations with key parties throughout the entire software development life cycle to guarantee that the chat application satisfies expectations and offers seamless usability for everyone concerned. Consistent communication and customer input collection are essential components of this process, as they enable developers to tailor their work to meet the needs of end-users and deliver a high-quality product.

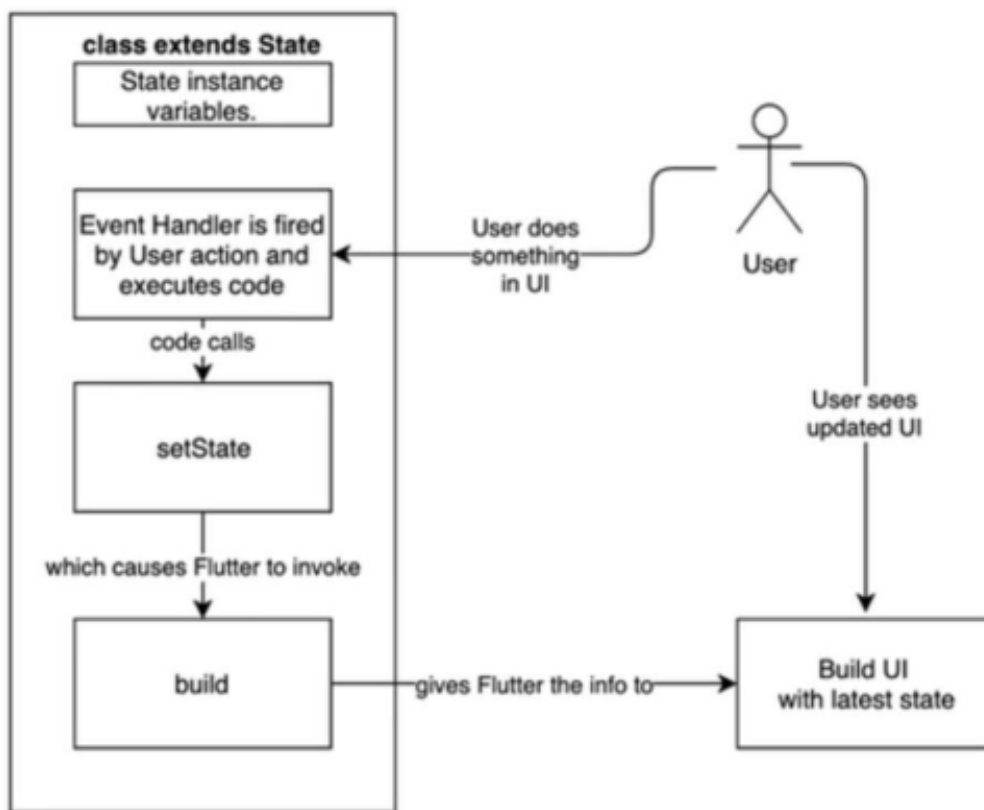


Fig. 3.1 User interaction with the Chat application

Advantages

1. Flutter offers cross-platform development capabilities, allowing developers to create a single codebase for various platforms such as iOS, Android, web, and desktop applications. This approach streamlines the development process and saves time and resources compared to creating separate codes for each platform.

2. The Flutter framework boasts faster development thanks to its 'hot reload' feature, which enables real-time updates and instant visualization of changes without requiring app restarts or loss of state. This feature significantly accelerates the development pace and simplifies modifying app appearances.
3. Flutter's architectural design ensures high-performance apps with seamless animations and transitions across different devices, making them ideal for users who prefer smoother interactions.
4. With its powerful widget architecture, Flutter enables developers to craft visually stunning and highly adaptive user interfaces that blend flawlessly into their respective platforms, providing an intuitive experience for users.
5. The vibrant Flutter developer community actively supports and contributes to the platform's growth through resource sharing and troubleshooting assistance, fostering a collaborative environment for continued innovation.

Disadvantages

1. Flutter applications can sometimes be bigger than their native counterparts because the Flutter engine and framework come packaged with each app. This added bulk means users may need extra storage space on their devices before they can download and install the software. Although the selection of third-party libraries available through Flutter's plugin architecture is expanding, it still lags behind other frameworks like React Native in terms of quantity.
2. Developers unfamiliar with Dart and the Flutter widget structure may find mastery difficult, as doing so necessitates learning a whole new language. While there are numerous online resources available to aid in this process, some programmers may nonetheless encounter difficulties. It's worth noting that while Flutter grants seamless entry to a broad array of native API functions, certain platform capabilities or low-end hardware may lack complete native API integration at present. Additionally, advanced

apps developed using Flutter may demand higher-performing hardware to run efficiently across all device

3.2 PROJECT DESIGN AND ARCHITECTURE

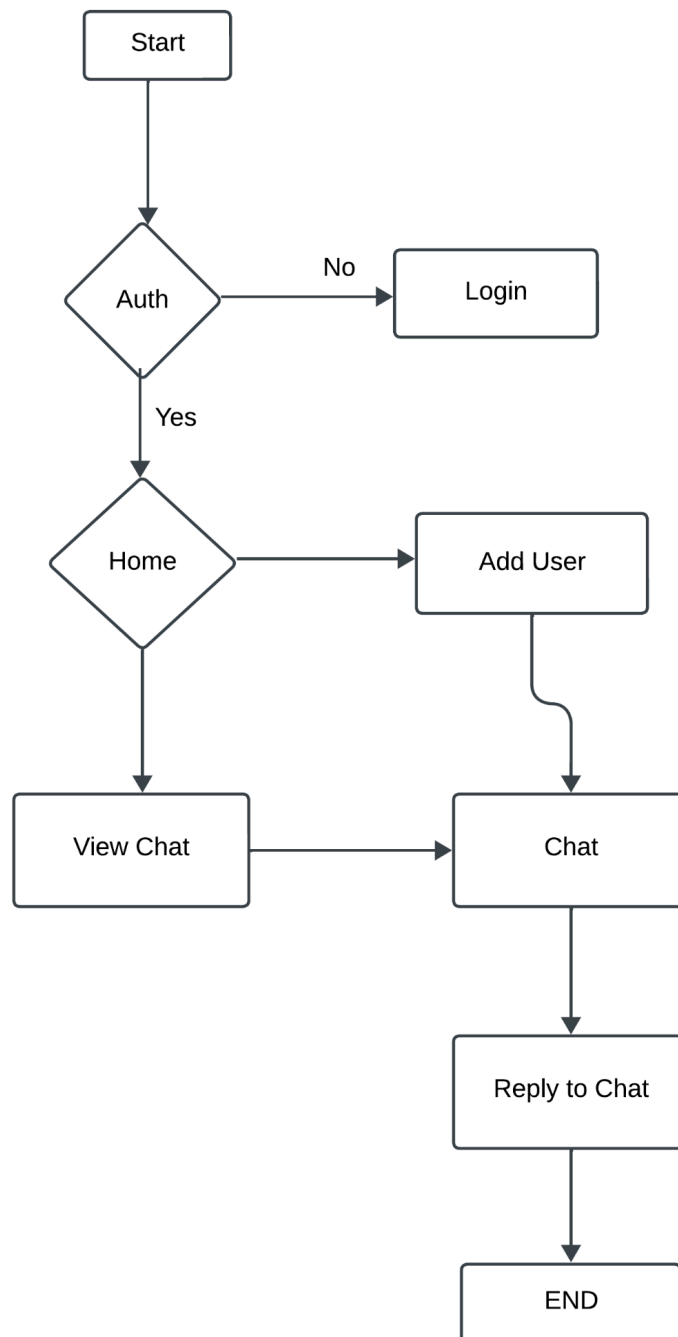


Fig 3.2 App Flow Diagram

Clean Architecture is a software architecture pattern that advocates for segregation of duties, modularity, and maintainability in application development. Its application to Flutter yields a systematic and extensible methodology for constructing reliable and testable apps.

At its essence, Clean Architecture emphasizes separating various components within an application into distinct layers, each with its unique set of obligations. The various strata that make up this architecture consist of the presentation layer, domain layer, and data layer. The presentation layer is primarily concerned with managing the User Interface (UI) aspects, including user interactions and graphical elements such as widgets while relegating significant business-related logical functions to the domain layer.

The domain layer forms the core of an application's architecture in Clean Architecture, embracing crucial business concepts, entities, and functionalities. By isolating the inherent logic from extraneous dependencies, the domain layer becomes more replicable and better equipped to withstand modifications.

In conclusion, the data layer is responsible for managing data within an application by controlling its storage, extraction, and connection to outside sources like databases, web services, or internal storage systems. Repositories, data sources, and models make up this layer, providing a barrier between the domain layer and the complexities of data management. As a result, the domain layer can concentrate exclusively on business-related activities without being sidetracked by data-related issues.

One advantage of utilizing Clean Architecture in Flutter is enhanced testability. Due to the logical partitioning of responsibilities, individual layers may be examined individually through unit tests, facilitating the identification and resolution of problems. Additionally, by separating interdependencies, such as databases or network connections, test surrogates or placeholders can be employed to mimic these dependencies throughout test processes.

Another significant benefit of Clean Architecture, this design approach also fosters greater modularity through its well-defined layering and dependency management principles. By separating concerns into distinct layers and ensuring that each layer has a clearly defined responsibility, developers can create more flexible and interchangeable parts. As a result,

developers can work on various aspects of the application without interfering with one another, leading to improved efficiency and scalability.

Furthermore, Clean Architecture enhances maintainability. Decoupling the business logic from external dependencies reduces the impact of modifications to frameworks, libraries, or UI technologies on the fundamental functionality of the application. Consequently, the application remains adaptable to evolving demands or cutting-edge innovations.

When developing Flutter applications using Clean Architecture, various design patterns and tools are employed to streamline communication between layers and manage state effectively. These include Business Logic Components (BLoC) or Providers, which help organize the application into distinct components and promote separation of concerns, modularity, and maintainability.

The result is a more structured and scalable development process that enables developers to craft high-quality, tested, and easily maintainable applications with a strong foundation for future expansion.

3.3 IMPLEMENTATION

AUTHENTICATION AND AUTHORIZATION IN FLUTTER AND GOLANG.

Authentication and authorization is one of the most important aspects of a Chat Application.

SETUP BASIC PROJECT IN GOLANG

- Create new project

```
go mod init <project-name>
```

- Install the Mango package for go

```
go get go.mongodb.org/mongo-driver
```

- Setup Database Model

- Create package db

- Create Struct to access the database

```
package db

import (
    "go.mongodb.org/mongo-driver/mongo"
)

type MongoCollections struct {
    Users *mongo.Collection
}
```

- Create a new model user.go

```
package model

import "go.mongodb.org/mongo-driver/bson/primitive"

type UserModel struct {
    ID          primitive.ObjectID `bson:"_id,omitempty"`
    Username    string              `bson:"username"`
    Password    string              `bson:"password"`
    Name        string              `bson:"name"`
}
```

- Password hashing: we are going to create a function for hashing passwords and checking passwords, Never store passwords in simple text.
 - Create package utils in root.
 - Install the required package go get golang.org/x/crypto/bcrypt.
 - Create HashPasword and CheckPassword function in password.go

```

package utils

import (
    "fmt"

    "golang.org/x/crypto/bcrypt"
)

// convert password to hash string
func HashPassword(password string) (string, error) {
    hashPassword, err := bcrypt.GenerateFromPassword([]byte(password), bcrypt.DefaultCost)
    if err != nil {
        return "", fmt.Errorf("failed to hash password %w", err)
    }
    return string(hashPassword), nil
}

// check password is valid or not
func CheckPassword(password string, hashPassword string) error {
    return bcrypt.CompareHashAndPassword([]byte(hashPassword), []byte(password))
}

```

- Setup Environment: We required a few environment variables like server_address, mongodurl, and jwt_key, etc.
 - Install Viper package for loading app.env.

```
go get github.com/spf13/viper
```

- Create a function in utils to load app.env file.

```
package utils

import (
    "time"

    "github.com/spf13/viper"
)

type ViperConfig struct {
    DBNAME          string    `mapstructure:"DB_NAME"`
    DBSource        string    `mapstructure:"DB_SOURCE"`
    RPCSERVERADDRESS string    `mapstructure:"RPC_SERVER_ADDRESS"`
    TokkenStructureKey string    `mapstructure:"TOKEN_SYMMETRIC_KEY"`
    AccessTokenDuration time.Duration `mapstructure:"ACCESS_TOKEN_DURATION"`
}

func LoadConfiguration(path string) (config ViperConfig, err error) {
    viper.AddConfigPath(path)
    viper.SetConfigName("app")
    viper.SetConfigType("env")
    viperAutomaticEnv()

    err = viper.ReadInConfig()

    if err != nil {
        return
    }
    err = viper.Unmarshal(&config)
    return
}
```

- Create app.env in root folder.

```
DB_NAME=grpc
DB_SOURCE=mongodb://localhost:27017
RPC_SERVER_ADDRESS=0.0.0.0:9090
GIN_MODE=debug
TOKEN_SYMMETRIC_KEY=12345678123456781234567812345678
ACCESS_TOKEN_DURATION=600m
```

- Setup Jwt: We need authorization in a few requests so it's going to bearer token-based authentication.
 - Install the required Jwt package

```
go get github.com/dgrijalva/jwt-go
```

- Create a new package token and create an interface to create a token and verify the token.

```
package token

import "time"

type Maker interface {
    CreateToken(id int64, username string, duration time.Duration) (string, error)
    VerifyToken(token string) (*Payload, error)
}
```

BASICS OF PROTOCOL BUFFER (.proto)

- Service: A unary service in gRPC involves a single request from the client to the server, which then sends a single response back to the client. To define a unary service, you need to create a .proto file that describes the service and the message types it uses.
- Message: A message is a data structure representing the information being sent between the client and server. It's defined using the message keyword in your .proto file.

CREATING PROTOCOL BUFFERS FOR OUR PROJECT

- Services: We will have the following services called login, signup, and get-user.
- Create user.proto, User object to return to the client.

```
syntax = "proto3";

package pb;
option go_package="github.com/djsmk123/server/pb";

message User{
    int32 id=1;
    string username=2;
    string name=3;
}
```

- Create rpc_login.proto, it will have LoginRequestMessage and LoginResponseMessage

```
syntax="proto3";
package pb;
import "user.proto";
option go_package="github.com/djsmk123/server/pb";

message LoginRequestMessage{
    string username=1;
    string password=2;
}

message LoginResponseMessage{
    User user=1;
    string access_token=2;
}
```

- Rpc_signup.proto

```
syntax="proto3";
package pb;
import "user.proto";
option go_package="github.com/djsmk123/server/pb";

message SignupRequestMessage{
    string username=1;
    string password=2;
    string name=3;
}

message SignupResponseMessage{
    User user=1;
}
```

- create message rpc_get_user.proto:

```
syntax="proto3";
package pb;
import "user.proto";
option go_package="github.com/djsmk123/server/pb";

message GetUserResponse{
    User user=1;
}
```

- Create service rpc_services.proto

```
syntax="proto3";
package pb;
option go_package="github.com/djsmk123/server/pb";
import "empty_request.proto";
import "rpc_get_user.proto";
import "rpc_login.proto";
import "rpc_signup.proto";

service GrpcServerService {
  rpc SignUp(SignupRequestMessage) returns (SignupResponseMessage){};

  rpc login(LoginRequestMessage) returns (LoginResponseMessage){};
  rpc GetUser(EmptyRequest) returns (GetUserResponse) {};
}
message EmptyRequest{
}
```

GENERATE CODE FOR GOLANG

- Create pb package in root folder.
- Run command to generate equivalent code for golang

```
protoc --proto_path=proto --go_out=pb --go_opt=paths=source_relative \
  --go-grpc_out=pb --go-grpc_opt=paths=source_relative \
  proto/*.proto
```

So, here is how the Data serialization will take place after running the above command.

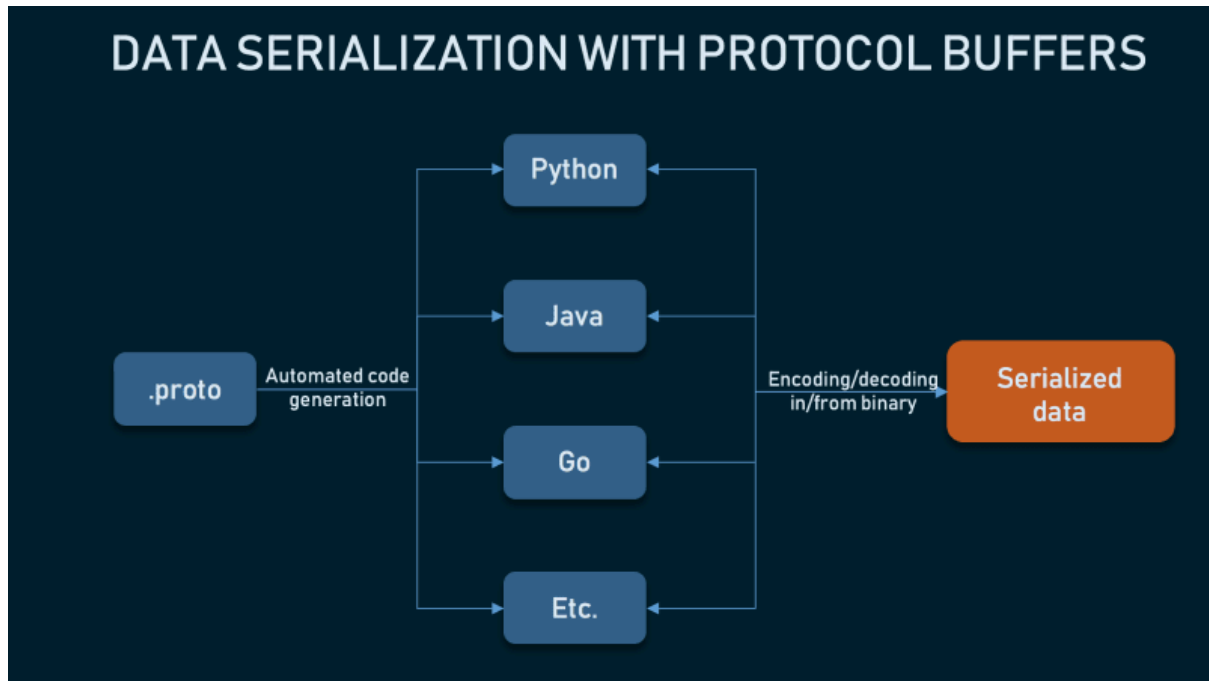


Fig 3.3 Data Serialization with Protocol Buffers

CREATE AUTH FUNCTION

- Create Auth Package: This package has basic functions like get-user, login, signup, tokenCreation, and a function to convert MongoDB userModel to gRPC user message.
- Create converter.go in the auth package.

```
package auth

import (
    "github.com/djsmk123/server/db/model"
    pb "github.com/djsmk123/server/pb"
)

func ConvertUserObjectToUser(model *model.UserModel) *pb.User {
    return &pb.User{
        Username: model.Username,
        Name:     model.Name,
        Id:       int32(model.ID.Timestamp().Day()),
    }
}
```

- Create login.go

```
package auth

import (
    "context"
    "errors"

    "github.com/djsmk123/server/db/model"
    "github.com/djsmk123/server/utils"
    "go.mongodb.org/mongo-driver/bson"
    "go.mongodb.org/mongo-driver/mongo"
)

var ErrUserNotFound = errors.New("user not found")
var ErrInvalidCredentials = errors.New("credentials are not valid")

func LoginUser(username string, password string, collection *mongo.Collection,

    filter := bson.M{"username": username}
    var user model.UserModel
    err := collection.FindOne(context, filter).Decode(&user)
    if err == mongo.ErrNoDocuments {
        return nil, ErrUserNotFound
    } else if err != nil {
        return nil, err
    }
    err = utils.CheckPassword(password, user.Password)
    if err != nil {
        return nil, ErrInvalidCredentials
    }
    return &user, nil
}
```

CONNECT THE WHOLE AND START SERVER

Finally, we are going to the main function in main.go, load the environment, connect the database and RunServer.

BI-DIRECTIONAL COMMUNICATION (HTTP/2)

gRPC: Bi-directional communication (HTTP/2)

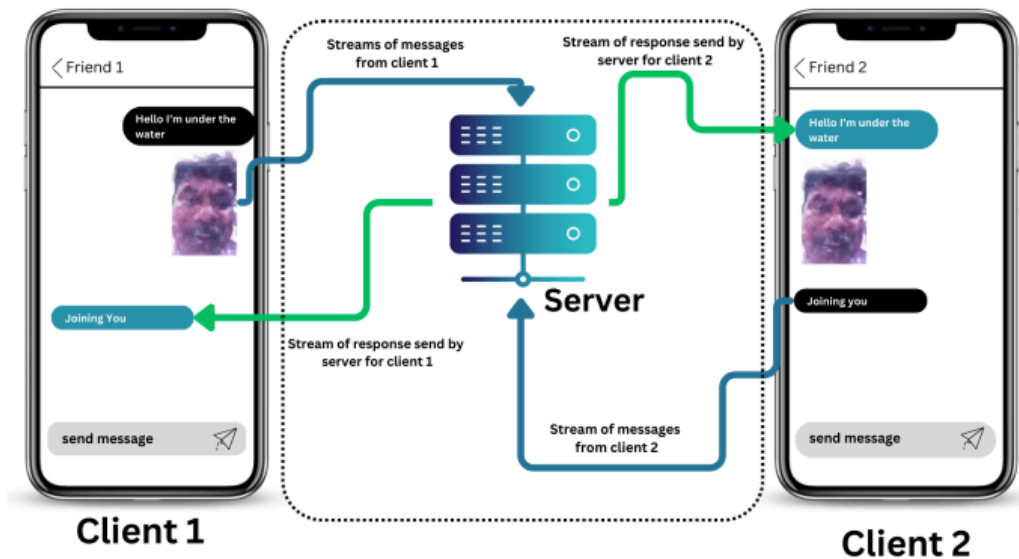


Fig 3.4 Bi-directional Communication

SETTING UP THE CHAT SERVICES

Establishing a chat service is a straightforward procedure.

- RPC Service Creation: Creating an RPC service is at the core of this setup.
- Bidirectional Message Stream: Implement a bidirectional stream of messages. Messages are initiated by the client and sent to the server.
- Bidirectional Response Stream: Creating a bidirectional stream for responses. Responses are generated by the server and sent back to the client.
- Acknowledgment Mechanism: To facilitate a seamless exchange of information between the client and server, it is crucial to establish an initial confirmation process before actual communication commences. This protocol verifies that both parties are connected successfully, thus ensuring a smooth interaction.

- Database Handling: Notably, the acknowledgment message serves only as a temporary notification and is not preserved within the database.
- Confirmation of Connection: The acknowledgment message serves as a confirmation of a successful connection.
- Setting up protos: Setup proto buffs for Sending and getting message

```

syntax="proto3";
package pb;

option go_package="github.com/djsmk123/server/pb";
import "google/protobuf/timestamp.proto";

message Message{
    string id=1;
    string sender=2;
    string receiver=3;
    string message=4;
    google.protobuf.Timestamp created_at = 5;
}

message SendMessageRequest{
    string message=1;
    string reciever=2;
}

message GetAllMessagesRequest{
    string reciever=1;
}

message GetAllMessagesResponse{
    repeated Message messages=1;
}

```

- Add these rpc functions into the service

```
service GrpcServerService {
  //add these lines
  rpc SendMessage(stream SendMessageRequest) returns (stream Message){};
  rpc GetAllMessage(GetAllMessagesRequest) returns (GetAllMessagesResponse){};
}
```

- Again generate equivalent code

```
protoc --proto_path=proto --go_out=pb --go_opt=paths=source_relative \
  --go-grpc_out=pb --go-grpc_opt=paths=source_relative \
  proto/*.proto
```

- Now create a Message Object for mongodb collection

```
package model

import (
    "time"

    "go.mongodb.org/mongo-driver/bson/primitive"
)

type Message struct {
    ID          primitive.ObjectID `bson:"_id,omitempty"`
    Sender      string              `bson:"sender"`
    Receiver    string              `bson:"receiver"`
    Message     string              `bson:"message"`
    CreatedAt   time.Time           `bson:"created_at"`
}
```

- Update MongoCollection

```
type MongoCollections struct {
    Users *mongo.Collection
    Chats *mongo.Collection
}
```

- Now deport the apk and install it on your respective devices.

AMAZON WEB SERVICE (AWS)

Amazon Web Services is a widely used cloud computing platform powered by Amazon. It offers many services including storage options, networking, databases, elastic kubernetes services, elastic container services and much more.

Some key aspects of AWS:

1. Compute Services

- a. AWS Lambda: Lambda offers serverless computing services that allow one to run applications without managing servers.
- b. Amazon EC2: Amazon Elastic Compute Cloud is a virtual server in the cloud that can be configured and scaled based on demand.
- c. Amazon EKS: Amazon Elastic Kubernetes Service offers pre-configure kubernetes environment, which auto-manages the clusters.

2. Storage Services

- a. Amazon S3: Simple Storage Service are scalable object storage for storing or backing up data.
- b. Amazon EBS: Elastic Block Stores are persistence block storages that are configured with EC2 instances once the volume is increased cannot be decreased of such storage stores.

3. Database Services

- a. Amazon RDS: Amazon Relational Database Services are managed data stores that support MySQL, PostgreSQL, Oracle, and SQL.
- b. Amazon DynamoDB: Dynamo Databases are fully managed NoSQL database services for applications needing consistent, single-digit millisecond latency.

4. Networking

- a. Amazon VPC: Virtual Private Cloud enables one to provision a logically isolated section of the AWS Cloud where one can launch their AWS resources.
- b. Amazon Route 53: Route 53 is a scalable domain name system (DNS) web service that overwrites an IP address with the purchased domain name.

VPC Architecture

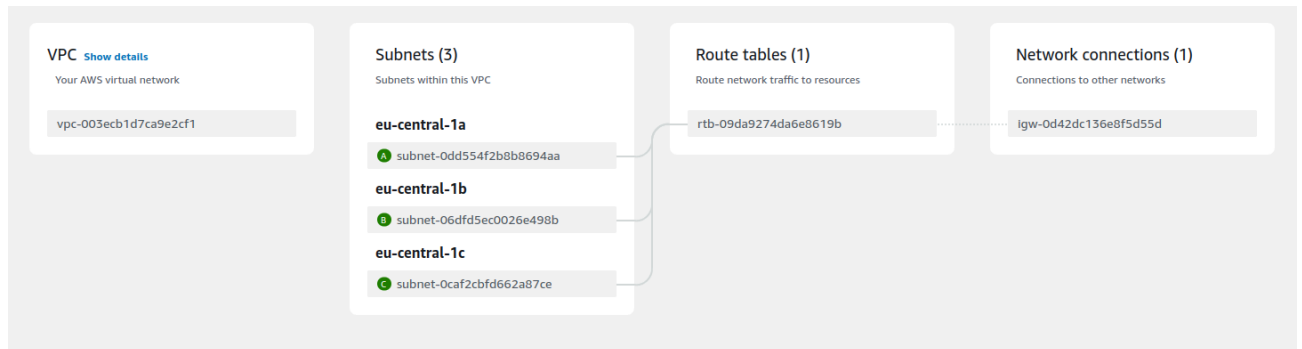


Fig 3.6 VPC Architecture

The above architecture shows the Virtual Private Cloud that contains many different components, are:

1. **VPC (Virtual Private Cloud):** A logically isolated section of the AWS Cloud where you can define your virtual network environment, control IP address ranges, and configure route tables and network gateways.
2. **Subnet:** A segmented portion of a VPC's IP address range in which you can place resources like EC2 instances, allowing for network isolation and security control.
3. **Route Table:** A set of rules, or routes, that determine where network traffic is directed within a VPC, specifying which subnets or external network traffic should be routed to.
4. **Network Connections:** These establish communication links between resources within a VPC or between a VPC and external networks, enabling connectivity and data transfer across different network environments.

Load Balancer

awseb-AWSEB-IG1KIGK71C27

Details

arn:aws:elasticloadbalancing:eu-central-1:225000673436:targetgroup/awseb-AWSEB-IG1KIGK71C27/844cb0084d8fc8d

Target type Instance	Protocol : Port HTTP: 80	Protocol version HTTP1	VPC vpc-003ecb1d7ca9e2cf1
IP address type IPv4	Load balancer awseb-AWSEB-Y8VZ6UH8LSXP		

1 Total targets

1 Healthy

0 Unhealthy

0 Anomalous

0 Unused

0 Initial

0 Draining

Distribution of targets by Availability Zone (AZ)

Select values in this table to see corresponding filters applied to the Registered targets table below.

Registered targets (1)

Instance ID	Name	Port	Zone	Health status	Health status details	Launch...	Anomaly detection result
i-042deabfbd7e413	Testing-env	80	eu-central-1a	Healthy	-	April 17, 2...	Normal

Fig 3.7 Load Balancer

AWS offers several types of load balancers that distribute incoming application or network traffic across multiple targets, such as EC2 instances, containers, or IP addresses. These load balancers enhance the availability and fault tolerance of your applications by ensuring that no single resource becomes overwhelmed with traffic. Amazon Application Load Balancer (ALB) is a powerful service provided by AWS for distributing incoming application traffic across multiple targets, such as EC2 instances, containers, or IP addresses. ALB operates at the application layer (Layer 7) of the OSI model, which allows it to make routing decisions based on content such as the URL, HTTP headers, or request method.

Key Pair

chat-keypair rsa

Fig 3.8 Key Pairs

An Amazon key pair is a set of cryptographic keys used to securely connect to Amazon EC2 instances. When you launch an EC2 instance, you specify a key pair that consists of a public key and a private key.

Security Group

sg-0e6d8949dd265f182 - launch-wizard-29

Details

Security group name launch-wizard-29	Security group ID sg-0e6d8949dd265f182	Description launch-wizard-29 created 2024-03-1
Owner 225000673436	Inbound rules count 4 Permission entries	Outbound rules count 1 Permission entry

Inbound rules | Outbound rules | Tags

Inbound rules (4)

<input type="checkbox"/>	Name	Security group rule...	IP version	Type	Protocol	Port range
<input type="checkbox"/>	-	sgr-06d528a4595f0549f	IPv4	HTTP	TCP	80
<input type="checkbox"/>	-	sgr-0a6bb3b7112625bcf	IPv4	Custom TCP	TCP	8443
<input type="checkbox"/>	-	sgr-06b497994b787a...	IPv4	HTTPS	TCP	443
<input type="checkbox"/>	-	sgr-0491a9a24fbcb152	IPv4	SSH	TCP	22

Fig 3.9 Security Groups

An Amazon Security Group in AWS functions as a virtual firewall for EC2 instances, regulating inbound and outbound traffic based on defined rules. Each security group is applied at the instance level and controls traffic based on protocols, ports, and IP address ranges. By default, all inbound traffic is denied unless explicitly allowed through security group rules, while outbound traffic is permitted by default. Security groups operate in a stateful manner, meaning return traffic for allowed connections is automatically permitted. They offer immediate application of changes without instance restarts and can be configured with granular controls for specific traffic requirements. Working alongside Network Access Control Lists (NACLs), security groups provide layered security for EC2 instances within AWS networks, enforcing the principle of least privilege and enhancing overall network security. Properly configuring security groups is essential for maintaining a secure and compliant AWS environment.

EC2 Instance

Instance summary for i-04c75abd66fc6f607 (aerial-tools) Info

Updated less than a minute ago

Instance ID i-04c75abd66fc6f607 (aerial-tools)	Public IPv4 address 52.59.240.157 open address	Private IPv4 addresses 172.31.17.181
IPv6 address -	Instance state Running	Public IPv4 DNS ec2-52-59-240-157.eu-central-1.compute.amazonaws.com open address
Hostname type IP name: ip-172-31-17-181.eu-central-1.compute.internal	Private IP DNS name (IPv4 only) ip-172-31-17-181.eu-central-1.compute.internal	Elastic IP addresses -
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. Learn more
Auto-assigned IP address 52.59.240.157 (Public IP)	VPC ID vpc-003ecb1d7ca9e2cf1	Auto Scaling Group name -
IAM Role -	Subnet ID subnet-0dd554f2b8b8694aa	
IMDSv2 Required		

Fig 3.10 EC2 Instance

An Amazon EC2 (Elastic Compute Cloud) instance is a virtual server in the AWS cloud that can be provisioned and scaled according to workload demands. EC2 instances come in various types optimized for different use cases, such as compute-intensive, memory-intensive, or storage-intensive workloads. Each EC2 instance is assigned virtual CPU (vCPU) cores, memory (RAM), storage, and networking capacity based on the selected instance type. Instances can run different operating systems and software applications, making them versatile for a wide range of use cases from web hosting to data processing and machine learning tasks. EC2 instances are billed based on usage hours, offering cost-effective and flexible computing resources in the cloud.

Route 53

Route 53 > Hosted zones

Hosted zones (3) View details Edit Delete

Automatic mode is the current search behavior optimized for best filter results. To change modes go to settings.

Filter records by property or value

Hosted zone name	Type	Created by	Record count	Description	Hosted zone ID
chatter.com	Public	Route 53	2	-	Z07262833FK8ZS09CXZC9

Fig 3.11 Route 53

Amazon Route 53 is a scalable and highly available Domain Name System (DNS) web service provided by AWS, offering reliable domain registration, DNS routing, and

health-checking capabilities. It efficiently translates human-readable domain names into IP addresses to route end-user requests to applications hosted on AWS or elsewhere. Route 53 supports various DNS routing policies, including simple routing, weighted routing, latency-based routing, geolocation routing, and failover routing, allowing flexible traffic management and failover strategies. It integrates seamlessly with other AWS services, enabling automated DNS management and providing a reliable and performant DNS solution for applications deployed on AWS infrastructure.

Target Group

The screenshot displays the AWS Management Console interface for a Target Group. At the top, the Target Group ID is 'awseb-AWSEB-IG1KIGK71C27'. The 'Details' section shows the following information:

- Target type:** Instance
- Protocol : Port:** HTTP: 80
- Protocol version:** HTTP1
- VPC:** vpc-003ecb1d7ca9e2cf1
- Load balancer:** awseb-AWSEB-Y8V26UH8LSXP

Below the details, a summary row indicates: 1 Total targets (1 Healthy, 0 Unhealthy), 0 Anomalous, 0 Unused, 0 Initial, and 0 Draining. A section titled 'Distribution of targets by Availability Zone (AZ)' is present but currently empty. The 'Registered targets' section shows one target with the following details:

Instance ID	Name	Port	Zone	Health status	Health status details	Launch...	Anomaly detection result
i-042deabfbd7e413	Testing-env	80	eu-central-1a	Healthy	-	April 17, 2...	Normal

Fig 3.12 Target Group

An Amazon Target Group is a logical grouping of targets, such as EC2 instances, ECS tasks, or Lambda functions, that receive traffic from an Application Load Balancer (ALB) or Network Load Balancer (NLB) based on defined routing rules. Target groups allow for fine-grained control over how traffic is distributed among backend resources, supporting features like health checks to ensure only healthy targets receive traffic. They enable dynamic scaling and management of application components by automatically registering and deregistering targets based on workload conditions, making them integral to modern, scalable architectures deployed on AWS load balancers.

SSL Certificate

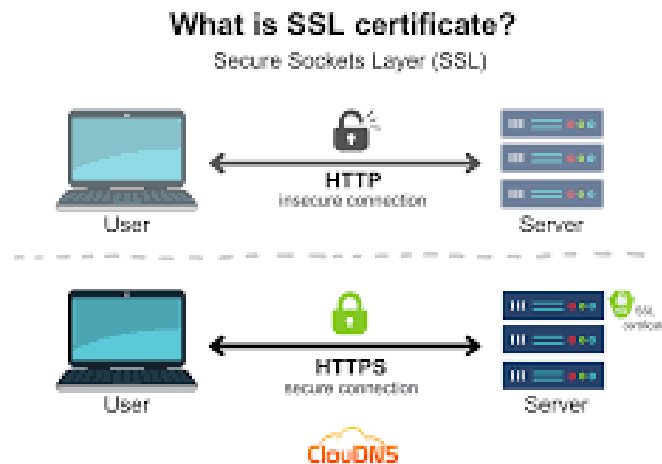


Fig 3.13 SSL Certificate

An SSL certificate (Secure Sockets Layer) is a digital certificate that encrypts data transmitted between a user's browser and a web server, ensuring secure communication over the internet. It verifies the identity of the website and establishes a secure connection using HTTPS protocol, protecting sensitive information such as login credentials and payment details. SSL certificates are issued by Certificate Authorities (CAs) and come in various types, including domain validated (DV), organization validated (OV), and extended validation (EV), offering different levels of validation and trust. They are essential for building trust with users and improving website security, as modern browsers indicate secure connections with a padlock icon and "https://" prefix. AWS Certificate Manager (ACM) provides a managed service for provisioning and managing SSL/TLS certificates for use with AWS services like Elastic Load Balancing, CloudFront, and API Gateway.

3.4 KEY CHALLENGES

Key challenges that we faced while building this project are mentioned below:

1. gRPC Setup and Configuration: Integrating gRPC with Golang and ensuring that both the server and client are configured correctly was a complex task. Managing protocol buffers and service definitions was challenging.

2. Proto File Management: Managing protocol buffer (.proto) files for communication between Flutter and Golang was really difficult. Changes in the protocol must be carefully handled so that the communication between the client and server may not break.
3. State Management in Flutter: Flutter offers various state management solutions, so managing and choosing the right one for the application was crucial.
4. Real-Time Updates: Implementing bidirectional real-time updates in a chat application.
5. User Authentication and Authorization: Ensuring a secure authentication and authorization system to ensure that only authorized users can access the chat, and protecting sensitive user data during communication.
6. Cross-Platform Compatibility: Flutter application works seamlessly across different platforms like iOS, Android, or the web. UI/UX considerations, responsiveness, and platform-specific nuances was needed to be addressed.
7. Optimizing for Mobile Performance: Flutter applications face performance challenges on mobile devices. Optimizing the application for performance, minimizing resource usage, and handling network requests efficiently are critical.
8. Error Handling and Debugging: Identifying and handling errors gracefully in both the Flutter client and Golang server, and setting up effective debugging tools, is crucial for a robust and maintainable application.
9. Scalability: Designing the system to handle a growing number of users and messages without sacrificing performance can be a significant challenge.
10. Security Concerns: Addressing security troubles which include securing conversation channels, preventing records breaches, and protecting towards not unusual internet vulnerabilities is crucial for manufacturing-prepared chat software.

CHAPTER 4: TESTING

4.1 TESTING STRATEGY

A comprehensive trying-out approach for the chat utility is vital to assure its clean operation, short response instances, and usual consumer satisfaction. As part of this method, performance evaluation includes figuring out ability issues and addressing them before they turn out to be issues. One key thing of the checking out strategy is assessing the application's reaction time and velocity. To ensure the choicest performance, developers need to reveal and examine key indicators along with page loading instances, statistics retrieval speeds, and reaction times to person inputs. This involves measuring those metrics and decoding their impact on the general consumer level in by way of identifying areas wherein enhancements are needed to enhance the performance of the software.

A crucial aspect of performance evaluation is assessing how well an application uses system resources and achieves its intended goals. Tracking the amount of system resources used by the software allows developers to recognize any inefficiencies or memory leaks that might cause instability or decreased functionality over time. To optimize resource use, techniques like proper garbage collection and implementation of effective data structures are employed, resulting in enhanced reliability and faster execution.

The networking capabilities of the college administration application must also undergo thorough examination during performance analysis. Measuring the time required to acquire information from distant machines, streamlining network demands, and coping with possible connection failures are all included in this. Effective network procedures and caches can substantially improve the app's general performance, particularly when working with big datasets or live updates

Battery use evaluation is essential to determine how much an application affects a gadget's battery lifespan. By analyzing battery usage trends, developers may identify areas where energy consumption is high, such as resource-intensive features or background processes consuming too much power. Addressing these issues through optimization techniques like

reducing network requests or optimizing background tasks can improve overall efficiency and user satisfaction with the app.

Another critical component of functionality analysis is user experience evaluation. Collecting user input and carrying out usability tests allows designers to gain insightful knowledge regarding the software's simplicity of use, intuition, and all-around gratification. Solving usability problems and enhancing the program's UI/UX can significantly affect end-user involvement and impression of overall performance.

Apart from a user-focused Testing approach, developers must also consider backend functionality. Evaluation of server-side operation entails checking data extraction and processing, database questions, and API replies. Enhancing backend performance enables effective interaction between the application and back-end structures, thereby boosting the total output and user encounter.

In essence, a rigorous testing approach for the chat app is essential to pinpoint and rectify any performance-related concerns that might impede the app's effectiveness, reactivity, or overall user experience. Through examination of factors like response time, memory utilization, network performance, power consumption, user satisfaction, and back-end functionality, developers can enhance the app's performance, thereby delivering a seamless, efficient, and enjoyable user experience. Ongoing testing and optimization initiatives are necessary to maintain the app's optimal functioning and user-friendliness over time.

CHAPTER 5: RESULTS AND EVALUATION

5.1 RESULTS



Register

Email

Password

Accept [terms and conditions](#) to register

[Register](#)

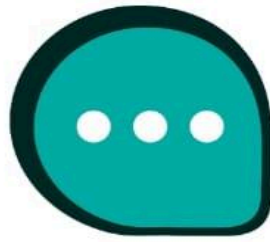
OR

[Register anonymously](#)

Already have an account ?

[Login](#)

Fig 5.1 Register Page



Register

Email



pandeyabhimanyu274@gmail.com

Password



.....



Accept **terms and conditions** to register

Register

OR

Register anonymously

Already have an account ?

Login

Fig 5.2 User Credentials



Email



Password

[Forgot password](#)

Login

Create an account or register anonymously?

Register

Fig 5.3 Login Page



kumarrakshit402

kumarrakshit402@gmail.com








-  Edit profile
-  Preferences
-  Blocked users
-  Hidden users
-  Delete account
-  Terms & privacy policy
-  FAQs



Fig 5.4 Settings Page

< Edit profile



Email

✉ kumarrakshit402@gmail.com

Name

👤 kumarrakshit402

Age

18



Male



Female



Other

Choose a country

Global

Choose a language

English

Bio



I love chatting and making new friends! Exploring the world through words. Curious and always ready to learn. Let's connect and share our

150/160

Fig 5.5 Edit Profile Page

Chatter



Chatter AI



pandeyabhimanyu274

02:19 PM

Message deleted

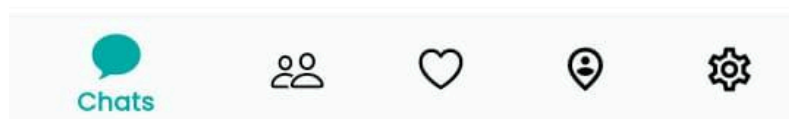




Fig 5.6 All Chats Window

<  **pandeyabhimanyu...**  
Online



 hiii...
02:20 PM 

Fig 5.7 Chat Window



kumarrakshit402
Online



hii....
02:20 PM

hlo buddy
02:20 PM ✓



Type here...



Fig 5.8 Chat Window

Chatter




No Squad found [add now](#)



Fig 5.9 Group Window

< Preferences

 Notifications

 Private Profile

 Show AI Bot

Fig 5.10 Preference Window

CHAPTER 6: CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION

The real-time application is an extremely useful tool that simplifies tasks, strengthens connection platforms, and makes it easier for people to find what they need. Using cutting-edge technology, this program provides a quick and easy way to connect with others in real-time, resulting in a more organized and open environment.

Through the real-time chat application, and integration of Flutter, it administrators a sleek and responsive user interface that ensures a smooth user experience across various platforms like iOS, Android, Linux, or Windows. Leveraging the power of gRPC and Golang, the backend of the application is equipped with robust and scalable communication protocols, enabling swift and reliable message delivery.

On the contrary, individuals are granted extensive privileges within the platform, allowing them to join open groups and take advantage of various features. These benefits enable them to remain well-informed about events occurring globally, actively contribute to their personal growth, and connect with others via discussions, video calls, and forming groups.

Making the chat application openly available to the world through platforms like GitHub, has made sure that the development of the application doesn't stop. Making it open-source enables developers around the globe to collaborate and work on it together. Also, it helps to project the user data more efficiently.

This chat application not only exemplifies technological prowess but also addresses the contemporary need for instant and reliable communication. By amalgamating the strengths of Flutter, gRPC, and Golang, it not only meets but exceeds the expectations of a modern chat platform, providing users with a cutting-edge and enjoyable communication experience.

6.2 FUTURE SCOPE

Improving the UI/UX of the application, integrating many more features like video calling, calling, forming groups, and AI chatbot many more.

Implementing blockchain technology to enhance the security of the chat application. Pushing the Chat application on Apple and Play Store, and also making it work on different desktops and web browsers.

By making the Chat application an open-source project, we encourage students on campus to participate in its development through collaboration. This enables us to keep the project current with the most recent developments in app creation by drawing from a diverse pool of knowledge and expertise. Our ultimate goal is to establish a reliable and communally driven platform that caters specifically to the needs of aspiring app developers within our academic community.

REFERENCE

1. H. Ardiansyah and A. Fatwanto, "Comparison of Memory usage between REST API in Javascript and Golang", *MATRIK: Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer*, vol. 22, no. 1, pp. 229-240, Nov. 2022.
2. SARI, Anisa Selena; HIDAYAT, Rahmat. "Designing website vaccine booking system using golang programming language and framework react JS", *JISICOM (Journal of Information System, Informatics and Computing)*, [S.l.], v. 6, n. 1, p. 22-39, june 2022. ISSN 2597-3673.
3. O. Dahl, "Exploring End User's Perception of Flutter Mobile Apps," Bachelor's Thesis, Department of Computer Science, Malmö University, Malmö, Sweden, February 07, 2019.
4. Wenhao Wu "React Native vs Flutter, Cross-platform Mobile Application Frameworks." Bachelor's Thesis, Metropolia University of Applied Sciences, Bachelor of Engineering in Information Technology, March 01, 2018.
5. S. Boukhary and E. Colmenares, "A Clean Approach to Flutter Development through the Flutter Clean Architecture Package," 2019 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2019, pp. 1115-1120, doi: 10.1109/CSCI49370.2019.00211.
6. C. L. Chamas, D. Cordeiro and M. M. Eler, "Comparing REST, SOAP, Socket and gRPC in computation offloading of mobile applications: An energy cost analysis," 2017 IEEE 9th Latin-American Conference on Communications (LATINCOM), Guatemala City, Guatemala, 2017, pp. 1-6, doi: 10.1109/LATINCOM.2017.8240185.
7. X. Wang, H. Zhao, and J. Zhu, "gRPC: A Communication Cooperation Mechanism in Distributed Systems," *ACM SIGOPS Operating Systems Review*, vol. 27, no. 3, pp. 75-86, Jul. 1993, doi: 10.1145/155870.155881.

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:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	<ul style="list-style-type: none">• All Preliminary Pages• Bibliography/Images/Quotes• 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

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