
# Department of Computer Engineering

## Secure Chat Application Report

05/01/2023

Ahmet Kaan Memioğlu – 1900005528
Şükrü Erim Sinal – 1900003587
Emrecan Üzüm – 1900005485

## Outline

### 1-) Introduction

### 2-) Tools  / Frameworks

- Main tools
- Non Functional testing tools
- Functional testing tools

### 3-) Development Method

- Methodology
- Planning
- Organization

### 4-) Flow of the Program

- Block Diagram
- Database Diagram
- Technical Diagram
- MVP Diagram

### 5-) Testing Procedures

- Non Functional Test Procedures
- Functional Test Procedures

### 6-) Target Audience

**GitHub Repository of Our Project**

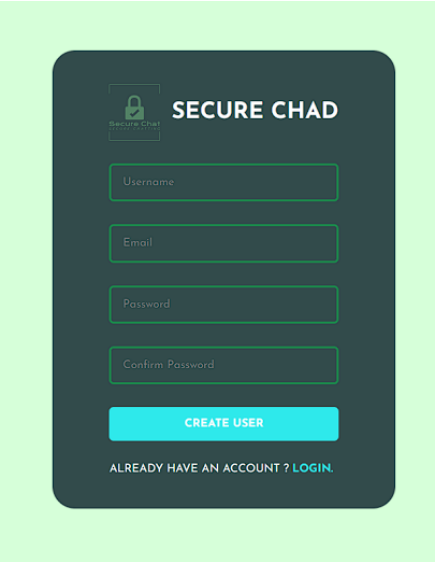
# 1-) Introduction : Secure Chad

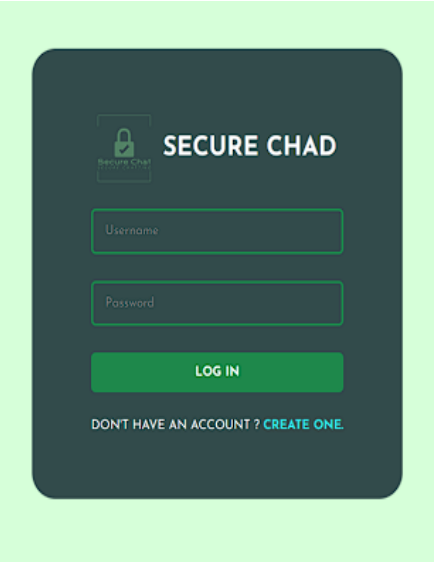
Our project is a full stack MERN project which means it utilizes Mongodb Express React and Node Js. Our front-end is written in React and in the Backend we have utilized express to run on localhost:3000 and in localhost we are collecting data which we integrated with our code via mongoose package. This package gets the Users name, emails, avatar, message sender, message receiver, messages itself(which are encrypted) and automatically stores them into two categories: chat and users tables respectively.

Here is our brief summary of our project:

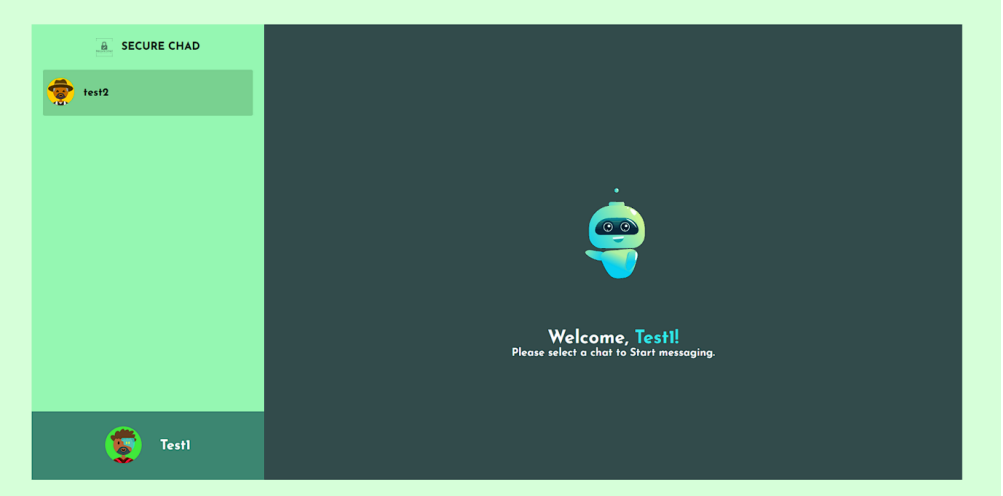
We have our first screen which is our registration page

If a user has created a fresh account after registering they get to choose an avatar for themselves

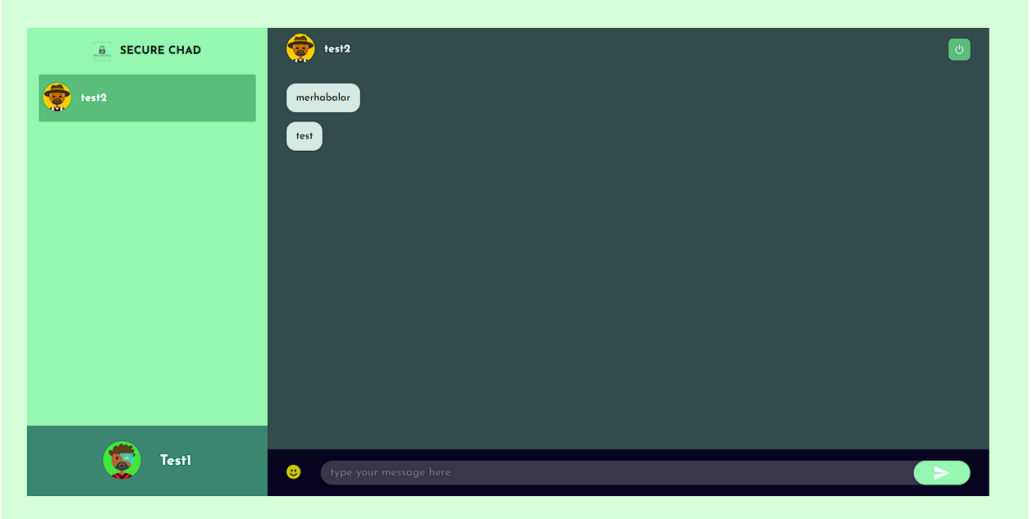



After setting up their account the user is greeted with a user screen that is located on their left screen as you can see.



And then you can pick a conversation up with the user and proceed to chat up. while chatting our AES encryption is doing its work: encrypting and decrypting respectively the encrypted messages are saved to mongodb with mongoose package.for further conversations of offline chat or logging the chat




For chatting we have also included a emoji picker and implemented a search bar for emoji searching we will try to implement this feature to our users sidebar because a concern of ours for future development is after some usage of the app the users tab can be clustered and full we can try to solve this issue by searching the chats.

## 2-) Tools / Frameworks

First of all we have used for development MERN ( MongoDB, Express React NodeJs) also Mongoose, Socket.io, Scrypt and Axios.

For the testing part, the helpful tools are Google Lighthouse, Cypress and Cypress-Chrome-Reader.


These testing materials divided into two category;
Non-Functional and Functional

For the non-functional part we use Google Lighthouse, because it's the best system due to our research, it has the best accessibility and also it's the best in overall performance.

For the functional part we used Cypress as a testing framework.

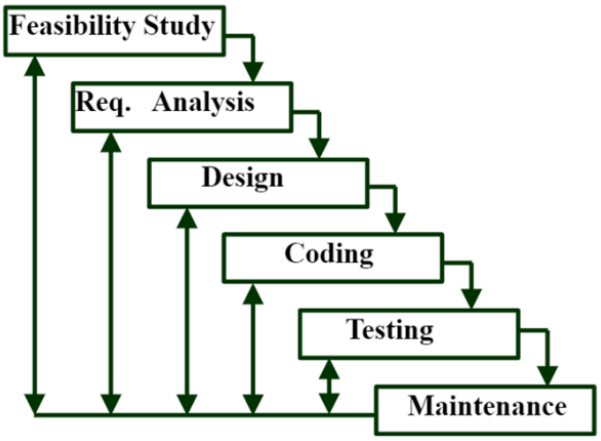
It gives best opportunities for testing algorithms such as recording Keystrokes, implementing scenarios and test features.

## 3-) Development Method

In addition , we decided to use the “iterative waterfall model”. From our experiences in older projects, the iterative model provides us an application with accurate and less buggy software if the process is employed according to schedule. Unfortunately, since we repeat each process one by one to find out all defects and errors, this can cause time-wasting problems as a result. Those are only problems we have encountered so far.

Some advantages about Iterative Waterfall:

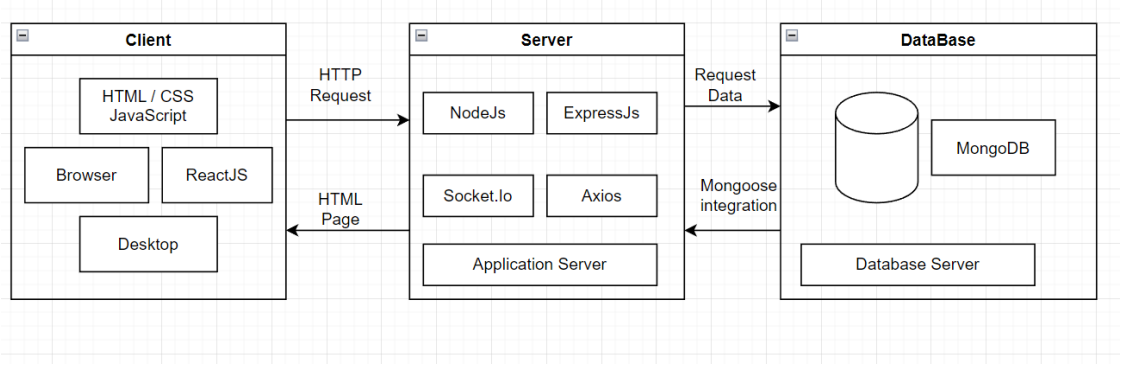
- Creating accurate applications.
- Providing a less buggy experience
- Repeating and refining features
- Beneficial for a small team like us




For methodology, the reason we chose this methodology was to finish coding quickly after the analysis and design phase, and to be able to easily return to the coding phase in case of errors when we test.

On the project organization side, everyone is aware of the overall functioning of the system and has authority over it. In the analysis and planning part we have identified three main parts: the connection and functioning of the system with the database, the communication and message management between users, and finally the security part based on encryption and decryption.

## 4-) Flow of the Program



First of all, our npm packages need to be downloaded to run the program. We explained this very clearly in the readme section of the project. In two terminals, we are changing the directory to the “public” and “server” folders separately. After we reach the location of packages, we basically write the “npm i” command for both of them to install necessary packages that are written in the “packages.json” file Then we run the “yarn start” command for both terminals.

Congrats, now we can reach the application that we need. This process opens the server system and client side socket protocol. Socket is the name of the protocol that gives us instantaneous data transfer

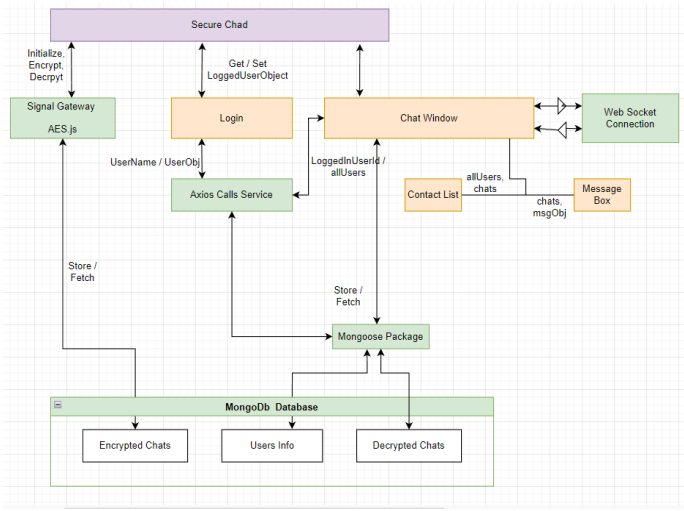
```
const handleSendMsg = async (msg) => {
  const data = await JSON.parse(
    localStorage.getItem(process.env.REACT_APP_LOCALHOST_KEY)
  );

  console.log("Mesaj:" + msg);
  let key = KeyExpansion(genRanHex(32));
  let encrypted = ''
  let hex = hexconvert(msg)
  console.log("Hex:" + hex);
  let loop = parseInt(hex.length / 32);
  for (let x = 0; x < loop; x++) {
      encrypted += AES_128_Encryption(hex.substring(32 * x,32*(x+1)),key) /// encryption
  }
  console.log(encrypted);
  socket.current.emit("send-msg", {// sokete giden mesaj
    to: currentChat._id,
    from: data._id,
    encrypted,
    key,
  });
  await axios.post(sendMessageRoute, {
    from: data._id,
    to: currentChat._id,
    message: msg,
  });
```

After seeing the above code block, it is quite simple to understand how the code works outside the background.

The red part represents the backend part that takes the message input from the user and brings it to our code.

The green part is the encryption part of the message. The blue part represents how the socket works. We're publishing our encrypted message, our randomly generated key, chat_id and data_id to the room.

The last two parameters are necessary for the database. Finally the orange part of code is all about collecting the detailed message information in our database.

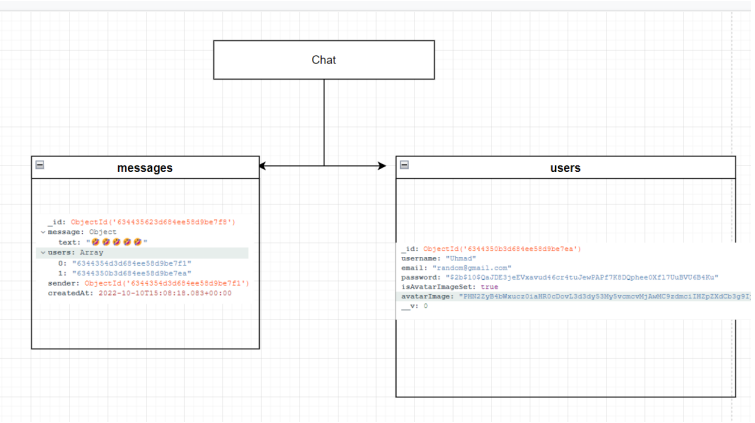



When we get a new message from our socket, first the system tries to decrypt it. After the decrypted part, write it to our screen. This is the most basic approach to our secure chat system. If we want to put this project in real world, we need some funds for;

Deploy the server part (backend) of our project to the AWS like server base.

Deploy the client part (frontend) of our project to a domain. Making a guess of this process's cost is so unpredictable, because it depends on the user count of our system.

If I need to give some hints about what is the next feature of our project;




● We need to implement group chat rooms like whatsapp.

● We will find a better way to hide the process of our message key.

● We will add the user's name on his/her message's above.

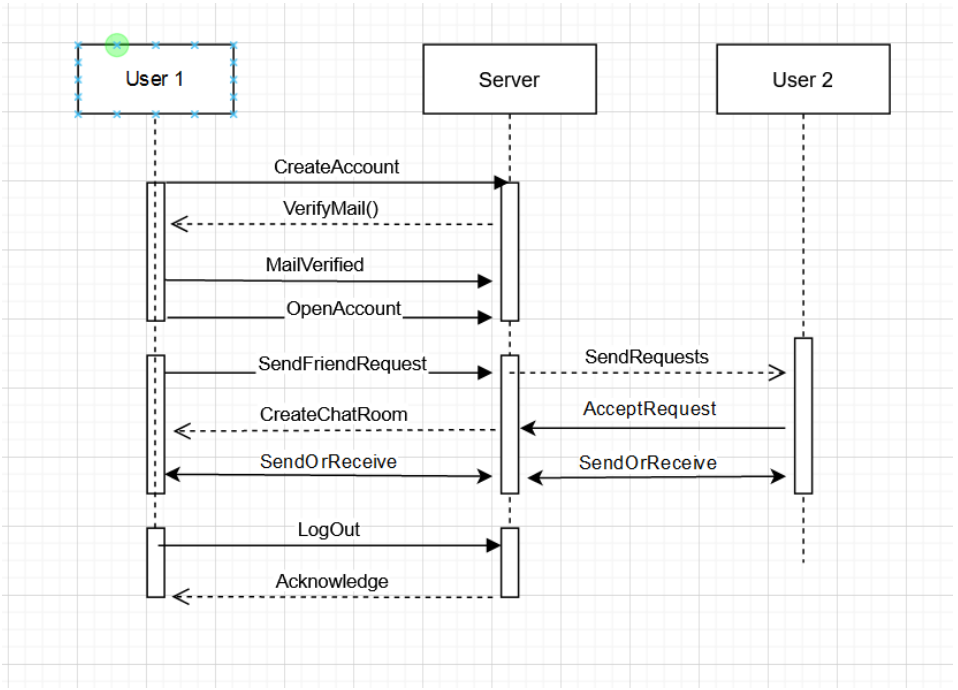
● We will add a search bar among friends.

Maybe friend requests. For now, it's too hard to make some precise predictions about our project at the moment. But we aspire to develop the project with common software development standards and sustainably

About the Security Part, for our message encryption. We are using handwritten AES-128 bit encryption since it's one of the most popular and used encryption methods in which supports 16 bit characters being encrypted into cipher text. But for now we still have a lack of security measures in server side and client side. Apart from the encryption algorithm working as expected, the key we are using for encryption is still in early stages. In order to test the algorithm, we made a temporary key generator. After encryption we send it to another client(‘s) along with cipher text to decrypt. This of course creates a security breach. We are planning some solutions to resolve this issue.

For example, we may create a key distributor service on the server side, giving clients a temporary key to encrypt and decrypt with a small TTL (Time to live) value. Another issue is storing messages in databases. In order to display offline pages we are storing messages in Our mongodb database as mentioned above is plain (not encrypted) text. Our plan for now will be storing all necessary data as encrypted via custom cipher functions from node.js packages.

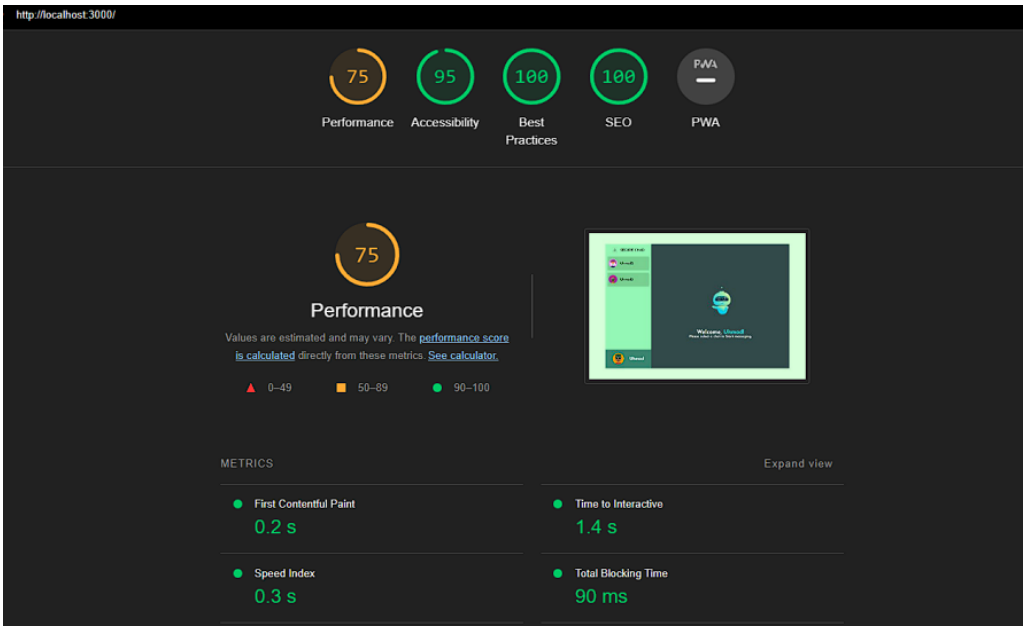
Also our MVP Diagram is like that:

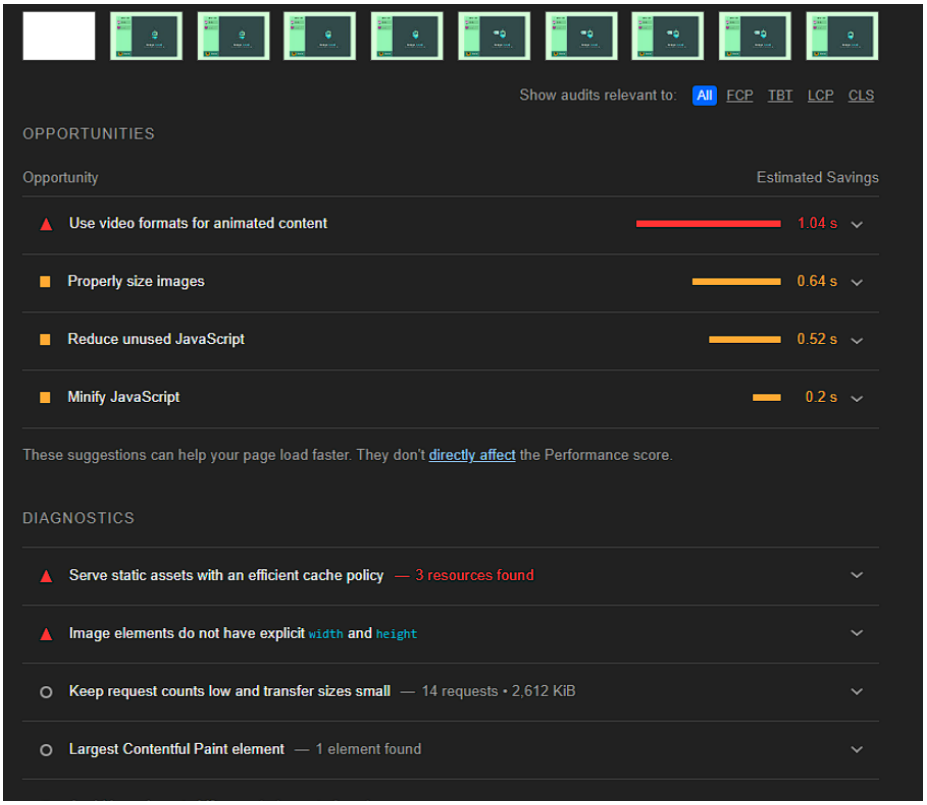


# 5-) Testing Procedures

We have made 2 testing types that go as: functional and non-functional

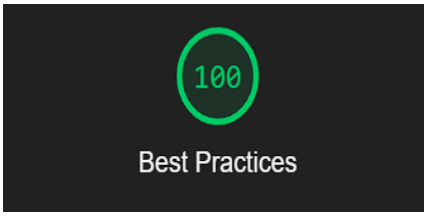
Our non-functional testing was done in Google's platform called “Lighthouse” . This platform recorded our MERN stack app’s performance on various categories and gave an overall score as well as the let downs of our code and gave insight about how to fix said issues

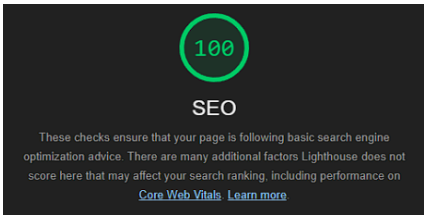


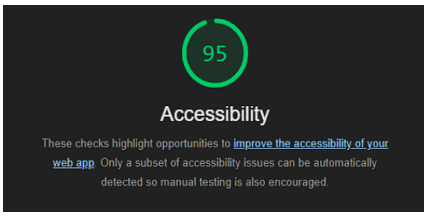



There are 5 categories in Google's metrics and those are:

1.Overall Performance
2.Accessibility
3.Best Practices
4.SEO (Search Engine Optimization Advices)
5.PWA (Progressive Web Application)



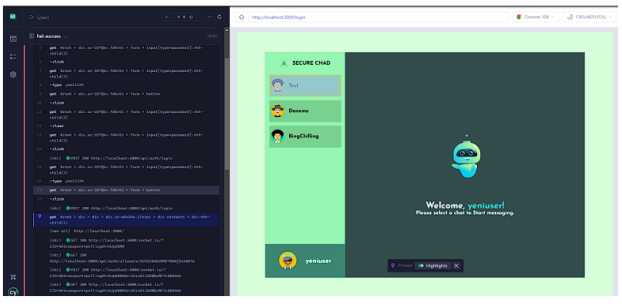



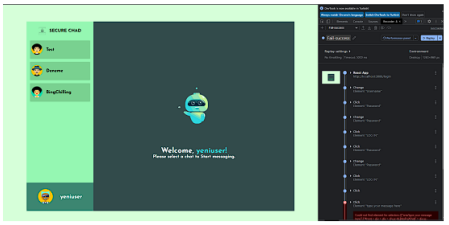



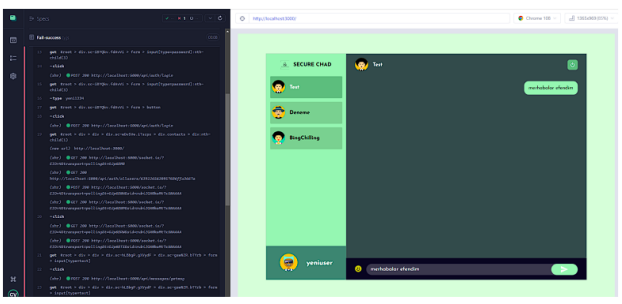


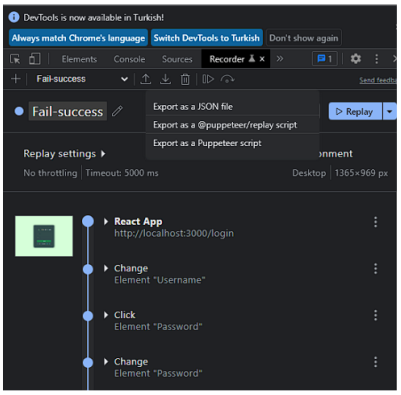
Now comes the part where our functional test begins. First of all we used Google Chrome's Record functionality to record keystrokes and clicks on the webpage so we can export a json file inorder to integrate with the Cypress software we are going to use in order to use json file in Cypress we are also using a package called Cypress-Chrome-Recorder on github this essentially gets the export from the chromes json file and converts the file to cypress format to use it in our testing.

Our test scenario is a user trying to login incorrectly to the program first then correcting themselves after that messaging a user on the platform. In the pictures below you can grasp the concept of our method



# 6-) Target Audience

**Identifying Business Idea**

Our app is trying to be a suitable and desirable platform in the Web Based platforms because our competitors are a little bit behind on the web-based parts of their applications.

It is trying to close the gap between those user's needs.

Average cost for MVP for an average app will be figures like "starting from $20K" and "up to $223K for an app like WhatsApp." Unfortunately, there's no single accurate number.

Real-time chat apps have changed the way we communicate and are changing the way we sell and buy things. Companies building a chat app create a million opportunities and benefits for the employees, partners, clients, and prospects they may not know yet. Telegram's success shows that the messenger's market still has room for new players and ideas.

The Chat Application market revenue was Million USD in 2016, grew to Million USD in 2022, and will reach Million USD in 2029. Considering the influence of COVID-19 on the global Chat Application market, this report analyzed the impact from both global and regional perspectives. From production end to consumption end in regions such as North America, Europe, China and Japan. The report put emphasis on analysis of the market under COVID-19 and corresponding response policy in different regions.

As more organizations continue to focus on specialized consumer bases, the worldwide Chat Application market is becoming increasingly competitive. Since the beginning of the pandemic, most companies have chosen different techniques to regional market conditions to recover from the pandemic. For example, most of the European customers continue to emphasize brands with a strong purpose and high values, whereas in several Asia Pacific economies, there has been a fundamental change away from critical items. The reports analyze the company activities, SWOT analysis and economic profile of Chat Application Industry.

Based on TYPE, the Chat Application market from 2022 to 2029 is primarily split into:

Cloud-Based, On-Premises.

**WhatsApp:** The most popular messaging app in the world, with over five billion downloads and two billion active users

**Facebook Messenger:** Facebook’s own messaging app isn’t too shabby either, reaching five billion downloads a year after WhatsApp

**Snapchat:** Snapchat kickstarter photo sharing and has since branched out its platform into a fully fledged social network

**Telegram:** Popular alternative messaging app in Central Asia and the Middle East, known for strong pro- privacy stance

**WeChat:** China's messaging super-app, with more than one billion users in the country and thousands of mini-programs

**LINE:** Popular in Indonesia, Japan, Taiwan, and Thailand. Part of Naver and SoftBank's tech conglomerate, Z Holdings

**iMessage:** Apple's messaging platform has about one billion users, although how many use it as the primary platform is unknown

**Discord:** Originally built as a communications platform for gamers, Discord has branched out into a whole host of other communities

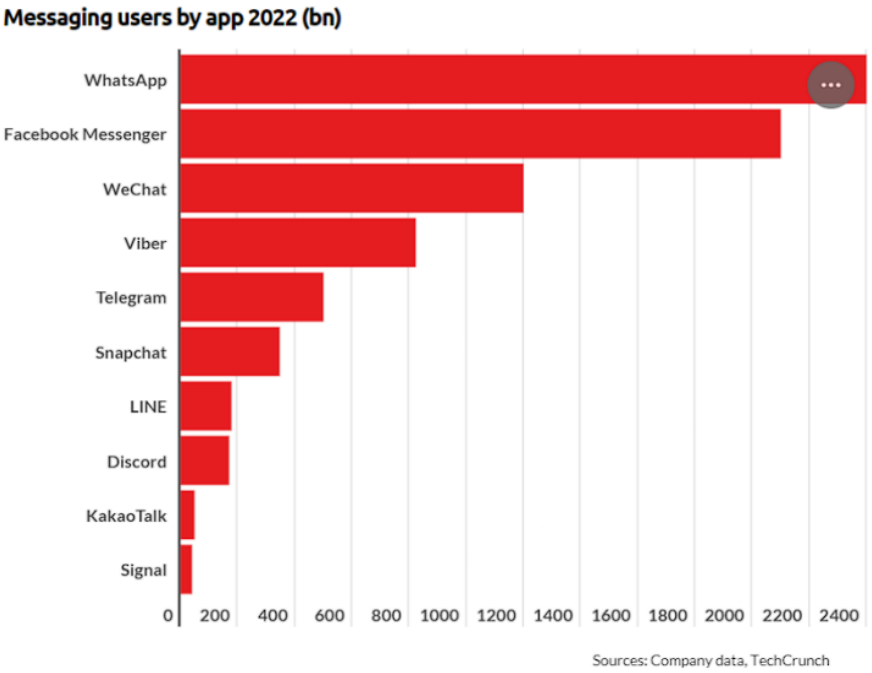
**Signal:** Hyper-focused on privacy and security, Signal has built a loyal following of disgruntled ex WhatsApp and Facebook users

**Viber:** Another messaging app that has been able to target emerging markets, such as Eastern Europe and the Middle East

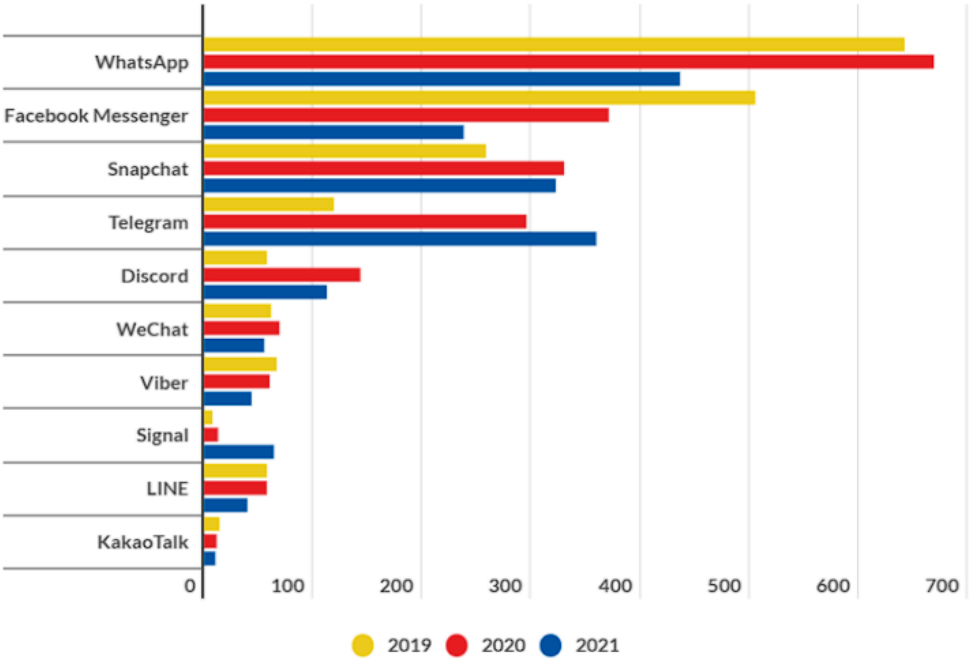
**KakaoTalk:** South Korea's homegrown messaging app, which is also the most popular in the country, with over 40 million active users

The dominance of Facebook and WhatsApp is even more apparent in India, with 92% of all sessions on these two apps. Penetration of Telegram and Signal, which saw its usage skyrocket during WhatsApp's privacy update, has not made much of a dent in total sessions.

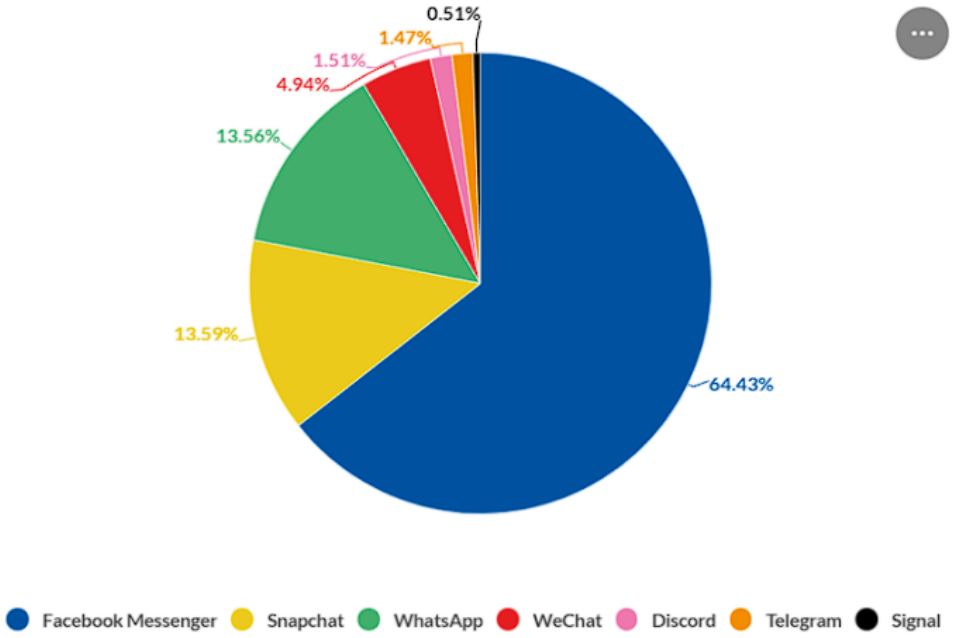
And here is Market Research of the Chatting apps in visual graphics:



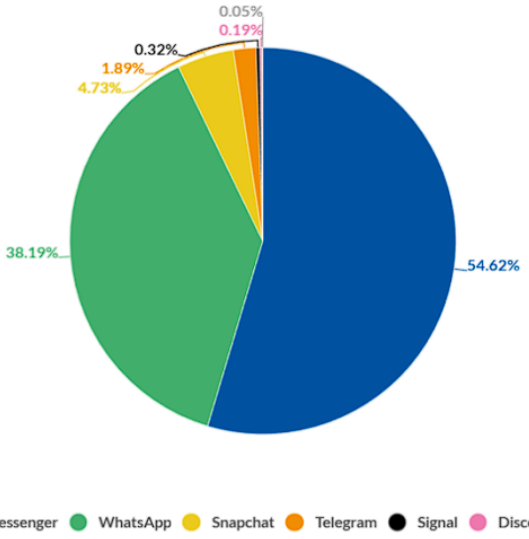
## Messaging app downloads 2019 to 2021 (mm)

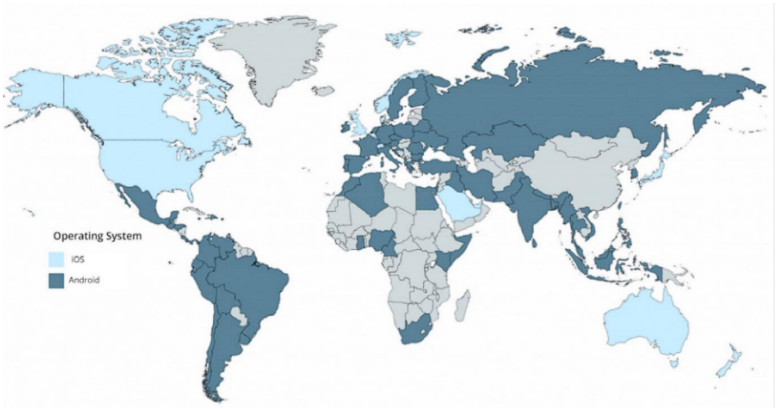



## Messaging app marketshare by usage in the United States 2021 (bn)

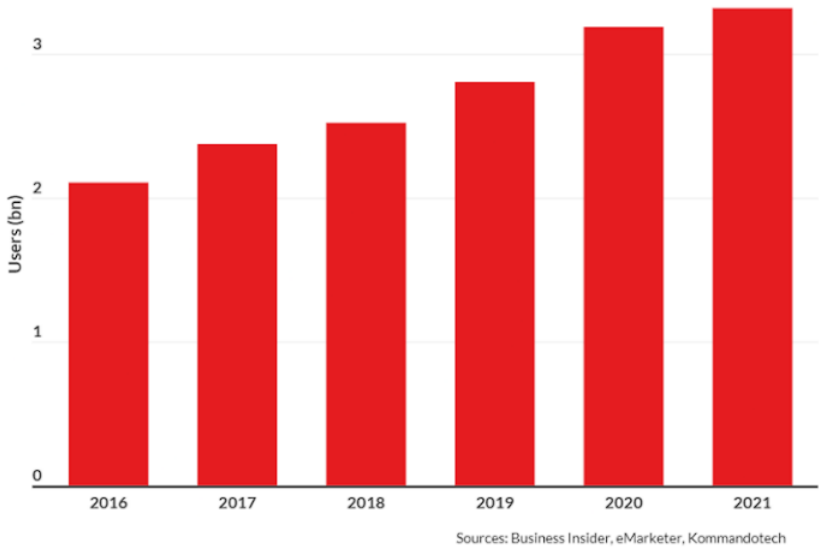


**Messaging app marketshare by usage in India 2021 (bn)**




**Messaging app annual users 2016 to 2021 (bn)**





Emrecan Üzüm **1900005485**

Şükrü Erim Sinal **1900003587**

Ahmet Kaan Memioğlu **1900005528**