

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR
(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)



Project Report

on

EmoChat

Submitted By:

Pragati Bansal

0901CS191080

Faculty Mentor:

Prof. Mir Shahnawaz Ahmad

Assistant Professor, Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE

MAY-JUNE 2022

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR

(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)



Project Report

on

EmoChat

A project report submitted in partial fulfilment of the requirement for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by:

Pragati Bansal

0901CS191080

Faculty Mentor:

Prof. Mir Shahnawaz Ahmad

Assistant Professor, Computer Science and Engineering

Submitted to:

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE

GWALIOR - 474005 (MP) est. 1957

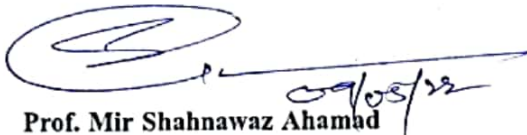
MAY-JUNE 2022

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR

(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)

CERTIFICATE

This is certified that **Pragati Bansal** (0901CS191080) has submitted the project report titled **EmoChat** under the mentorship of Mr. Mir Shahnawaz Ahmad, in partial fulfilment of the requirement for the award of degree of Bachelor of Technology in Computer Science and Engineering from Madhav Institute of Technology and Science, Gwalior.



Prof. Mir Shahnawaz Ahmad
Faculty Mentor
Assistant Professor
Computer Science and Engineering



Dr. Manish Dixit
Professor and Head
Computer Science and Engineering

Dr. Manish Dixit
Professor & HOD
Department of CSE
M.I.T.S. Gwalior

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR

(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)

DECLARATION

We hereby declare that the work being presented in this project report, for the partial fulfilment of requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering at Madhav Institute of Technology & Science, Gwalior is an authenticated and original record of my work under the mentorship of **Prof. Mir Shahnawaz Ahmad**, Assistant Professor, Computer Science and Engineering.

We declare that we have not submitted the matter embodied in this report for the award of any degree or diploma anywhere else.



Pragati Bansal

0901CS191080

III Year

Computer Science and Engineering

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR

(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)

ACKNOWLEDGEMENT

The full semester project has proved to be pivotal to my career. I am thankful to my institute, **Madhav Institute of Technology and Science**, for allowing me to continue my disciplinary project as a curriculum requirement, under the provisions of the Flexible Curriculum Scheme (based on the AICTE Model Curriculum 2018), approved by the Academic Council of the institute. I extend my gratitude to the Director of the institute, **Dr. R. K. Pandit** and Dean Academics, **Dr. Manjaree Pandit** for this.

I would sincerely like to thank my department, **Department of Computer Science and Engineering**, for allowing me to explore this project. I humbly thank **Dr. Manish Dixit**, Professor and Head, Department of Computer Science and Engineering, for his continued support during the course of this engagement, which eased the process and formalities involved.

I am sincerely thankful to my faculty mentors. I am grateful to the guidance of **Prof. Mir Shahnawaz Ahmad**, Assistant Professor, Computer Science and Engineering for their continued support and guidance throughout the project. I am also very thankful to the faculty and staff of the department.


Pragati Bansal

0901CS191080

III Year

Computer Science and Engineering

Abstract

While talking to a person, it's not just their words that make an impact in the way we perceive their point of view but body language also plays an important role. As the world is moving towards more virtual meetings and chats in particular, it becomes very easy to misinterpret someone's emotions if we can't see them. A simple solution for this problem is to just video conference the person whom we want to talk to. But hey, it's not always possible and we can't ask someone randomly for a video call.

We are building this app with the aim in mind to reduce this miscommunication/misinterpretation. Here we are taking the help of machine learning to predict what the chat message might want to convey using a sentiment score along with a relevant mood emoji. The receiver will be able to see the message which the sender wants to send along with these two elements which our deep learning model will add to the message, making it more meaningful and less easy to misinterpret.

Keywords: Chats, Misinterpretation, Deep Learning Model, Sentiment, Meaningful.

सार:

किसी व्यक्ति से बात करते समय, यह न केवल उनके शब्द हैं जो उनके दृष्टिकोण को समझने के तरीके को प्रभावित करते हैं, बल्कि बॉडी लैंग्वेज भी एक महत्वपूर्ण भूमिका निभाती है। जैसे-जैसे दुनिया अधिक आभासी बैठकों और विशेष रूप से चैट की ओर बढ़ रही है, अगर हम किसी की भावनाओं को नहीं देख सकते हैं तो उनकी गलत व्याख्या करना बहुत आसान हो जाता है। इस समस्या का एक सरल उपाय है कि हम जिस व्यक्ति से बात करना चाहते हैं, उसकी सिर्फ वीडियो कॉन्फ्रेंसिंग करें। लेकिन हे, यह हमेशा संभव नहीं होता है और हम किसी से वीडियो कॉल के लिए बेतरतीब ढंग से नहीं पूछ सकते हैं।

हम इस गलत संचार / गलत व्याख्या को कम करने के उद्देश्य से इस ऐप का निर्माण कर रहे हैं। यहां हम यह अनुमान लगाने के लिए मशीन लर्निंग की मदद ले रहे हैं कि एक प्रासंगिक मूड इमोजी के साथ एक सेंटिमेंट स्कोर का उपयोग करके चैट संदेश क्या संदेश देना चाहेगा। प्राप्तकर्ता उस संदेश को देख सकेगा जिस प्रेषक इन दो तत्वों के साथ भेजना चाहता है जिसे हमारा गहन शिक्षण मॉडल संदेश में जोड़ देगा, जिससे यह अधिक सार्थक और गलत व्याख्या करने में कम आसान हो जाएगा।

INDEX

Abstract	IV
संक्षेपः	V
List of figures	VII
Chapter 1: Introduction	1
Chapter 2: Steps to Run	2
Chapter 3: Tech Stack	3
1.1 Front-end	3
1.1.1 React Native	3
1.1.2 Expo	3
1.1.3 Android Emulator	3
1.2 Back-end	4
1.2.1 Firebase	4
1.2.2 ML Model	6
1.2.2.1 Sentiment Categories	7
1.2.2.2 Accuracy vs Epochs and Loss vs Epochs graph	7
1.2.2.3 Model Saving	8
1.2.2.4 Model Simulation	8
Chapter 4: WorkFlow	10
4.1 App creation	10
4.2 Setting up Firebase	11
4.3 Adding components and Screens	11
4.3.1 Components include:	11
4.3.2 Screens include:	11
4.4 Running expo cli	11
4.5 Adding ML (NLP) Model	12
4.6 Final App	13
Chapter 5: Functionalities	15
5.1 App	15
5.2 UI	15
5.3 Chats	16
5.3.1 Live Sentiment Prediction	16
5.3.2 Emoji Support	17
5.3.3 Images Support	17
5.3.4 Time Stamp	17

Chapter 6: Challenges	18
6.0.1 Integrating ML Model with React native	18
6.0.2 Adding Time Stamp	18
Chapter 7: Conclusion and Future Work	19
7.1 Conclusion	19
7.2 Future Work	19
7.2.1 Weekly/Monthly Mood Indicator	19
7.2.2 More Functionalities within the app	19
References	20

LIST OF FIGURES

Figure Number	Figure Caption	Page no.
1.1	Message with Sentiment Elements	1
3.1	Tech Stack Mindmap	3
3.2	Android Emulator	4
3.3	Firebase Authentication	5
3.4	Firebase Storage	5
3.5	Firebase Profile Picture Update	6
3.6	Firebase room information	6
3.7	ML Model Sentiment Categories	7
3.8	Mapping classes to index	7
3.9	Epoch Accuracy and Loss Graphs	8
3.10	Saving ML Model	8
3.11	ML Model results	9
4.1	WorkFlow Diagram	10
4.2	Loading Dataset	12
4.3	ML Model - Tokenization	12
4.4	ML model - Padding sequences	12
4.5	Model Training Epochs	13
4.6	Final App chat prediction	14
5.1	Functionalities Mindmap	15
5.2	App Login Screen	15
5.3	App profile pic upload screen	16
5.4	Live sentiment prediction	17
5.5	Emoji Support	17
5.6	Images Support	17
5.7	Timestamp	17

CHAPTER 1: INTRODUCTION

1.1 EmoChat is a mobile application that has the ability to show live chat sentiments (emotions). Every time when a user sends or receives a message, the sentiment that is reflected by that particular message will be displayed next to that message in real-time.

The sentiment part of the chat message has two parts:

- **Sentiment score** - A value between -10 to 10, representing the polarity of the sentiments. 10 reflects the most positive sentiment whereas -10 is indicated the most negative sentiment.
- **An emoji** - An emoji depending on the sentiment value is also displayed, which essentially represents the possible mood of the sender while we receive their message.

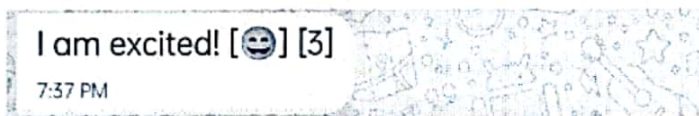


Fig 1.1 Message with Sentiment Elements

CHAPTER 2: STEPS TO RUN

2.2 For running the app, we have to first download the code file (if in zip format, just extract it).

- Open terminal in that directory (either dedicated or your fav. code editor's terminal) and run the following commands
 - *npm install*
 - *npm install --global expo-cli*
- Now let the processes run and install required files.
- After that, we can simply run the app using the command
 - *npm start*
- This will either redirect us to a browser window or give an url which can be copied and pasted to open a browser window.
- In that windows, we will find a QR code and some links to run the app. Make sure to navigate the selected option to **"tunnel"** instead of "LAN" to enable stable connection. App might not run on the "LAN" option.
- We have to download the expo client app - **"Expo Go"** from the playstore in order to scan and run that app on our smartphone or any android emulator for that matter.
- After scanning, let it download the contents and the app will be ready to rock!

Chapter 3: TECH STACK

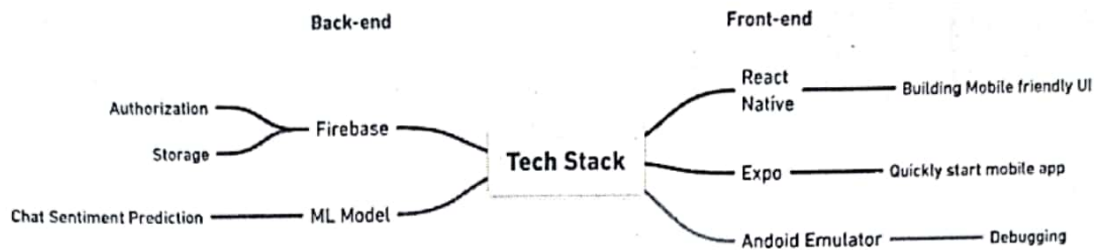


Fig 3.1 Tech Stack Mindmap

The tech stacks used here are:

3.1 Front-end

3.1.1 React Native

React Native is an open-source framework that enables cross-platform mobile application building using JavaScript and React, which is an open-source JavaScript library.

React Native uses JSX to create native apps for Android as well as iOS. It is a powerful tool that aids in creating apps with beautiful UIs, which are the same as native apps. It makes mobile app creation very easy and efficient, that is why I chose this front-end framework for my app.

3.1.2 Expo

Expo is a React Native app development framework. It's a collection of React Native-specific tools and services. It will make it simple for us to start creating React Native apps.

It gives us a list of tools to make creating and testing React Native apps easier. Aside from that, Expo offers a more robust and simple development approach that is also more flexible.

3.1.3 Android Emulator

It is a tool that I used to see the app in real-time in order to debug it and add elements to it. Android emulator can be found in Android Studio.

Although I also used my physical mobile phone to run and see the application apart from the emulator.

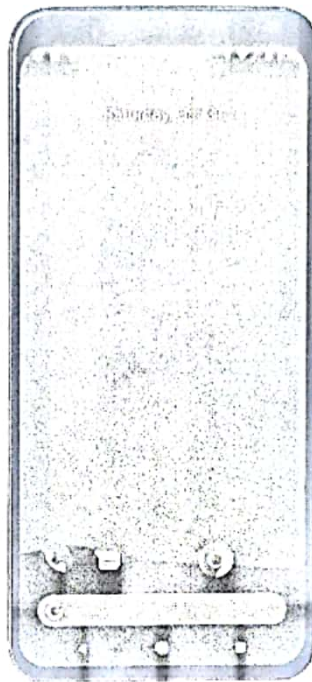


Fig 3.2 Android Emulator

3.2 Back-end

3.2.1 Firebase

Firebase is a Google-backed mobile and web app development platform based on the backend as a service (BaaS) system and consists of several pragmatic services and purposeful APIs to develop high-quality applications.

Firebase provides swift and secure hosting for applications. With trusted Google authentication and stability, It was easy for me to choose firebase for **data storage** (for chats and images) and **authentication.**'

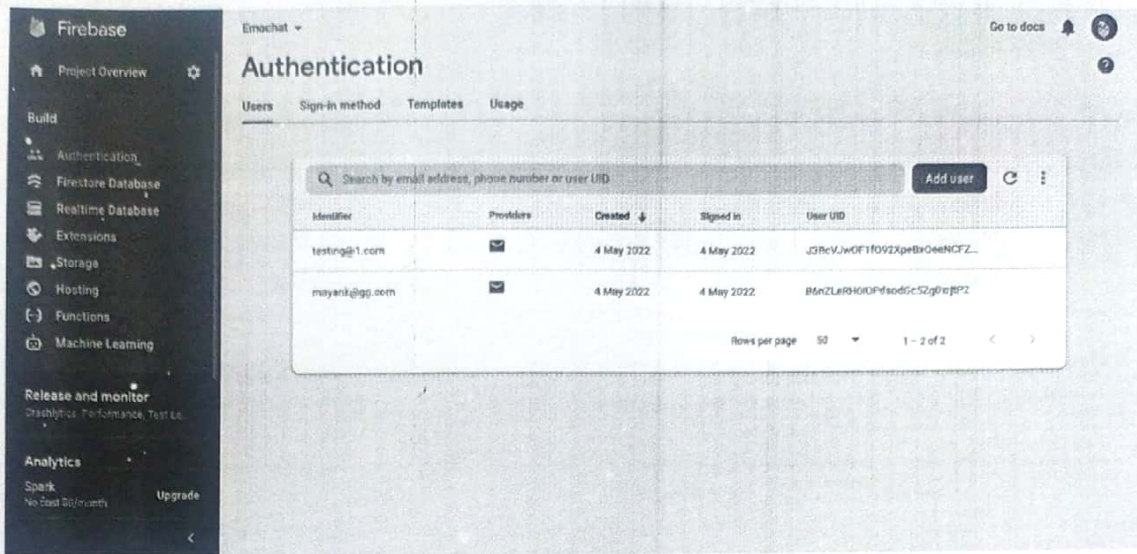


Fig 3.3 Firebase Authentication

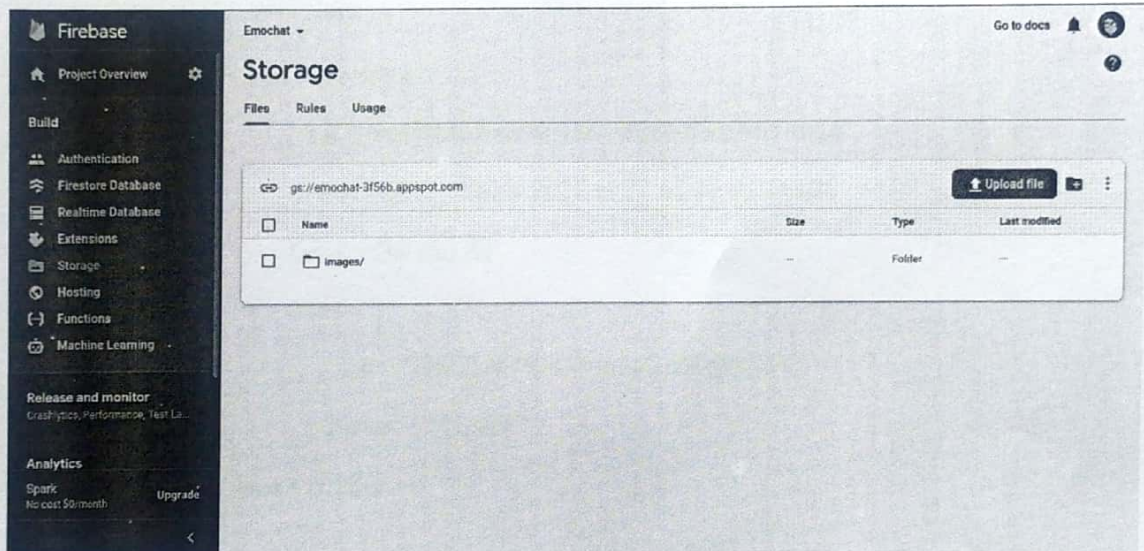


Fig 3.4 Firebase Storage

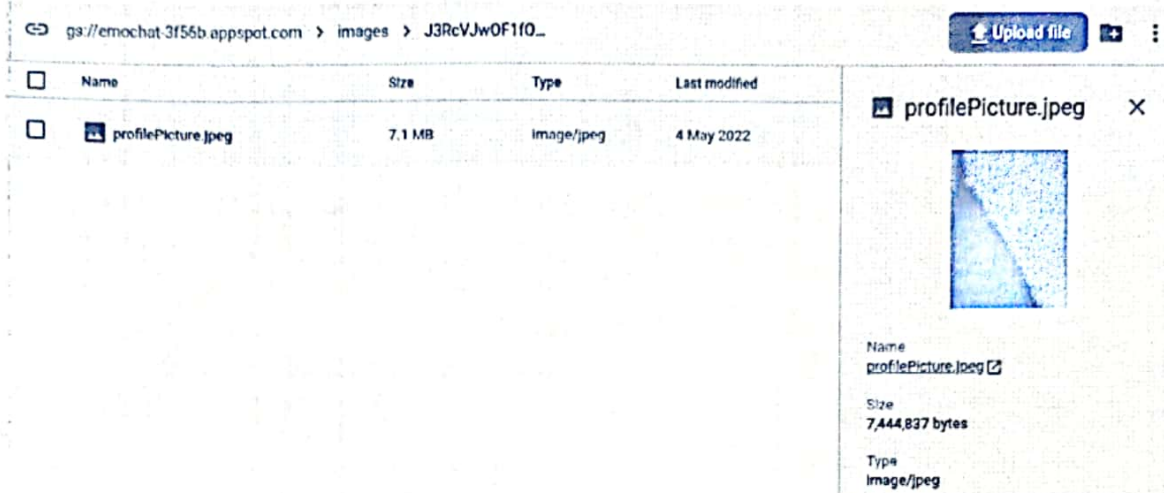


Fig 3.5 Firebase Profile Picture Update

```

+ Add field
- lastMessage
  _id: "6c98d387-e399-4483-86c0-d6c53bf73f55"
  createdAt: 4 May 2022 at 22:45:06 UTC+5:30
  text: "hii 😊 [0]"
- user
  _id: "B6nZLeRH0IOPdsodGc5Zg0xrjtP2"
  name: "Mayank"
- participants

```

Fig 3.6 Firebase room information

3.2.2 ML Model

In order to predict live sentiments in chat messages, I prepared an ML model.

I started with a machine learning model but they were not giving that much accuracy (around 50%), so I used **Deep learning** for sentiment prediction using NLP techniques. The deep learning model was giving an accuracy of over **90%**.

3.2.2.1 Sentiment Categories

The sentiment analysis is divided into 6 categories - sadness, anger, love, surprise, fear and joy. It means that whenever any word or sentence will be put in this model, it will categorize it in any one of the above mentioned categories.

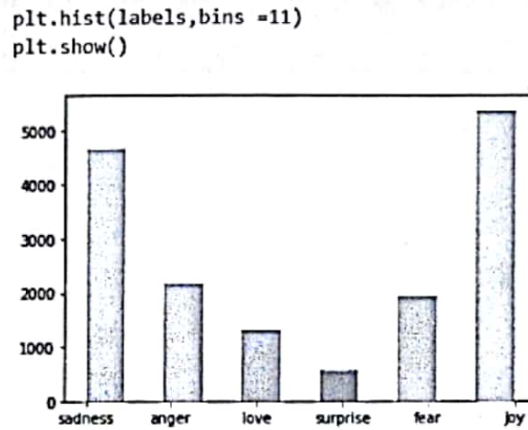


Fig 3.7 ML Model Sentiment Categories

```
class_to_index
```

```
{'anger': 0, 'fear': 3, 'joy': 4, 'love': 2, 'sadness': 5, 'surprise': 1}
```

Fig 3.8 Mapping classes to index

3.2.2.2 Accuracy vs Epochs and Loss vs Epochs graph

The graph of epochs vs training and validation accuracy is present on the left and the graph at the right is between epoch vs training and validation loss.

The number of epochs used were 20.

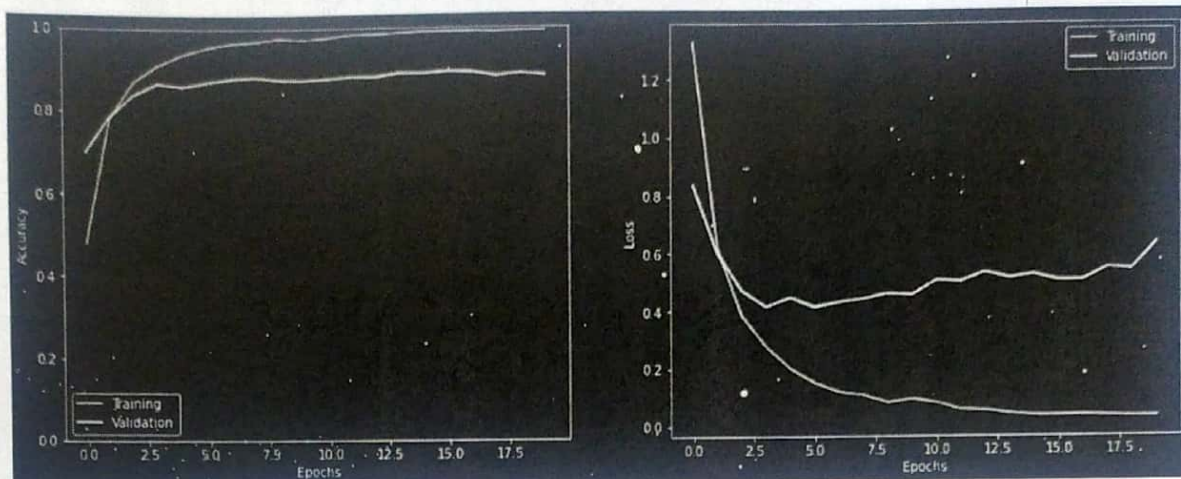


Fig 3.9 Epoch Accuracy and Loss Graphs

Final Accuracy - over 98%

3.2.2.3 Model Saving

After training the model, It needs to be saved in this state in order to use it for sentiment prediction.

```

model.save("EmoChat")

import pickle
with open('tokenizer.pickle','wb') as handle:
    pickle.dump(tokenizer,handle,protocol = pickle.HIGHEST_PROTOCOL)

with open('label_encoder.pickle','wb') as ecn_file:
    pickle.dump(index_to_class,ecn_file,protocol = pickle.HIGHEST_PROTOCOL)

```

Fig 3.10 Saving ML Model

3.2.2.4 Model Simulation

After saving, here is the live simulation of the sentiment prediction from sentences.

```
txt = ""
while(txt != "exit"):
    txt = input("Enter chat : ")
    print("Emotion : ", predicting(txt))
```

```
Enter chat : hurray! You won!
Emotion : anger
Enter chat : good
Emotion : joy
Enter chat : alas!
Emotion : anger
Enter chat : oh!
Emotion : joy
Enter chat : congratulations!
Emotion : joy
Enter chat : damn it!
Emotion : anger
Enter chat : I am sad
Emotion : sadness
```

Fig 3.11 ML Model results

Google Colab Link: [Link](#)

CHAPTER 4: WORKFLOW

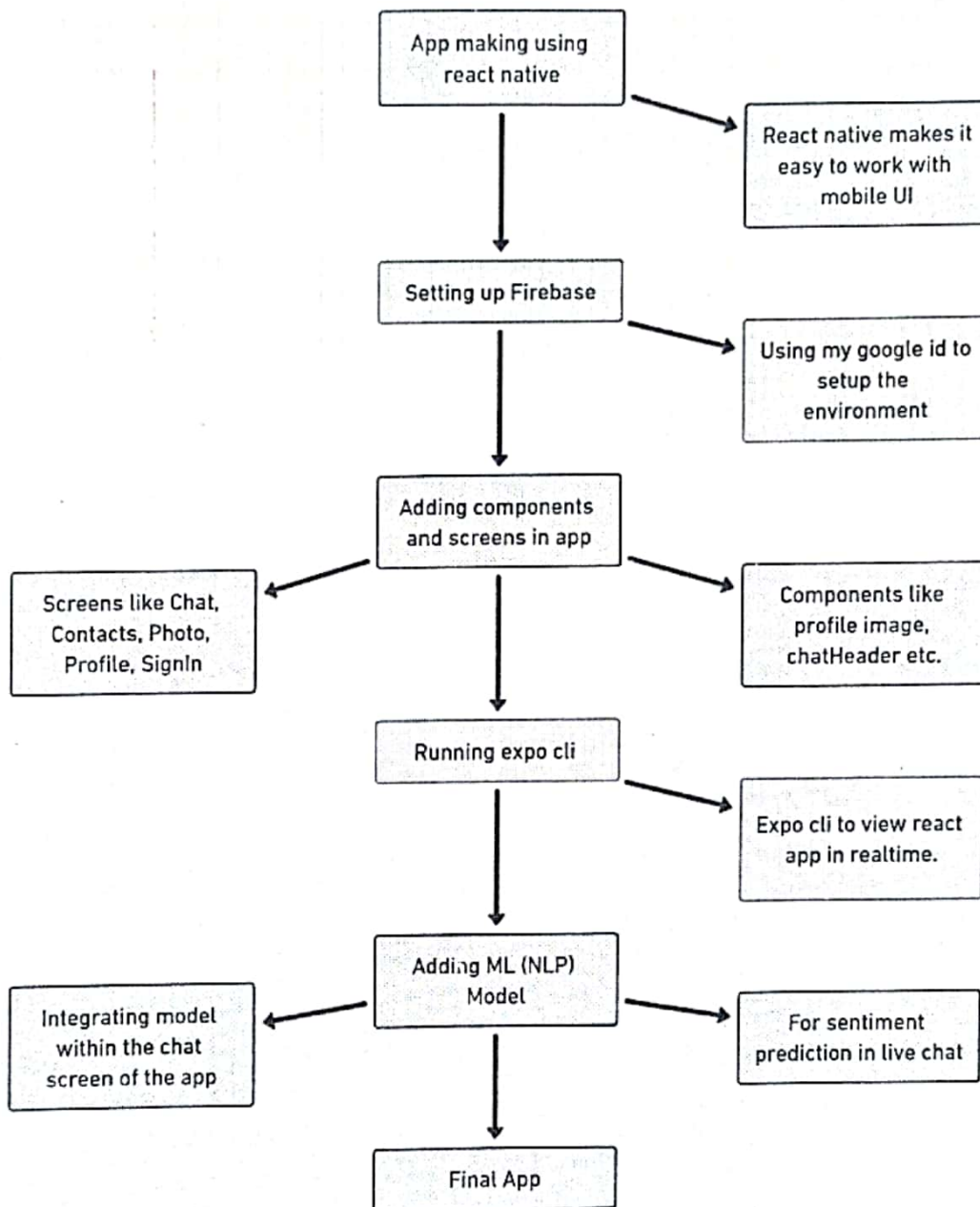


Fig 4.1 WorkFlow Diagram

4.1 App creation

App was created using React native. For this we have to set up node modules and other files using `npm install -g create-react-native-app`.

Initiated work on the app till the login page before setting up firebase for authentication.

4.2 Setting up Firebase

Next step was to set up authentication for the app in firebase, in order to make the app login and signup process hassle-free.

4.3 Adding components and Screens

Then I moved on to adding various components and screens of the app.

4.3.1 Components include:

- Avatar.js
- ChatHeader.js
- ContactsFloatingIcon
- ListItem

4.3.2 Screens include:

- Chat.js
 - Individual chat screen for a separate room. Room (in database) signifies one-to-one conversation between two users.
- Chats.js
 - This screen is basically the collection of rooms as it shows all the users who you chatted with recently.
- Contacts.js
 - This screen shows all the list of contacts that have been registered in the app's database and are present in our contact list.
- Photo.js
 - This screen comes up everytime we upload or send a photo within the app.
- Profile.js
 - This screen is the next screen after Signup/Login. It requires a name and an optional profile picture when the user Signs in for the first time.
- SignIn.js
 - This is the first screen with which a new user will interact with.
 - Ofcourse, an already existing user won't meet this screen as the app will automatically recognize him/her as an existing user.

4.4 Running expo cli

Expo cli is used hand-in-hand with react native in order to see the app live for debugging.

For this, we have to first run the react native app using `npm start` which in turn runs expo cli interface having a QR code and all the links to run the app on various screens and Operating systems.

4.5 Adding ML (NLP) Model

Uptill now, I have pretty much wrapped up the app UI part. Now I need to add the NLP layer over live chats to show relevant **sentiment score** and **mood emoji** with the msgs themself.

First step is to train the model. For this - these steps must be followed:

- We need a dataset, fortunately, an emotion dataset is available in the nlp library.

```
dataset = nlp.load_dataset('emotion')
```

Fig 4.2 Loading Dataset

- For feeding the model with the dataset, I processed it using Tokenization, word to sequence conversion etc.

```
tokenizer = Tokenizer(num_words=10000, oov_token = '<UNK>')  
tokenizer.fit_on_texts(tweets)
```

```
tokenizer.texts_to_sequences([tweets[0]])
```

Fig 4.3 ML Model - Tokenization

- After tokenization, padding should be done which ensures the sentence length of the same size for feeding the model.

```
maxlen = 50  
from tensorflow.keras.preprocessing.sequence import pad_sequences  
def get_sequences(tokenizer,tweets):  
    sequence = tokenizer.texts_to_sequences(tweets)  
    padded = pad_sequences(sequence, truncating = 'post', padding = 'post', maxlen = maxlen)  
    return padded  
padded_train_seq = get_sequences(tokenizer,tweets)
```

Fig 4.4 ML model - Padding sequences

- The main step comes to actually run the model training. For this I found optimum accuracy to be coming at 20 epochs.

```

h = model.fit(
    padded_train_seq, train_labels,
    validation_data = (val_seq, val_labels),
    epochs = 20,
)

```

```

Epoch 1/20
500/500 [=====] - 33s 49ms/step - loss: 1.3253 - accuracy: 0.4790 - val_loss: 0.8356 - val_accuracy: 0.7000
Epoch 2/20
500/500 [=====] - 22s 44ms/step - loss: 0.6200 - accuracy: 0.7796 - val_loss: 0.6015 - val_accuracy: 0.7855
Epoch 3/20
500/500 [=====] - 22s 45ms/step - loss: 0.3848 - accuracy: 0.8691 - val_loss: 0.4712 - val_accuracy: 0.8385
Epoch 4/20
500/500 [=====] - 25s 49ms/step - loss: 0.2766 - accuracy: 0.9095 - val_loss: 0.4150 - val_accuracy: 0.8665
Epoch 5/20
500/500 [=====] - 23s 46ms/step - loss: 0.2000 - accuracy: 0.9359 - val_loss: 0.4464 - val_accuracy: 0.8590
Epoch 6/20
500/500 [=====] - 23s 46ms/step - loss: 0.1502 - accuracy: 0.9531 - val_loss: 0.4143 - val_accuracy: 0.8700
Epoch 7/20
500/500 [=====] - 23s 45ms/step - loss: 0.1181 - accuracy: 0.9638 - val_loss: 0.4307 - val_accuracy: 0.8790
Epoch 8/20
500/500 [=====] - 23s 46ms/step - loss: 0.1062 - accuracy: 0.9671 - val_loss: 0.4420 - val_accuracy: 0.8820
Epoch 9/20
500/500 [=====] - 22s 45ms/step - loss: 0.0802 - accuracy: 0.9753 - val_loss: 0.4598 - val_accuracy: 0.8755
Epoch 10/20
500/500 [=====] - 27s 54ms/step - loss: 0.0923 - accuracy: 0.9715 - val_loss: 0.4553 - val_accuracy: 0.8740
Epoch 11/20
500/500 [=====] - 22s 45ms/step - loss: 0.0792 - accuracy: 0.9759 - val_loss: 0.5051 - val_accuracy: 0.8775

```

Fig 4.5 Model Training Epochs

After training, I saved the model using inbuilt python methods and then integrated it with my native app UI.

4.6 Final App

After completing every subpart in the workflow, the final App is ready in action!

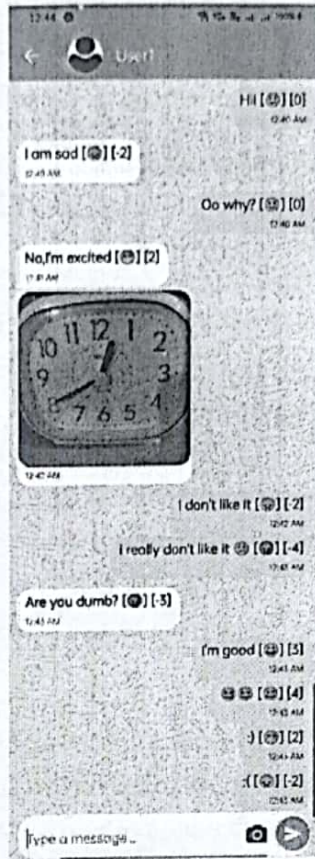


Fig 4.6 Final App chat prediction

CHAPTER 5: FUNCTIONALITIES



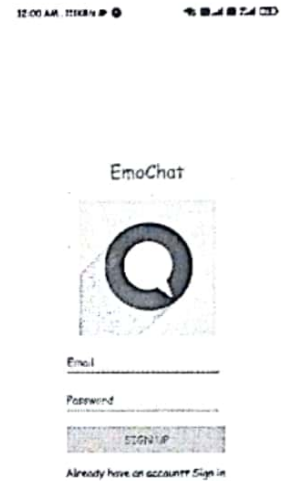
Fig 5.1 Functionalities Mindmap

5.1 App

The application requires mandatory user registration in order to access it. If a user is visiting the app for the first time, he/she needs to register himself/herself on the app and can use that id and password for logging in the next time.

This functionality is implemented using firebase, which verifies the entered user id and password with its database. If it is matched with an entry in the database, then it allows the user to access the application.

Fig 5.2 App Login Screen



5.2 UI

In the Application UI, users can also add their profile pictures in order to make it more intuitive and not that simple. It also makes it easy to identify a chat without having to see the name.

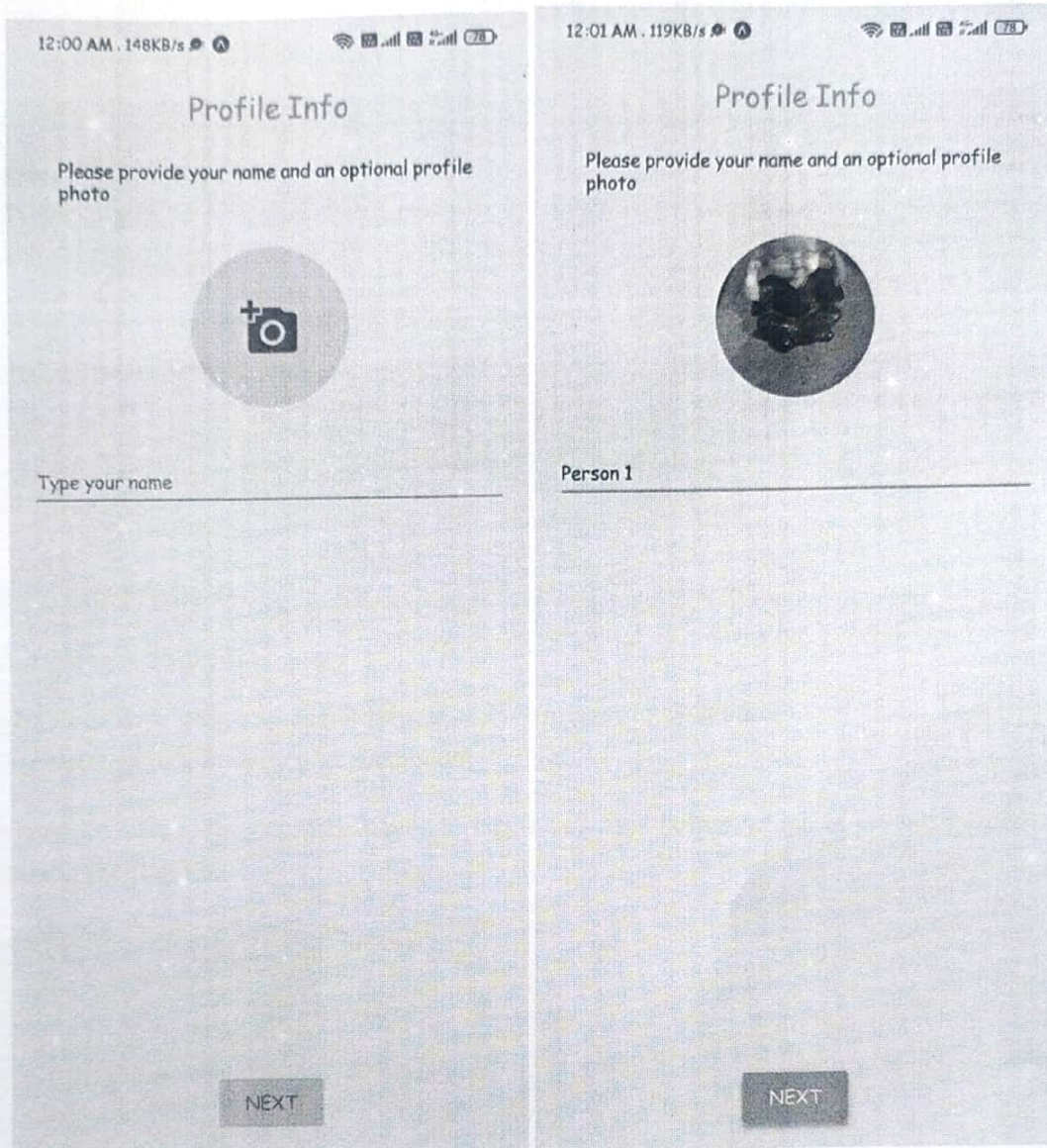


Fig 5.3 App profile pic upload screen

5.3 Chats

5.3.1 Live Sentiment Prediction

It is the most unique feature of this application. We have used chatting apps like WhatsApp, Facebook, Instagram, and whatnot. But my application can show in real-time the sentiment which is reflected by each message. It makes it very easy **not-to-misinterpret** the sense of the message and the user can then reply accordingly.

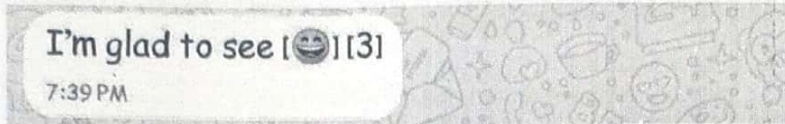


Fig 5.4 Live sentiment prediction

5.3.2 Emoji Support

This app also supports the use of emojis. Some apps show emojis as garbage texts but this app shows them as relevant emojis.

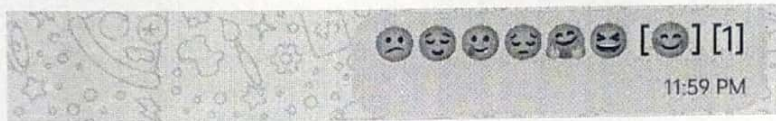


Fig 5.5 Emoji Support

5.3.3 Images Support

Apart from text, users can also send and receive images to each other in the chats. Note that images don't play any role in sentiment prediction.

To send an image, the user just needs to tap the camera button which is present before the send button in the chat UI.

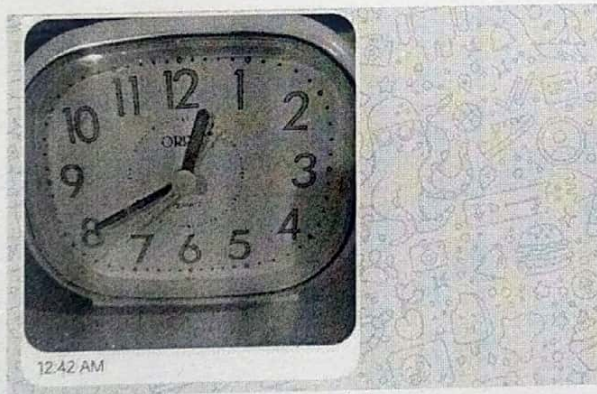


Fig 5.6 Images Support

5.3.4 Time Stamp

The messages which are sent and received in the chat UI also carry their time of sending along with them, which makes the chat more informative.



Fig 5.7 Timestamp

CHAPTER 6: CHALLENGES

6.1 Integrating ML Model with React native

Integrating the ML NLP Model was a challenge as it was not easy to create and manipulate the API of the ML model.

I had to first save the trained model and then Its API can be used in the parsing of chat messages.

To apply the model, I took the last message from the database and passed it to the ML model, which gave a value (a number) as an output. I then used that number to show relevant emojis along with the sentiment value using if-else statements.

6.2 Adding Time Stamp

To add this, I have to grab the time of sending the message from the database using `getTime()` function and then manipulate the received value to show just below the sent and received message.

CHAPTER 7: CONCLUSION AND FUTURE WORK

7.1 Conclusion

We have made an application using react native which can predict and display the emotion with a score in real time along with the text. It supports emoji, images. It runs through expo client and has its authentication and storage on Firebase.

7.2 Future Work

7.2.1 Weekly/Monthly Mood Indicator

We can use the collected chat data for a week or month to make an overall sentiment analysis of the user. This might help in detecting **psychological stress** he/she might be going through.

- For this we need to add an export data button within the app to export those chats and then this data can be analysed by our ml model.
- We can also show the results within the app in order to make this feature more usable.

7.2.2 More Functionalities within the app

More functionalities like

- ability to delete chats,
- read receipts,
- voice message support

can also be added to the app.

References

- I. [React Native · Learn once, write anywhere](#)
- II. [Expo CLI - Expo Documentation - Expo Documentation](#)
- III. [How to connect ML model which is made in python to react native app - Stack Overflow](#)
- IV. [Deploy Machine Learning Model using Flask - GeeksforGeeks](#)
- V. Book-Machine Learning for face,emotion and pain recognitiion