

A Project-II Report
on
**AUTOMATED CHATBOT FOR LANGUAGE TRANSLATION
AND RECOGNITION**

Submitted in partial fulfillment of the requirement
For
The award of degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE & ENGINEERING
2018-2022

Submitted By
GUNDAPU NITISH KUMAR 18311A0519

Under the Guidance of
DR. HALAVATH BALAJI
Associate Professor



Department of Computer Science & Engineering
SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY
(AUTONOMOUS)
Yamnampet (V), Ghatkesar (M), Hyderabad – 501301, A.P.

AY-2021-2022

Department of computer Science and Engineering
Sreenidhi Institute of Science and Technology



CERTIFICATE

This is to certify that this Project-II report on "**AUTOMATED CHATBOT FOR LANGUAGE TRANSLATION AND RECOGNITION**", submitted by **GUNDAPU NITISH KUMAR (18311A0519)** in the year 2022 in the partial fulfillment of the academic requirements of Jawaharlal Nehru Technological University for the award of the degree of Bachelor of Technology in Computer Science and Engineering, is a bonafide work that has been carried out by them as a part of their **Project-II during Fourth Year Second Semester**, under our guidance. This report has not been submitted to any other Institute or university for the award of any degree.

Internal guide	Project Coordinator	Head of the Department
Dr. Halavath Balaji Associate Professor Department of CSE	Mrs. Bejjam Vasundhara Devi Assistant Professor Department of CSE	Dr. Aruna Varanasi Professor & HOD Department of CSE

Signature of the External Examiner
Date: -

Lenskart Solutions Pvt. Ltd.

12/1, 7th Floor, Vatika Mindscapes, National Highway 2
Sector 27D, Faridabad, Haryana - 121003 | www.lenskart.com
CIN: U51494DL2008PTC178355



Feb 28, 2022

Gundapu Nitish Kumar

Subject: Letter of Engagement for Retainership

This has reference to your proposal of offering your services and expertise as '**Intern- Technology**' on retainership basis and subsequent discussions you had with us. We are pleased to engage you as Intern on retainership basis as per the detailed area of work on which your expertise will be required.

The engagement shall be based on the following terms and conditions:

1. That you will provide consultancy as per the area of work to the Company initially, at **South2-Bangalore**
2. None of the terms and conditions set forth herein shall be interpreted to create any agency, master-servant, employment or any other relationship between you and the Company, or you and any of the Company's employees.
3. That you shall interact with the Company's 'Representative' as designated by the Company from time to time, to assign you the work and for all other acts, deeds and things as may be incidental in relation thereto. The Representative shall assist you for the performance coordination of the above consultancy job and you shall have to comply with the specifications required and laid out by such Representative.
4. The continued assignment of consultancy job to you will depend entirely at the sole discretion of the Company and nothing herein will be deemed to give you any hope or expectation for continuity of work.
5. That you will exercise due diligence and professional excellence in the exercise of your professional duties under this retainership arrangement. You further undertake that services shall be performed with due care and in a workmanlike manner. Any willful negligence, carelessness or any other professional misconduct

Lenskart Solutions Pvt. Ltd.

12/1, 7th Floor, Vatika Mindscapes, National Highway 2
Sector 27D, Faridabad, Haryana - 121003 | www.lenskart.com
CIN: U51494DL2008PTC178355



shall result in the cancellation of your retainership and/or also an action for recovery of the loss if the Company suffers on account of any lapse on your part.

1. That it will be entirely on discretion of the company whether to provide any assignment to you and it will not be your right to demand assignments from the Company.
2. That you will not disclose either in writing or by word of mouth or by any other means, any confidential information of the Company, regarding the know-how, process of work, marketing, equipment or any other business secrets which you may acquire during the course of the present retainership to any other external person, firm, company or institutions or body of persons or your other clients, if any.
3. You shall promptly notify and fully disclose to Company, in writing, the existence and nature of any and all ideas, designs, practices, processes, improvements and inventions ("Inventions"), which you have conceived or actually reduced to practice during the term of this Retainership or within six-months after termination of this Retainership, if such Inventions relate to a product or process upon which you worked during the term of this Retainership.
4. It is expressly agreed that this Retainership does not, in any manner, permit or assign any right to you upon the trademark/copyrights/other Intellectual Properties of the Company.
5. This Retainership is entered on a mutually exclusive basis. During the term of this Retainership and for a period of twelve months after the termination of this Retainership you shall not, directly or indirectly, on your own or on a third party's behalf, in any capacity, solicit the business or patronage of any customer/s of the Company; divert, entice, or otherwise take away from the Company any customer/s or attempt to do so; solicit or induce any Customer or employee of the Company to terminate or reduce its relationship with the Company. You shall maintain ethical behavior at work and shall not interfere in any manner or matter between our employees and the Company nor shall you indulge into inciting or instigating or in any way divert Company employees from the Company.
6. That you will be paid a monthly retainership fee of **Rs. 20000/-** (subject to deduction of applicable taxes, duties etc. as per the applicable State and Central government rules and regulations) for the duration of this Retainership. The company shall not reimburse or make any additional payments to you for the services rendered under this Retainership except those duly approved by the Company in writing. That you shall be liable to pay the municipal taxes, as applicable, on your own.
7. The monthly retainership fee payment shall be made subject to the satisfactory performance of services by you. The Company has the right to withhold the monthly fee wherein the Company decides that the services have not been rendered with due professional care and skill or for any other reason as the Company may

Lenskart Solutions Pvt. Ltd.

12/1, 7th Floor, Vatika Mindscapes, National Highway 2
Sector 27D, Faridabad, Haryana - 121003 | www.lenskart.com
CIN: U51494DL2008PTC178355



deem fit. The payments shall be withheld till the time further actions in the regard are determined by the Company.

1. It is made clear that this engagement is on retainership basis and in no way to be construed as employment and you will not be entitled to any facility, amenity etc. to which the full time or part time employees of the Company are entitled., other than the allowances and reimbursements on account of official travel as mentioned in clause 11. You will not be entitled to any other payment as any allowances or business contributions etc.
2. To the fullest extent permitted by law, you shall indemnify the Company, hold it harmless, and defend and protect it from and against any and all loss, damage, liability, judgment, claim, cost or expense (including reasonable attorneys' fees and other costs and expenses), of any sort, resulting from injury or damage of any sort to any person or entity, arising out of or in connection with your performance under this Retainership.
3. It is agreed that in the event of any default or breach on your part in the performance and observance of any one or all of the terms and conditions set out in this Retainership, the Company shall be entitled and be at liberty to claim such damages against you as it deems fit or might avail any other remedy that may be available to it under the law.
4. That you shall not assign and/or delegate this retainership either fully or partly, to any other party as the consultancy on retainership is awarded purely due to your personal competence and skill, and hence has been mutually agreed upon.
5. You shall take all such reasonable and adequate measures to protect any personal information/data which might be revealed to you in the course of your duties or revealed to you for the performance of your duties/obligations set out in this Retainership. All liabilities arising for unauthorized disclosure of personal information by you shall be borne solely by you.

The Company reserves the sole discretion to extend the validity period of the Retainership on the terms and conditions as may be determined by the Company. However, the Company reserves the right to terminate the Retainership any time during the term of this Retainership with immediate effect in case of breach of any of the terms and condition of this agreement or in case your performance is assessed by the Company as unsatisfactory.

Lenskart Solutions Pvt. Ltd.

12/1, 7th Floor, Vatika Mindscapes, National Highway 2
Sector 27D, Faridabad, Haryana - 121003 | www.lenskart.com
CIN: U51494DL2008PTC178355



1. It is specifically agreed between both the parties that either Party reserves the right to terminate this Retainership at any time during the term by giving a prior notice of 15 days without assigning any reason whatsoever or by paying the proportionate amount of the Notice period in lieuthereof.
2. Prevention of sexual harassment: Harassment of any kind will not be tolerated by the Company whether during the course of duty as set out in this Retainership or even while you are not acting under the duties/obligations of this Retainership. You are expected to comply with the Company's policy prohibiting sexual harassment, a copy of which is attached herein for your reference (Annexure II).
3. You shall not indulge in corrupt practices or shall commit any act or omission while performing any obligation under this Retainership which might be considered as unethical and/or corrupt under general and reasonable norms and standards.
4. If any provision set forth herein is invalid or unenforceable under any law, the validity of the remainder of this Retainership shall not be affected and such provision/s shall be deemed modified to the minimum extent necessary to make it consistent. No amendment or variation to this Retainership will be effective unless in writing signed by an authorized representative of each Party.
5. The terms and conditions herein shall be governed by and construed in accordance with the laws of India and the courts in Delhi shall have exclusive jurisdiction over the matter concerning this Retainership.
6. This Letter constitutes the entire agreement between the Parties with respect to the services.
7. As a token of acceptance of the above terms and conditions, please sign the duplicate copy of the letter hereof and return the same to us.

Yours truly,
For Lenskart Solutions Pvt. Ltd.

Received and Accepted

A handwritten signature in black ink, appearing to read "Rakesh".

Authorized Signatory

DECLARATION

I **GUNDAPU NITISH KUMR (18311A0519)** students of **SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY, YAMNAMPET, GHATKESAR**, studying IV year II semester, **COMPUTER SCIENCE AND ENGINEERING** solemnly declare that the Project-II work, titled "**AUTOMATED CHATBOT FOR LANGUAGE TRANSLATION AND RECOGNITION**" is submitted to **SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY** for partial fulfillment for the award of degree of Bachelor of technology in **COMPUTER SCIENCE AND ENGINEERING**.

It is declared to the best of our knowledge that the work reported does not form part of any dissertation submitted to any other University or Institute for award of any degree.

ACKNOWLEDGEMENT

I would like to express my gratitude to all the people behind the screen who helped me to transform an idea into a real application.

I would like to express my heart-felt gratitude to my parents without whom I would not have been privileged to achieve and fulfill my dreams. I am grateful to our principal, **DR. T. Ch. Siva Reddy**, who most ably runs the institution and has had the major hand in enabling me to do my project.

I profoundly thank **Dr. Aruna Varanasi**, Head of the Department of Computer Science & Engineering who has been an excellent guide and also a great source of inspiration to my work.

I would like to thank our Coordinator **Mrs. Bejjam Vasundhara Devi** & my internal guide **Dr. Halavath Balaji** for Project-II for their technical guidance, constant guidelines, encouragement and support in carrying out my project on time at college.

The satisfaction and euphoria that accompany the successful completion of the task would be great but incomplete without the mention of the people who made it possible with their constant guidance and encouragement crowns all the efforts with success. In this context, I would like to thank all the other staff members, both teaching and non-teaching, who have extended their timely help and eased my task.

GUNDAPU NITISH KUMAR

18311A0519

ABSTRACT

These days Chat has become the new way of conversation and changed the way of life and the view that the world used to see before and due to Industrial revolution 4.0 , the gradual increase in machine learning and artificial intelligence fields has gone to higher and many companies are reaching customers to get their products with more ease . This is where chatbots are used. It all started with one question! can machines think? The concept of chatbots came into existence to check whether the machines could fool users and make them think that they are actually talking to humans and not robots. On the Other hand, with the Successes Rate of Chat bots, Different companies Started using machines for having conversations with their customers about everything which made their work simpler and reduced the need of man power. There are many different types of building a chatbot but in this project, we will mainly focus on the chatbot used for the language translation.

LIST OF FIGURES			
S.NO	Figure No.	Title of Figure	Page No
1	4.1.1	Architectural Design	5
2	4.2.1	Use Case Diagram	6
3	4.2.2	Class Diagram	7
4	4.2.3	Activity Diagram	8
5	4.2.4	Sequence Diagram	9
6	5.3.1	Login Page	57
7	5.3.2	Invalid Username	57
8	5.3.3	Invalid Password	58
9	5.3.5	Chatbot Initial page	59
10	5.3.6	After Clicking first Option	59
11	5.3.7	Place an order online	60
12	5.3.8	Place an order offline	60
13	5.3.9	Finding nearest lenskart store -1	61
14	5.3.10	Finding nearest lenskart store -2	61
15	5.3.11	Cancellation of Ordered items -1	62
16	5.3.12	Cancellation of Ordered items -2	62
17	5.3.13	Making home appointment -1	63
18	5.3.14	Making home appointment -2	63
19	5.3.15	Language translation option -1	64
20	5.3.16	Other Chatbot Message	64
21	5.3.17	Telugu translated options -1	65
22	5.3.18	Telugu translated options -2	65
23	5.3.19	Hindi translated options -1	66
24	5.3.20	Hindi translated options -2	66
25	5.3.21	Tamil translated options -1	67
26	5.3.22	Tamil translated options -2	67
27	5.3.23	kannada translated options -1	68
28	5.3.24	kannada translated options -2	68

INDEX

	Page No
List of Figures	i
List of Tables	ii
1. INTRODUCTION	1
1.1 Project Introduction	1
1.2 Scope and Project Overview	1
1.4 Objectives	1
2. LITERATURE SURVEY	2
2.1 Existing System	2
2.2 Proposed System	2
2.3 Related Work	2
3. SYSTEM ANALYSIS	3
3.1 Functional Requirements	3
3.2 Performance Requirements	3
3.3 Software Requirements	4
3.4 Hardware Requirements	4
4. SYSTEM DESIGN	5
4.1 System Architecture	5
4.2 UML Diagrams	6
4.2.1 Use Case Diagram	6
4.2.2 Class Diagram	7
4.2.3 Sequence Diagram	8
4.2.4 Activity Diagram	9
4.3 Modules	10
5. IMPLEMENTATION AND RESULTS	11
5.1 Language / Technology Used	11
5.2 Code	11
5.3 Results (Accuracy) / Output Screens	57
6. TESTING	69
6.1 Types of Testing	69
6.2 Test Cases	70
7. CONCLUSION AND FUTURE SCOPE	72
8. BIBLIOGRAPHY	73
PLAGIARISM REPORT	74
PAPER PUBLICATION	75
ABSTRACT	81
PROJECT CORRELATION WITH PO's AND PSO's	82
NATURE OF THE PROJECT	83
DOMAIN OF THE PROJECT	84

1. INTRODUCTION

1.1 Project Introduction

As we know these days chat is one of the most efficient ways of communication in which user communicates efficiently with the system and gives his appropriate views of the topic and gets to know the product better. These days many companies are providing chatbot facility in their products such as websites and apps hence giving more opportunities for the customer to interact with the system and server. Hence lenskart is also trying to improve its approachability by increasing the number of customers using chatbot. But this seemed to be a little problematic in India as there are many languages in different parts of India such as telugu, hindi, tamil, kannada, marathi etc. hence the lenskart team has decided to introduce their chatbot service in different regional languages so that the user can feel comfortable while having a conversation.

1.2 Scope and Project Overview

The project is mainly based on the chat responses given after the user has given any input. Hence a recurrence neural network-based model is defined to understand the words in the message of the user and to predict the outcome of the result.

The front end of the project is designed in react-js for the code simplicity and optimizability. The backend for the server uses google cloud api for the google translation of the messages of the user. Lastly the machine learning model is deployed to get the result. The whole project consists of 3 major modules which are important for the building of the project.

Originally the project is done in HTML and CSS and JAVASCRIPT as front end and node js along with EXPRESS JS as backend. Due to copyright issues, it has been done in different programming language.

1.3 Objectives

The main objective of this project is to provide the chatbot services of the lenskart to different regional people and to expand the services of the company to many regions. Using this project's

concept many people can interact with the user using chatbot and can have then spectacles ordering simpler and easier manner.

2. LITERATURE SURVEY

2.1 Existing System

The system consists the chatbot services without language friendly and the message are not allowed to be typed until and unless the chat bot asks. It also lacks the effective communication between the user and the server hence making it more difficult for the interaction and also the messages are only available in one language hence making it difficult for other regional people.

2.2 Proposed System

The proposed system has the additional features and the also embedded with a machine learning model which enables the chat responses to be more realistic than computerized.

Some additional features are:

- ⇒ Language changing feature
- ⇒ Language recognizing and translating feature
- ⇒ More realistic messages produced using model
- ⇒ The backend server is more responsive as it is using unicorn server

2.3 Related Work

This kind of the chatbot feature is not new in the industry as we can see many companies already implemented language translation feature in its respective chatbots. But this kind of development is new and first time in lenskart history as we are trying to reach out many different regions and expand its approachability.

3. SYSTEM REQUIREMENT ANALYSIS

For the analysis of our system, I will take apart our project and thoroughly inspect it section by section, discussing the function of each individual section. I will also provide the performance requirements along with the hardware and software requirements.

3.1 Functional Requirement Specifications

For the Frontend Functional Requirements are concerned, this project was implemented using react-js module using node js as a component. Hence any IDE capable of running JS code will satisfy the functional requirements respectively. It is preferable to use WEBSTORM or VS CODE for having the computation process easier.

For the Backend Functional Requirements are concerned, this project was implemented in python_3.9 using tensorflow_gpu 2.7 version in Jupyter Notebook. So, any IDE capable of running TensorFlow Python code using keras_2.4.3 api or higher and python 3.9 or higher will satisfy the functional requirements respectively. It is preferable to use PYCHARM or VS CODE for having the computation process easier.

3.2 Performance Requirements

For our proposed system to be adept, it should have a few basic pre-defined performance requirements. For our system the performance requirements can be briefly put into the points as below:

- The OS and the python, node-js versions of the system must be up-to date.
- The GPU of the system will be used; hence the respective drivers should be installed respectively and it must to be up-to date.
- It would be preferable to use a modern browser such as Chrome or Firefox so that the CSS can be reflected correctly.

3.3 Software Requirements

- **Operating System:** Windows, MacOS, or any other Linux-based operating system which supports TensorFlow and Python.
- **IDE :** Any IDE which can run python Code. Preferable PyCharm or Visual studio Code.
- **Applications Used :** ReactJS, FastAPI, Firebase are used for the creating this project.
- **Browsers :** Chrome, Firefox or Safari
- **Deployment platform :** Heroku, hugging faces for backend and netlify for frontend.

3.4 Hardware Requirements

- **Processor :** Any processor i3 or higher.
- **Speed :** 1.2 GHZ or more.
- **Disk :** 24 GB or more.
- **RAM :** A minimum of 4 GB.
- **GPU :** NVIDIA TESLA K80 or higher is required.

4.SYSTEM DESIGN

4.1 System Architecture

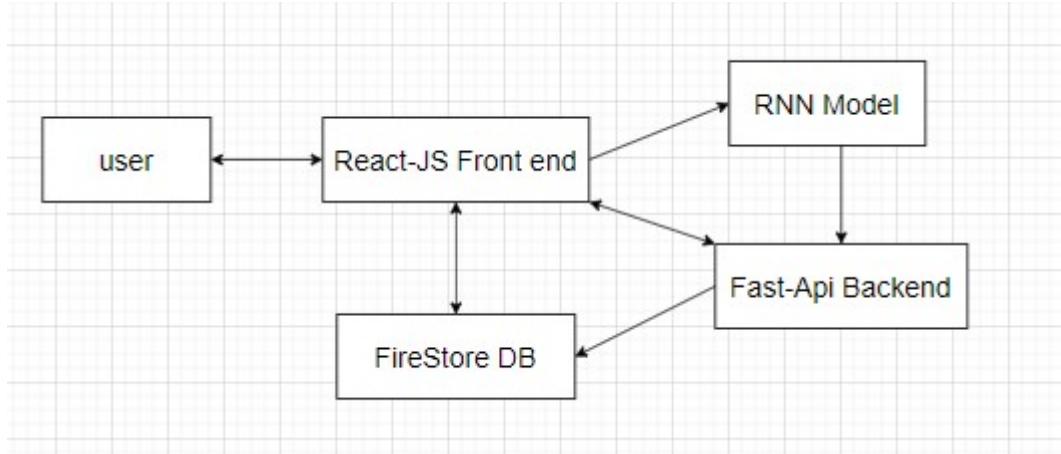


Fig 4.1.1 Architectural Design

This is the architectural design of our project. Our project consists of a user interface designed in react-js which is embedded with the trained RNN model with the training data that is used to take the input message and gives the appropriate response for the message.

The training data that is used in the model are the messages that are classified based on the context such as greeting, hello, help, hi etc. and are trained on the sequential model with a RNN layer to get the trained model.

The backend server of this project is designed in fast-api using python, which helps to retrieve and translate the messages of the chatbot from the user. The fast-api server is also used for other functionalities of the chatbot and helps to get a better response.

The Fire Store is a google based online database system which helps in storing the data of a website and helps to recognize or retrieve data from it. We have used firestore in this project to store the user messages in the form of message collections and the admin will have the access to it so that they can do the sentimental analysis of it to get a better data. The additional feature is firestore can be used to store the messages of different languages.

4.2 UML Diagrams

UML Diagrams for our application are as follows:

4.2.1 Use Case Diagram

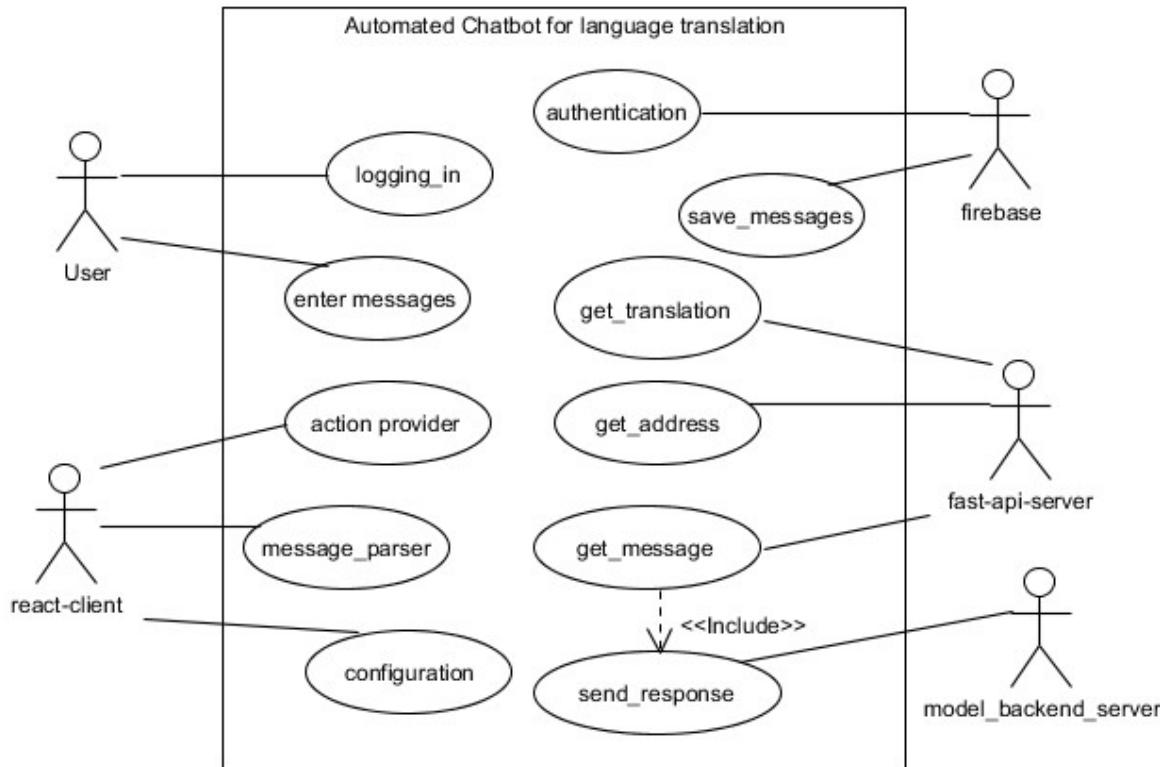


Fig 4.2.1 Use Case Diagram

This is the Use case diagram of our project which describes the expected behaviour of our project. Here we have five actors one is user and other are react-client, firebase, fast-api-server and model-backend. Where the user is able to get the predicted message when he/she types the message in the chatbot.

As for the react-client it acts as the front-end of our project which offers the UI and fast-api-server acts as the backend-server offering the functionalities of the frontend and firebase acts as the online database to store the responses of the user. Model is used to produce the output.

4.2.2 Class Diagram

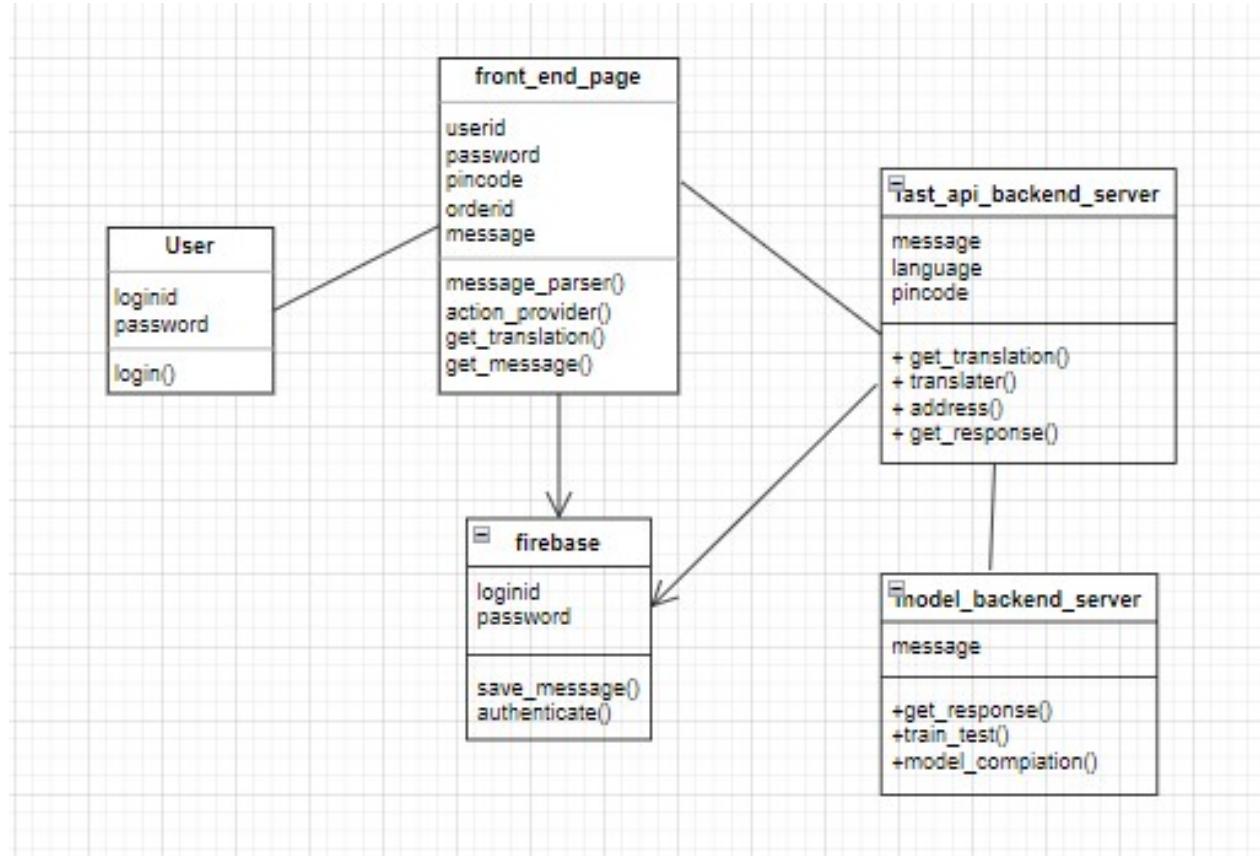


Fig 4.2.2 Class Diagram

This is the class diagram of our project where it presents a type of static structural diagram that describes the structure of our system by showing the system's classes, their attributes, operations, and the relationships among class present in the system. Our class diagram has a total of 5 classes where the classes front end, firebase and backed, firebase are in one way relationship where they are the part of the backend that we used in our project. It shows that the messages are stored in the firestore whenever the messages are coming to backend.

As for the remaining classes, the user is connected with the frontend showing the 2-way relationship between where the user can login and type the message and get the appropriate response for the message, and the model is connected with the backend for message translation and also connected with the front end showing the predicted message and also the translated response for the user's message.

4.2.3 Activity Diagram

This is the activity diagram of our project which represents the activity flow of the project. Here first the user is allowed to login if the login is failed, he will be in the same page until he gets it right and if the login is successful, he will be redirected to next page where he can interact with the chatbot.

From there the user messages are read and the appropriate responses for that message are calculated using the model and the responses are translated in backend sever and these messages are sent into the front-end server for displaying.

This whole process continues until the user is satisfied. i.e., the user getting what he wanted and he can logout which is present in the chatbot options which gets him back in the login page.

This whole process is only for the typed messages and if there are any options then the react client will able to take care of it with out the help of the model for the prediction.

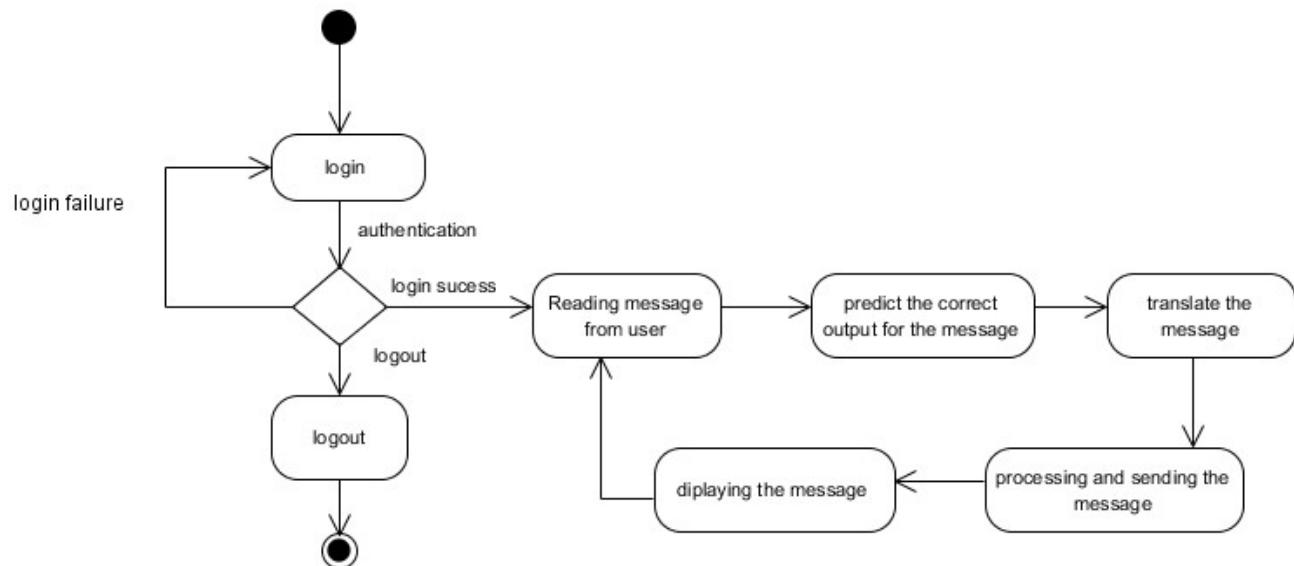


Fig 4.2.3 Activity Diagram

4.2.4 Sequence Diagram

This is the sequence diagram which represents the interactions between the system components and shows the sequence of interactions between the system components. first the user interacts with the system by logging in and typing a message and the system interacts with the model by processing the message and the model interacts with itself by loading the saved data and predicting the output and the response is sent into the backend server for the language translation and sends the response to the react client which displays the message. This whole process continues until the user interactions are completed and at last he can logout from chatbot options and return to the login page.

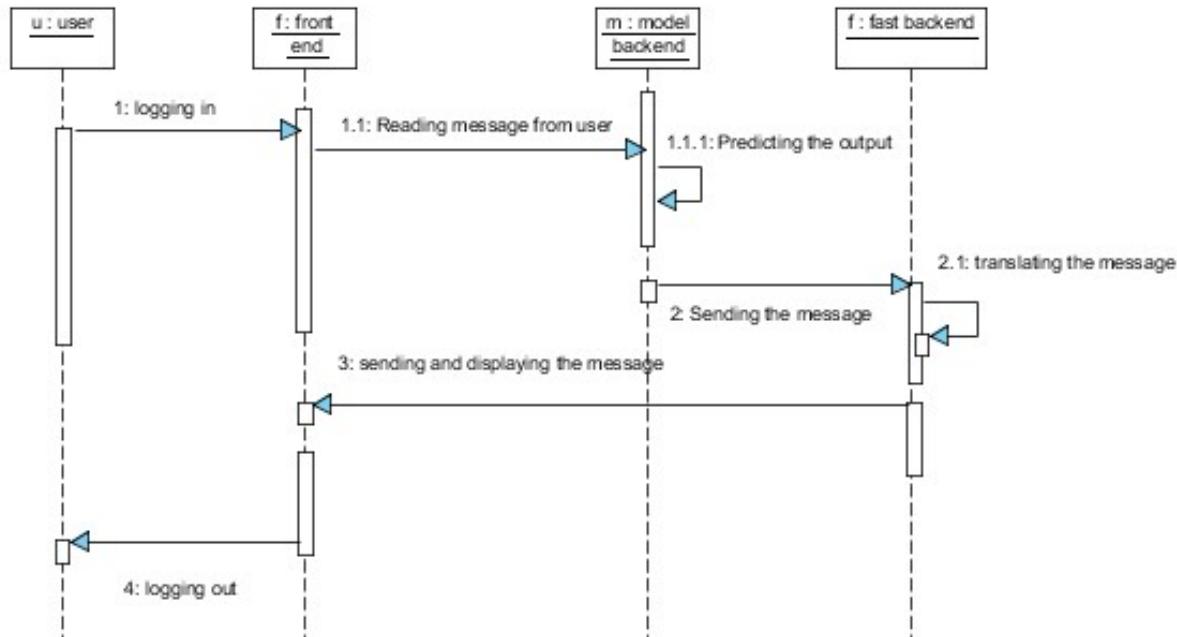


Fig 4.2.4 Sequence Diagram

This Concludes the System Architecture and the UML Diagrams which represents the internal working of the project.

4.3 Modules

The Modules present in the project are:

1. **User Interface:** It act as the base of the whole project and it acts as the intermediate between the model, online database and the backend server. This is designed in react in JavaScript. It is also used to authenticate the user for security
2. **Backend Model:** This accepts the text message from the user and predicts the appropriate response for the message. It is designed using recurrence neural networks in tensorflow using sequential model.
3. **Online Database:** It is used to store the messages that are typed or selected by the user for the sentiment analysis purpose and these messages are accessible only by admin.
4. **Backend Server:** It is used to satisfy the functionalities of the chatbot of the UI, and also for the language translation purposes. It is the main backbone of the project which supports the front-end UI.

This Concludes the Modules of the project.

5. IMPLEMENTATION AND RESULTS

5.1 Language and technologies

This project is divided into 3 parts. They are:

1. **Front end:** designed in ReactJS, a popular web framework used to design the static and dynamic websites front-end in JavaScript.
2. **Back end:** has two components
 1. **Model:** Designed in Tensorflow, a popular API for neural networks in python.
 2. **Server:** Designed in Fast-API, an API which provides REST services in python.
3. **Database:** Firebase is used as the online Storage.

5.2 Code

5.2.1 Backend Server (Fast-api)

main.py

```
1  from fastapi import *
2  from test import get_res
3  from states import get_address
4  from pydantic import BaseModel
5  from googletrans import Translator
6  from fastapi.middleware.cors import CORSMiddleware
7
8  translator = Translator()
9  app = FastAPI()
10
11 origins = ["*"]
12
13 app.add_middleware(
14     CORSMiddleware,
15     allow_origins=origins,
16     allow_credentials=True,
17     allow_methods=["*"],
18     allow_headers=["*"],
19 )
```

```

22 @app.get("/translater")
23     async def translater(msg,lang):
24         result = translator.translate(msg,dest=lang)
25         return str(result.text)
26
27
28 @app.get("/translation")
29     async def translation(msg,lang):
30         res = []
31         messages = msg.split(",")
32         for a in messages:
33             res.append(translator.translate(a,dest=lang).text)
34         return res
35
36 @app.get("/address")
37     async def getAddress(pincode,lang):
38         res = []
39         result1 = translator.translate("The Nearest Store is : ",dest=lang).text
40         messages = get_address(int(pincode))
41         for a in messages:
42             res.append(translator.translate(a,dest=lang).text)
43         return result1 + res[0]
44
45 @app.get("/get_response")
46     async def getResponse(msg,lang):
47         index = -5
48         if "help" in msg:
49             index=2
50         elif "hi" in msg or "hello" in msg:
51             index=1
52         elif "bye" in msg:
53             index=3
54         else:
55             index=4
56
57         if index == 4:
58             res = translator.translate("please enter the correct input",dest=lang).text
59         else:
60             r = get_res(index)
61             res = translator.translate(r,dest=lang).text
62         return res

```

States.py

```
3 states = {
4     500094 : ["H No 37 74/11/1, Plot No E-62, Survey No 218/1/3, JJ Nagar Ramakrishnapuram, "+  
5         "Malkajgiri Municipality and Mandal Ranga Reddy , Neredmet ,Hyderabad, 500094"],  
6     500015 : ["Plot No 6 S, Nos 60 6162/1, New Vasavi Colony, Second Venture Kakaguda, Village"+  
7         " Secunderabad Cantonment Area, Vikrampuri"],  
8     500039 : ["FF 02 & 03, First Floor, DSL Virtue Mall, Ramanthapur Main Road IDA Uppal"],  
9     500079 : ["Shop No 17/1/383/N/80/31, Block No 1, Brindavan Colony Nagar"],  
10    500074 : ["Plot No 124 In Survey No 64, Ward No 3, Block No 8, Door No 3-11-22, Sri Enclave"+  
11        ", Suvidha Arcade, RTC Colony, LB Nagar, Bahadurguda"],  
12    500072 : ["D#12-6-2/216, P#1 SY#577, Ground Floor, Kukatpally","MIG 161,"+  
13        " Ground Floor, Road No 1, KPHB Colony"],  
14    500084 : ["Door No. 4-1382, Raja Rajeswara Nagar, Kondapur Village, "+  
15        "Serilingampally Mandal, Ranga Reddy"],  
16    500054 : ["Shop No 6/7, House No 60/9/9, MPR Complex , IDPL Y Junction, Chintal"],  
17    500062 : ["Victory Plaza, Plot No 11 Extension, ECIL X Main Road, Khushaiguda"],  
18    500036 : ["Shop No 16/10/27/105/7/4/1, Tirumala Pride, Main Road ,Malakpet"],  
19    500003 : ["Shop No 1,7 & 396, Sandhu Apartment, Jain Estate, Kalasiguda, "+  
20        "Secunderabad ,Kalasiguda","PB 1549, M G Road, Kalasiguda, Secunderabad,MG Road"],  
21    500044 : ["House No 1/9/670/69, Ground Floor, SRT Quarters,Vidya Nagar, Adikmet"],  
22    500072 : ["MIG 42, Ground Floor, KPHB Phase 1, Dharma Reddy Colony ,Kukatpally"],  
23    500084 : ["Shop No 1/112/80,Kondapur<br>Hyderabad, 500084"],  
24    500026 : ["Building No, 3 & 6,101/C/1J, H/228, Chiranjeevi Apartment,"+  
25        " West Marredpally Road, Boosareddy Guda, Aswini Colony , West Marredpally"],  
26    500044 : ["H No 2/2/647/3/A/3, Ground Floor, Shivam Road, Amberpet, "+  
27        " APHB Colony, New Nallakunta New Nallakunta"],  
28    501218 : ["23/65/1, Madhura Nagar, Shamsahabad, Rangareddy,Shamsahbad"],  
29    500081 : ["H No 1-90/20/G/1, Sai Siddartha Complex, Metro Pillar No 1760, "+  
30        "Main Road , Madhapur","Shop No 64, SN DK1A, Level 4, Food Court, Inorbit Mall"+  
31        " , APIIC Software Layout, Madhapur"],  
32    500007 : ["Shop No 1/4/53/1, Street No 8 ,Habsiguda"],  
33    500015 : ["Plot No 57, Ground Floor, Part A1, P & T Colony ,Trimulgherry"],  
34    500029 : ["Shop No 3, 6/564/5 To 8, Circle, Himayat Nagar, AP State"+  
35        " Housing Board,Himayat Nagar"],  
36    500038 : ["262/3RT, Ground Floor, MN Reddy Classic, SR Nagar"],  
37    500060 : ["Shop No 13/4/30, Ground Floor,Dilsukhnagar, Vikas Nagar"],  
38    500070 : ["Ground Floor, Shop No 1, Old Court Building, Nagarjun Sagar"+  
39        " Road, Omkar Nagar,Omkar Nagar"],  
40    500050 : ["2-88, National Highway 65, Gangaram, Chanda Nagar BHEL"],  
41    500049 : ["Shop No G2, Ground Floor, Kalyan Tulasi Ram Chambers, Madeenaguda",  
42        "Shop No FF 101, First Floor, GSM Mall, Madinaguda"],  
43    500031 : ["H No 1-57/19, Sri Ram Nagar, Botanical Garden Road, Kondapur"],  
44    500029 : ["Flat No 3-6-365, Y V Rao Mansion;105 Himayatnagar Road, "+  
45        "Gagan Mahal, Himayatnagar"],  
46    500059 : ["Shop No 2, Ground Floor, 17-1-376/3, 34 Khaja Estate, Shantosh Nagar"],  
47    500034 : ["MCH No 6/3/348/B/201A, Ground Floor, BKE Owners Society, Road 1"],  
48    500019 : ["Ground Floor, GHMC 7-7/205, Nallagandla, Near Reliance Digital , Nallagandla"],  
49    500008 : ["House No 8-1-284/OU/20, Ground Floor, OU Colony, Shaikpet",  
50        "13-6-798/1/21, New Friends Colony, Langar Houz"],
```

```

51    500010 : ["171 12, NR Temple, Alwal village, GHMC, Alwal Circle, "+  

52    | "Malkajgiri Mandal, Ranga Reddy"],  

53    500018 : ["House No 3-6-23/4, Plot No 297, Survey Numbers 137&139, "+  

54    "Bagh Ameeri village, Balanagar Madal, Kukatpally Circle GHMC, Ranga Reddy"],  

55    500036 : ["16-11-741/C/13/5, Ground Floor, Vijetha Royal Empire, "+  

56    | "Indira Nagar, Dilsukh Nagar"],  

57    500062 : ["House No 3-30/1, Plot No1, Sri Nagar Colony, Kapra Village &"+  

58    | " Mandal, Medchal District"],  

59    500048 : ["Ground Floor, Baba Towers, Tilak Road, King Koti"],  

60    500067 : ["Plot No 6, Ground Floor, Ganesh Housing Colony, "+  

61    | "Suchitra X Roads, Quthbullapur"],  

62    500085 : ["Dno-16-2-M/14/G1, Ground Floor, Sai Plaza, Charma Reddy Colony, "+  

63    | "Phase II, Hydernagar Village, kukatpally"],  

64    500089 : ["Plot # 68A, Alkapur Road # 7, Puppal Guda"],  

65    500033 : ["Municipal No 8/2/293/82/A471, Road No 36"],  

66    500076 : ["EMR Complex No 4-1-91/6, Bhavani Nagar, Nacharam Main Road, Nacharam",  

67    "House No 3-498/1/1/3, Ground Floor, Plot No 3, Situated Mallapur"+  

68    | " village, Uppal Mandal, Kapra Circle"]  

69  }  
  

71  

72  def closest_value(input_list, input_value):  

73    difference = lambda input_list : abs(input_list - input_value)  

74    res = min(input_list, key=difference)  

75    return res  

76  

77  def get_address(pincode):  

78    val=closest_value(sorted(states.keys()),pincode)  

79    return states[val]  

80

```

Google Cache

```

1  #LWP-Cookies-2.0  

2  Set-Cookie3: 1P_JAR="2022-05-17-07"; path="/"; domain=".google.co.in"; path_spec;  

3  | domain_dot; secure; expires="2022-06-16 07:06:26Z"; version=0  

4  Set-Cookie3: AEC="AakniG0dGmcxJ87zGKBjyf4IR4YZsfrh1iU-Hv-EPdXhuCs0_V9bxSzVJuU"; path="/";  

5  | domain=".google.co.in"; path_spec; domain_dot; secure; expires="2022-11-13 07:04:21Z";  

6  | HttpOnly=None; SameSite=lax; version=0  

7  Set-Cookie3: NID="511=LXUPVh4Kd0HxA7CLPfqL87fA_Xh7oRz_qKZ78PxuGoKkzMNzR80eO_adKHJ6E-9-  

8  | Phf-S8d3zkz1dQ6AGXonnisvii2jL0qzrf_F-swysNedheCmI-61u0mGWTEScfRu0waHsD10jRQDtWj  

9  | z0bi5Bzg_izEJM6046IxESxlRw3A"; path="/";  

10 | domain=".google.co.in"; path_spec; domain_dot;  

11 | expires="2022-11-16 07:04:23Z"; HttpOnly=None; version=0

```

Requirements.txt

```
1  absl-py==0.15.0
2  anyio==3.5.0
3  asgiref==3.4.1
4  astunparse==1.6.3
5  beautifulsoup4==4.11.1
6  CacheControl==0.12.11
7  cached-property==1.5.2
8  cachetools==4.2.4
9  certifi==2020.6.20
10 chardet==3.0.4
11 charset-normalizer==2.0.12
12 clang==5.0
13 click==8.0.4
14 colorama==0.4.4
15 contextlib2==21.6.0
16 contextvars==2.4
17 cycler==0.11.0
18 dataclasses==0.6
19 fastapi
20 firebase==3.0.1
21 firebase-admin==5.2.0
22 flatbuffers==1.12
23 gast==0.4.0
24 google==3.0.0
25 google-api-core==2.7.2
26 google-api-python-client==2.45.0
27 google-auth==1.35.0
28 google-auth-httplib2==0.1.0
29 google-auth-oauthlib==0.4.6
30 google-cloud-core==2.3.0
31 google-cloud-firestore==2.4.0
32 google-cloud-storage==2.0.0
33 google-crc32c==1.3.0
34 google-pasta==0.2.0
35 google-resumable-media==2.3.2
36 google-search==1.1.1
37 googleapis-common-protos==1.56.0
38 googletrans==3.1.0a0
39 grpcio==1.44.0
40 grpcio-status==1.44.0
41 gunicorn==20.1.0
42 h11==0.9.0
```

```
43 h2==3.2.0
44 h5py==3.1.0
45 hpack==3.0.0
46 hstspreload==2021.12.1
47 httpcore==0.9.1
48 httplib2==0.20.4
49 httptools==0.3.0
50 httpx==0.13.3
51 hyperframe==5.2.0
52 idna==2.10
53 immutables==0.17
54 importlib-metadata==4.8.3
55 importlib-resources==5.4.0
56 itsdangerous==2.0.1
57 Jinja2==3.0.3
58 joblib==1.1.0      You, 4 weeks ago •
59 kiwisolver==1.3.1
60 lxml==4.8.0
61 Markdown==3.3.7
62 MarkupSafe==2.0.1
63 matplotlib==3.3.4
64 msgpack==1.0.3
65 mysql-connector==2.2.9
66 nltk==3.6.7
67 numpy==1.19.5
68 oauthlib==3.2.0
69 opencv-python==4.5.4.60
70 opt-einsum==3.3.0
71 pickle-mixin==1.0.2
72 Pillow==8.4.0
73 proto-plus==1.20.3
74 protobuf==3.19.4
75 pyasn1==0.4.8
76 pyasn1-modules==0.2.8
77 pydantic==1.9.0
78 pyparsing==3.0.6
79 python-dateutil==2.8.2
80 python-dotenv==0.20.0
81 python-multipart==0.0.5
82 PyYAML==6.0
83 regex==2022.4.24
84 requests==2.27.1
85 requests-oauthlib==1.3.1
```

```
86 rfc3986==1.5.0
87 scipy==1.5.4
88 six==1.15.0
89 sniffio==1.2.0
90 soupsieve==2.3.2.post1
91 starlette==0.17.1
92 termcolor==1.1.0
93 threadpoolctl==3.0.0
94 tqdm==4.64.0
95 typing-extensions==3.7.4.3
96 uritemplate==4.1.1
97 urllib3==1.26.9
98 uvicorn==0.16.0
99 watchdog==0.7
100 websockets==9.1
101 Werkzeug==2.0.3
102 wincertstore==0.2
103 wrapt==1.12.1
104 zipp==3.6.0
```

Procfile

```
4 web: gunicorn -w 4 -k uvicorn.workers.UvicornWorker main:app
```

Test.py

```
1 import requests
2
3 def get_res(index):
4     r = requests.post(url="https://hf.space/embed/nitishkumar"+
5                         "gundapu793/chat-bot-response/api/predict/",
6                         json={'fn_index': 2, 'data': [index],
7                               'session_hash': "73g8yepkp3b"})
8     return r.json()['data'][1]
```

Runtime.txt

```
4 python-3.9.4
```

5.2.2 Backend Model (Hugging Faces)

App.py

```
2  import gradio as gr
3  import nltk
4  from nltk.stem import WordNetLemmatizer
5  lemmatizer = WordNetLemmatizer()
6  import pickle
7  import numpy as np
8  import nltk
9  nltk.download('punkt')
10 nltk.download('wordnet')
11 nltk.download('omw-1.4')
12 nltk.download('pros_cons')
13 nltk.download('reuters')
14
15 from keras.models import load_model
16 model = load_model('models/rnn_model.h5')
17 import json
18 import random
19 intents = json.loads(open('data.json').read())
20 words = pickle.load(open('texts.pkl','rb'))
21 classes = pickle.load(open('labels.pkl','rb'))
22
23 def clean_up_sentence(sentence):
24     sentence_words = nltk.word_tokenize(sentence)
25     sentence_words = [lemmatizer.lemmatize(word.lower())
26                      for word in sentence_words]
27     return sentence_words
28
29 def bow(sentence, words, show_details=True):
30     sentence_words = clean_up_sentence(sentence)
31     bag = [0]*len(words)
32     for s in sentence_words:
33         for i,w in enumerate(words):
34             if w == s:
35                 bag[i] = 1
36                 if show_details:
37                     print ("found in bag: %s" % w)
38     return(np.array(bag))
```

```

40 def predict_class(sentence, model):
41     # filter out predictions below a threshold
42     p = bow(sentence, words, show_details=False)
43     res = model.predict(np.array([p]))[0]
44     ERROR_THRESHOLD = 0.25
45     results = [[i,r] for i,r in enumerate(res)
46                if r>ERROR_THRESHOLD]
47     # sort by strength of probability
48     results.sort(key=lambda x: x[1], reverse=True)
49     return_list = []
50     for r in results:
51         return_list.append({"intent": classes[r[0]],
52                            "probability": str(r[1])})
53     return return_list
54
55 def getResponse(ints, intents_json):
56     tag = ints[0]['intent']
57     list_of_intents = intents_json['intents']
58     for i in list_of_intents:
59         if(i['tag']== tag):
60             result = random.choice(i['responses'])
61             break
62     return result
63
64 def chatbot_response(msg):
65     ints = predict_class(msg, model)
66     res = getResponse(ints, intents)
67     return res
68
69 gr.Interface(fn=chatbot_response, inputs=["text"],
70               outputs=["text"], examples=[["hello"], ["hi"], ["help"],
71               ["bye"]]).launch()

```

\

Chat-retriver.py

```
2  import nltk
3  from nltk.stem import WordNetLemmatizer
4  lemmatizer = WordNetLemmatizer()
5  import pickle
6  import numpy as np
7
8  from keras.models import load_model
9  model = load_model('models/rnn_model.h5')
10 import json
11 import random
12 intents = json.loads(open('data.json').read())
13 words = pickle.load(open('texts.pkl','rb'))
14 classes = pickle.load(open('labels.pkl','rb'))
15
16 def clean_up_sentence(sentence):
17     sentence_words = nltk.word_tokenize(sentence)
18     sentence_words = [lemmatizer.lemmatize(word.lower())]
19     for word in sentence_words:
20         return sentence_words
21
22 def bow(sentence, words, show_details=True):
23     sentence_words = clean_up_sentence(sentence)
24     bag = [0]*len(words)
25     for s in sentence_words:
26         for i,w in enumerate(words):
27             if w == s:
28                 bag[i] = 1
29                 if show_details:
30                     print ("found in bag: %s" % w)
31     return(np.array(bag))
32
33 def predict_class(sentence, model):
34     p = bow(sentence, words,show_details=False)
35     res = model.predict(np.array([p]))[0]
36     ERROR_THRESHOLD = 0.25
37     results = [[i,r] for i,r in enumerate(res)]
```

```

38     if r>ERROR_THRESHOLD]
39     # sort by strength of probability
40     results.sort(key=lambda x: x[1], reverse=True)
41     return_list = []
42     for r in results:
43         return_list.append({"intent": classes[r[0]],
44                             "probability": str(r[1])})
45     return return_list
46
47 def getResponse(ints, intents_json):
48     tag = ints[0]['intent']
49     list_of_intents = intents_json['intents']
50     for i in list_of_intents:
51         if(i['tag']== tag):
52             result = random.choice(i['responses'])
53             break
54     return result
55
56 def chatbot_response(msg):
57     ints = predict_class(msg, model)
58     res = getResponse(ints, intents)
59     return res

```

Requirements.txt

```

3    nltk
4    tensorflow

```

nltk.txt

```

3    wordnet
4    pros_cons
5    reuters

```

training.py

```
2  import nltk
3  from nltk.stem import WordNetLemmatizer
4  lemmatizer = WordNetLemmatizer()
5  import json
6  import pickle
7
8  import numpy as np
9  from keras.layers import Dense
10 from tensorflow.keras import layers
11 from keras.models import Sequential
12 from keras.layers import Dense
13 import random
14
15 words=[]
16 classes = []
17 documents = []
18 ignore_words = ['?', '!']
19 data_file = open('data.json').read()
20 intents = json.loads(data_file)

23 for intent in intents['intents']:
24     for pattern in intent['patterns']:
25         w = nltk.word_tokenize(pattern)
26         words.extend(w)
27         documents.append((w, intent['tag']))

29         if intent['tag'] not in classes:
30             classes.append(intent['tag'])

32 words = [lemmatizer.lemmatize(w.lower())
33           for w in words if w not in ignore_words]
34 words = sorted(list(set(words)))
35 classes = sorted(list(set(classes)))
36
37
38 pickle.dump(words,open('texts.pkl','wb'))
39 pickle.dump(classes,open('labels.pkl','wb'))
```

```

42 training = []
43 output_empty = [0] * len(classes)
44 for doc in documents:
45     bag = []
46     pattern_words = doc[0]
47     pattern_words = [lemmatizer.lemmatize(word.lower())]
48     for word in pattern_words:
49         for w in words:
50             bag.append(1) if w in pattern_words else bag.append(0)
51     output_row = list(output_empty)
52     output_row[classes.index(doc[1])] = 1
53
54     training.append([bag, output_row])
55 random.shuffle(training)
56 training = np.array(training)
57
58 train_x = np.array(list(training[:,0]))
59 train_y = np.array(list(training[:,1]))
60
61
62 model = Sequential()
63 model.add(layers.Embedding(input_dim=len(train_x),
64 output_dim=len(train_y)))
65 model.add(layers.SimpleRNN(len(train_y)))
66 model.add(Dense(len(train_y[0]), activation='softmax'))
67
68 ~ model.compile(loss='categorical_crossentropy', optimizer='adam',
69   metrics=['accuracy'])
70
71 ~ hist = model.fit(np.array(train_x), np.array(train_y),
72   epochs=100, batch_size=5, verbose=1)
73 model.save('models/rnn_model.h5', hist)
74
75
76 print("model created")
77 print(model.summary())

```

Data.json

```
3  {
4      "intents": [
5          {
6              "tag": "greeting",
7                  "patterns": ["Hi there", "How are you", "Is anyone there?", "Hey", "Hola", "Hello", "Good day"],
8                  "responses": ["Good to see you again", "Hi there, how can I help?"],
9                  "context": []
10         },
11         {
12             "tag": "thanks",
13                 "patterns": ["Thanks", "Thank you", "That's helpful", "Awesome, thanks", "Thanks for helping me"],
14                 "responses": ["Happy to help!", "Any time!", "My pleasure"],
15                 "context": []
16         },
17         {
18             "tag": "help",
19                 "patterns": ["help me", "cant get any idea"],
20                 "responses": ["click on start for the chat restart and", "select the options for your use"],
21                 "context": []
22         },
23         {
24             "tag": "goodbye",
25                 "patterns": ["Bye", "See you later", "Goodbye", "Nice chatting to you, bye", "Till next time"],
26                 "responses": ["See you!", "Have a nice day", "Bye! Come back again soon."],
27                 "context": []
28         }
29     ]
30 }
31 }
```

Procfile

```
4 web: gunicorn -w 4 -k uvicorn.workers.UvicornWorker main:app
```

5.2.3 Front end (Reactjs)

```
2  {
3      "homepage": ".",
4      "name": "react-client",
5      "version": "0.1.0",
6      "private": true,
7      "dependencies": {
8          "@testing-library/jest-dom": "^5.16.4",
9          "@testing-library/react": "^13.2.0",
10         "@testing-library/user-event": "^13.5.0",
11         "axios": "^0.27.2",
12         "firebase": "^9.8.1",
13         "react": "^18.1.0",
14         "react-chatbot-kit": "^2.0.1",
15         "react-dom": "^18.1.0",
16         "react-router-dom": "^6.3.0",
17         "react-scripts": "^2.1.3",
18         "web-vitals": "^2.1.4"
19     },
20     "scripts": {
21         "predeploy": "npm run build",
22         "deploy": "gh-pages -d build",
23         "start": "react-scripts start",
24         "build": "react-scripts build",
25         "test": "react-scripts test",
26         "eject": "react-scripts eject"
27     },
28     "eslintConfig": {
29         "extends": [
30             "react-app",
31             "react-app/jest"
32         ]
33     },
34     "browserslist": {
35         "production": [
36             ">0.2%",
37             "not dead",
38             "not op_mini all"
39         ],

```

```
40     "development": [
41       "last 1 chrome version",
42       "last 1 firefox version",
43       "last 1 safari version"
44     ]
45   },
46   "devDependencies": {
47     "gh-pages": "^4.0.0"
48   }
49 }
50
```

Index.html

```
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="utf-8" />
5      <meta name="viewport" content="width=device-width, initial-scale=1" />
6      <meta name="theme-color" content="#000000" />
7      <meta
8        name="description"
9        content="Web site created using create-react-app"
10       />
11    </script>
12    <title>Chatbot</title>
13  </head>
14  <body>
15    <noscript>You need to enable JavaScript to run this app.</noscript>
16    <div id="root"></div>
17  </body>
18</html>
19
```

Src/app.js

```
3 √ import React from "react";
4   import "./App.css";
5   import SignIn from './loginpage/SignIn'
6   import ChatBot from './chatbot/ChatBot'
7   import {Routes,Route} from "react-router-dom";
8
9   function App() {
10     return (
11       <div className="wrapper">
12         <Routes>
13           <Route path="/" element={<SignIn />} />
14           <Route path="/chatbot" element={<ChatBot />} />
15         </Routes>
16       </div>
17     );
18   }
19
20   export default App;
```

Src/index.js

```
3   import React from 'react';
4   import ReactDOM from 'react-dom/client';
5   import App from './App';
6   import { BrowserRouter } from "react-router-dom"
7
8   const root = ReactDOM.createRoot(document.getElementById('root'));
9   root.render(
10     <BrowserRouter><App /></BrowserRouter>
11   );
```

Src/App.css

```
3 √ .App {
4   text-align: center; background-color: #282c34;
5   min-height: 100vh; display: flex;
6   justify-content: center; align-items: center;
7 }
```

Src/api/language.jsx

```
2  class language {
3    |  constructor() {
4    |    this.language = "english"
5    |
6
7    |  getLanguage() {
8    |    return this.language
9    |
10
11   |  setLanguage(lang) {
12   |    if (lang === 1){
13   |      this.language = "telugu"
14   |    }
15   |    if (lang === 2){
16   |      this.language = "hindi"
17   |    }
18   |    if (lang === 3){
19   |      this.language = "english"
20   |    }
21   |    if (lang === 4){
22   |      this.language = "tamil"
23   |    }
24   |    if (lang === 5){
25   |      this.language = "kannada"
26   |    }
27   |  }
28 }
29
30
31 var a = new language()
32 export default a
```

Src/api/retriver.jsx

```
3 import axios from "axios";
4 export default axios.create({baseURL: "https://chatbot-fast-server.herokuapp.com",});
```

Src/api/tranlator.jsx

```
3 import api from "./retriver"
4
5 const get_translation = async (message,language) => {
6   const response = await api.get("/translater",{
7     params : {
8       msg: ''.concat(message),
9       lang: language
10    }})
11   return response;
12 }
13
14 export default get_translation
```

Src/loginpage/signout.jsx

```
4 √ import React from 'react';
5   import {useNavigate} from 'react-router-dom'
6
7 √ export default function SignOut() {
8   const navigate = useNavigate()
9 √   return(
10 √     <>
11       <div>Home Page</div>
12       <button onClick={() => navigate(-1)}>Signin</button>
13     </>
14   );
15 }
16
```

Src/loginpage/styles.css

```
3 √ .app {
4   display: flex; align-items: center;
5   justify-content: center; flex-direction: column;
6   gap: 20px; height: 90vh;
7   font-family: Consolas;
8   background-color: ■#282c34;
9 }
```

```
11  input[type="text"],  
12  input[type="password"] {  
13    height: 25px;  
14    border: 1px solid #rgba(0, 0, 0, 0.2);  
15    font-family: Consolas;  
16  }  
17  
18  input[type="submit"] {  
19    margin-top: 10px;  
20    cursor: pointer;  
21    font-size: 15px;  
22    background: #2898ec;  
23    border: 1px solid #2898ec;  
24    color: #fff;  
25    padding: 10px 20px;  
26    font-family: Consolas;  
27  }  
28  
29  input[type="submit"]:hover {  
30    background: #rgb(36, 116, 190);  
31  }  
32  
33  .button-container {  
34    display: flex;  
35    justify-content: center;  
36  }  
37  
38  .login-form {  
39    background-color: #white;  
40    padding: 2rem;  
41    box-shadow: 0 4px 8px 0 #rgba(0, 0, 0, 0.2),  
42    0 6px 20px 0 #rgba(0, 0, 0, 0.19);  
43  }  
44  
45  .list-container {  
46    display: flex;  
47  }  
48  
49  .error {  
50    color: red; font-size: 12px;  
51  }
```

```
53  .title {
54    font-size: 22px;
55    margin-bottom: 20px;
56  }
57
58  .input-container {
59    display: flex;
60    flex-direction: column;
61    gap: 8px;
62    margin: 10px;
63  }
```

Src/loginpage/signin.jsx

```
3  import React, { useState } from "react";
4  import {useNavigate} from 'react-router-dom'
5
6  import "./styles.css";
7
8  function SignIn() {
9
10  const navigate = useNavigate()
11  const [errorMessages, setErrorMessages] = useState({});
12  const [isSubmitted, setIsSubmitted] = useState(false);
13
14  const database = [
15    {
16      username: "user1@lenskart.in",
17      password: "pass1"
18    },
19    {
20      username: "user2@lenskart.in",
21      password: "pass2"
22    },
23    {
24      username: "user3@lenskart.in",
25      password: "pass3"
26    }
27  ];
```

```

29  | const errors = {
30  |   uname: "invalid username",
31  |   pass: "invalid password"
32  | };
33
34  | const handleSubmit = (event) => {
35  |   event.preventDefault();
36  |   var { uname, pass } = document.forms[0];
37  |   const userData = database.find((user) =>
38  |     user.username === uname.value);
39
40  |   if (userData) {
41  |     if (userData.password !== pass.value) {
42  |       setErrorMessages({ name: "pass",
43  |         message: errors.pass });
44  |     } else {
45  |       setIsSubmitted(true);
46  |     }
47  |   } else {
48  |     setErrorMessages({ name: "uname",
49  |       message: errors.uname });
50  |   }
51  | };
52
53  | const renderErrorMessage = (name) =>
54  |   name === errorMessage.name && (
55  |     <div className="error">{errorMessage.message}</div>
56  |   );
57
58  | const renderForm = (
59  |   <div className="form">
60  |     <form onSubmit={handleSubmit}>
61  |       <div className="input-container">
62  |         <label>Username </label>
63  |         <input type="text" name="uname" required />
64  |         {renderErrorMessage("uname")}
65  |       </div>
66  |       <br><br>
67  |       <div className="input-container">
68  |         <label>Password </label>
69  |         <input type="password" name="pass" required />
70  |         {renderErrorMessage("pass")}

```

```

71     </div>
72     <br><br>
73     <div className="button-container">
74       <input type="submit" />
75     </div>
76   </form>
77 </div>
78 );
79
80 return (
81   <div className="app">
82     <div className="login-form">
83       <div className="title">Login to Your Account</div><br><br>
84       {isSubmitted ? navigate('chatbot') : renderForm}
85     </div>
86   </div>
87 );
88 }
89
90 export default SignIn;

```

Src/components/main.css

```

3   .options-container {
4     display: flex;
5     flex-wrap: wrap;
6   }
7
8   .order-options-container{
9     display: flex;
10    flex-wrap: wrap;
11  }
12
13  .option-button {
14    margin: 5px; border-radius: 25px;
15    max-width: 210px; padding: 8px;
16    border-color: #2898ec; background: transparent;
17    text-align: center; font-family: "Consolas";
18  }

```

```

20  .option-button {
21    transition-duration: 0.1s;
22  }
23
24  .option-button:hover {
25    background-color: #2898ec;
26    color: white;
27  }
28
29  .option-button-ok {
30    margin: 5px;
31    border-radius: 25px;
32    max-width: 210px;
33    padding: 8px;
34    border-color: #2898ec;
35    background: transparent;
36    font-family: "Consolas";
37  }
38
39  .option-button-ok {
40    transition-duration: 0.1s;
41  }
42
43  .option-button-ok:hover {
44    background-color: #2898ec;
45    color: white;
46  }
47
48  .pincode-text {
49    width : 80px
50 }

```

Src/components/options.jsx

```

2  import React from "react";
3  import {useState,useEffect} from "react";
4  import api from "../api/retriver"
5  import a from "../api/language"
6  import "./main.css";

```

```

8  const text1 = [
9    "1. Steps to place an order online or offline stores",
10   "2. Finding nearest Lenskart Store in your surroundings",
11   "3. Cancellation of Ordered Items with order number",
12   "4. Making an appointment for a home eye test and try-on",
13   "5. Select different input and display language",
14   "6. Logout from your lenskart account"
15 ]
16
17 const Options = (props) => {
18   const [text,setText] = useState([])
19   const get_translation = async (message,language) => {
20     const response = await api.get("/translation",{
21       params : {
22         msg: ''.concat(message),
23         lang: language
24       })
25     return setText(response.data);
26   };
27
28   useEffect(() => {get_translation(text1,a.getLanguage()),[]});
29
30   var text_0 = text[0]
31   var text_1 = text[1]
32   var text_2 = text[2]
33   var text_3 = text[3]
34   var text_4 = text[4]
35   var text_5 = text[5]
36
37   const options = [
38     {
39       text: text_0,
40       handler: props.actionProvider.starting_option_1,
41       id: 1,
42     },
43     {
44       text: text_1,
45       handler: props.actionProvider.starting_option_2,
46       id: 2,
47     },

```

```
48  {
49    text: text_2,
50    handler: props.actionProvider.starting_option_3,
51    id: 3,
52  },
53  {
54    text: text_3,
55    handler: props.actionProvider.starting_option_4,
56    id: 4,
57  },
58  {
59    text: text_4,
60    handler: props.actionProvider.starting_option_5,
61    id: 5,
62  },
63  {
64    text: text_5,
65    handler: props.actionProvider.starting_option_6,
66    id: 6,
67  }
68];
69
70 const buttonsMarkup = options.map((option) => (
71   <button key={option.id} onClick={option.handler}
72     className="option-button">
73     {option.text}
74   </button>
75));
76
77 return <div className="options-container">{buttonsMarkup}</div>;
78};
79
80 export default Options;
81
```

Src/components/options1.jsx

```
3  import React from "react";
4  import {useState,useEffect} from "react";
5  import api from "../api/retriver"
6  import a from "../api/language"
7  import "./main.css";
8
9  const text11 = [
10    "1.Place an order online",
11    "2.Place an order offline"
12  ]
13
14 const Options1 = (props) => {
15   const [text1,setText] = useState({})
16   const get_translation = async (message,language) => {
17     const response = await api.get("/translation",{
18       params : {
19         msg: ''.concat(message),
20         lang: language
21       })
22     return setText(response.data);
23   };
24   useEffect(() => {get_translation(text11,a.getLanguage()),[]});
25   const options= [
26     {
27       text: text1[0],
28       handler: props.actionProvider.place_an_order_online,
29       id: 1,
30     },
31     {
32       text: text1[1],
33       handler: props.actionProvider.place_an_order_offline,
34       id: 2,
35     },
36   ];
37
38
39   const buttonsMarkup = options.map((option) => (
40     <button key={option.id} onClick={option.handler} className="option-button">
41       {option.text}
42     </button>
43   )));
44   return <div className="order-options-container">{buttonsMarkup}</div>;
45 }
46 export default Options1;      You, 4 weeks ago • Initial commit ...
```

Src/components/options3.jsx

```
3  import React from "react";
4  import api from "../api/retriver"
5  import a from "../api/language"
6  import "./main.css";
7  import {useState,useEffect} from "react";
8
9  const text1 = [
10    "1. Yes",
11    "2. No"
12  ]
13
14 const Options3 = (props) =>
15 {
16   const [text,setText] = useState({})
17   const get_translation = async (message,language) => {
18     const response = await api.get("/translation",{
19       params : {
20         msg: ''.concat(message),
21         lang: language
22       })
23     return setText(response.data);
24   };
25   useEffect(() => {get_translation(text1,a.getLanguage()),[]});
26   const options = [
27     {
28       text: text[0],
29       handler: props.actionProvider.cancel_order_yes,
30       id: 1,
31     },
32     {
33       text: text[1],
34       handler: props.actionProvider.cancel_order_no,
35       id: 2,
36     }
37   ];
38   const buttonsMarkup = options.map((option) => (
39     <button key={option.id} onClick={option.handler} className="option-button">
40       {option.text}
41     </button>
42   )));
43   return <div className="options-container">{buttonsMarkup}</div>;
44 }
45 export default Options3;
```

Src/components/options4.jsx

```
3  import React from "react";
4  import api from "../api/retriver"
5  import a from "../api/language"
6  import "./main.css";
7  import {useState,useEffect} from "react";
8
9  const text1 = [
10    "1. Home Eye Test",
11    "2. Try On"
12  ]
13
14  const Options4 = (props) =>
15  {
16    const [text,setText] = useState({})
17    const get_translation = async (message,language) => {
18      const response = await api.get("/translation",{
19        params : {
20          msg: ''.concat(message),
21          lang: language
22        })
23      return setText(response.data);
24    };
25    useEffect(() => {get_translation(text1,a.getLanguage()),[]});
26    const options = [
27      {
28        text: text[0],
29        handler: props.actionProvider.home_eye_test,
30        id: 1,
31      },
32      {
33        text: text[1],
34        handler: props.actionProvider.try_on,
35        id: 2,
36      }
37    ];
38    const buttonsMarkup = options.map((option) => (
39      <button key={option.id} onClick={option.handler} className="option-button">
40        {option.text}
41      </button>
42    )));
43    return <div className="options-container">{buttonsMarkup}</div>;
44  }
45  export default Options4;
```

Src/components/options5.jsx

```
3  import React from "react";
4  import api from "../api/retriver"
5  import a from "../api/language"
6  import "./main.css";
7  import {useState,useEffect} from "react";
8
9  const text1 = [
10    "1. Telugu",
11    "2. Hindi",
12    "3. English",
13    "4. Tamil",
14    "5. kannada"
15  ]
16
17 const Options5 = (props) =>
18 {
19   const [text,setText] = useState({})
20
21   const get_translation = async (message,language) => {
22     const response = await api.get("/translation",{
23       params : {
24         msg: ''.concat(message),
25         lang: language
26       })
27     return setText(response.data);
28   };
29
30   useEffect(() => {get_translation(text1,a.getLanguage()),[]});
31
32   const options = [
33     {
34       text: text[0],
35       handler: props.actionProvider.language_option_1,
36       id: 1,
37     },
38     {
39       text: text[1],
40       handler: props.actionProvider.language_option_2,
41       id: 2,
42     },
43   ]
44 }
```

```
43  <div>
44    <button key={option.id} onClick={option.handler}
45      className="option-button">
46        {option.text}
47      </button>
48    );
49  );
50  <div className="options-container">
51    {buttonsMarkup}</div>;
52  );
53  <div>
54    <button key={option.id} onClick={option.handler}
55      className="option-button">
56        {option.text}
57      </button>
58    );
59  );
60  const buttonsMarkup = options.map((option) => (
61    <button key={option.id} onClick={option.handler}
62      className="option-button">
63        {option.text}
64      </button>
65  )));
66  );
67  return <div className="options-container">
68    {buttonsMarkup}</div>;
69  );
70  );
71  export default Options5;
```

Src/chatbot/main.css

```
3   .App {
4     text-align: center;
5     background-color: #282c34;
6     display: flex;
7     justify-content: center;
8     align-items: center;
9   }
10
11 .react-chatbot-kit-user-chat-message-container {
12   display: flex;
13   margin: 15px 0;
14   justify-content: flex-end;
15   font-family: "Consolas";
16 }
17
18 .react-chatbot-kit-user-avatar-container {
19   width: 40px;
20   height: 40px;
21   border-radius: 50%;
22   margin-left: 12.5px;
23   background-color: #3d4e8d;
24   display: flex;
25   justify-content: center;
26   align-items: center;
27 }
28 .react-chatbot-kit-user-avatar-icon {
29   fill: #fff;
30   width: 15px;
31   height: 15px;
32 }
33
34
35 .react-chatbot-kit-user-chat-message {
36   background-color: #2898ec;
37   padding: 10px;
38   border-radius: 5px;
39   font-size: 0.9rem;
40   color: white;
41   font-weight: medium;
42   position: relative;
43   text-align: left;
44 }
```

```
46 .react-chatbot-kit-user-chat-message-arrow {  
47   width: 0;  
48   height: 0;  
49   border-top: 8px solid transparent;  
50   border-bottom: 8px solid transparent;  
51   border-left: 8px solid #2898ec;  
52   position: absolute;  
53   right: -7px;  
54   top: 13px;  
55 }  
56  
57 .react-chatbot-kit-chat-bot-message-container {  
58   display: flex;  
59   margin: 15px 0;  
60   font-family : "Consolas";  
61   justify-content: flex-start;  
62 }  
63  
64 .react-chatbot-kit-chat-bot-avatar-container {  
65   width: 40px;  
66   height: 40px;  
67   border-radius: 50%;  
68   font-size: 1.2rem;  
69   margin-right: 12.5px;  
70   background-color: #d8d8d8;  
71   display: flex;  
72   justify-content: center;  
73   align-items: center;  
74 }  
75  
76 .react-chatbot-kit-chat-bot-avatar-icon {  
77   fill: #494646;  
78   width: 22.5px;  
79   height: 22.5px;  
80 }  
81  
82 .react-chatbot-kit-chat-bot-avatar-letter {  
83   color: #1d1d1d;  
84   margin: 0;  
85   padding: 0;  
86 }
```

```
88  .react-chatbot-kit-chat-bot-message {
89    background-color: #2898ec;
90    padding: 10px;
91    border-radius: 5px;
92    font-size: 0.9rem;
93    color: #fff;
94    font-weight: medium;
95    position: relative;
96    width: 184.5px;
97    margin-left: auto;
98    text-align: left;
99  }
100
101 .react-chatbot-kit-chat-bot-message-arrow {
102   width: 0;
103   height: 0;
104   border-top: 8px solid transparent;
105   border-bottom: 8px solid transparent;
106   border-right: 8px solid #2898ec;
107   position: absolute;
108   left: -7px;
109   top: 13px;
110 }

112 .react-chatbot-kit-chat-bot-loading-icon-container {
113   height: 17px;
114   width: 25px;
115 }
116
117 .chatbot-loader-container {
118   display: flex;
119   width: 100%;
120   justify-content: center;
121 }
122
123 #chatbot-loader #chatbot-loader-dot1 {
124   animation: load 1s infinite;
125 }
126
127 #chatbot-loader #chatbot-loader-dot2 {
128   animation: load 1s infinite;
129   animation-delay: 0.2s;
130 }
```

Src/components/chatbot.jsx

```
4  import React from "react";
5
6  import config from "./config";
7  import ActionProvider from "./ActionProvider";
8  import MessageParser from "./MessageParser";
9
10 import Chatbot from "react-chatbot-kit";
11
12 import './main.css'
13
14 function ChatBot() {
15   return (
16     <div className="App">
17       <div>
18         <Chatbot config={config} actionProvider={ActionProvider}
19           messageParser={MessageParser} />
20       </div>
21     </div>
22   );
23 }
24
25 export default ChatBot
```

Src/components/chathead.jsx

```
1  import React from "react";
2  import "./main.css";
3  import Capture from './images/Capture.PNG';
4
5  function ChatHead() {
6    return (
7      <div className="chat-header">
8        <img src={Capture} width={300}
9          height={75} alt="Logo" />
10      </div> );
11    }
12
13
14 export default ChatHead
```

Src/components/config.js

```
4  import React from "react";
5  import { createChatBotMessage } from "react-chatbot-kit";
6  import ChatHead from "./ChatHead"
7  import Options from "../components/Options";
8  import Options1 from "../components/Options1";
9  import Options3 from "../components/Options3";
10 import Options5 from "../components/Options5";
11 import Options4 from "../components/Options4";
12
13 const config = {
14   botName: "Sara",
15   initialMessages: [
16     createChatBotMessage("Hello, I'm Sara. How may I help you today ",
17       {widget: "options",}),
18   ],
19   widgets: [
20     {
21       widgetName: "options",
22       widgetFunc: (props) => <Options {...props} />
23     },
24     {
25       widgetName: "options1",
26       widgetFunc: (props) => <Options1 {...props} />
27     },
28     {
29       widgetName: "options5",
30       widgetFunc: (props) => <Options5 {...props} />
31     },
32     {
33       widgetName: "options3",
34       widgetFunc: (props) => <Options3 {...props} />
35     },
36     {
37       widgetName: "options4",
38       widgetFunc: (props) => <Options4 {...props} />
39     }
40   ],
41   customComponents: {header: () => <ChatHead />},
42 };
43 export default config;
```

Src/components/messageparser.js

```
3  class MessageParser {
4    constructor(actionProvider) {
5      this.actionProvider = actionProvider;
6    }
7    parse = (message) => {
8      const lowercase = message.toLowerCase();
9      if(lowercase.includes("start")){
10        this.actionProvider.start_again();
11      }
12      else if(lowercase.includes("5000")){
13        this.actionProvider.locate_store(lowercase);
14      }
15      else if(lowercase.includes("1000")){
16        this.actionProvider.cancel_order(lowercase);
17      }
18      else{
19        this.actionProvider.warning_1(lowercase);
20      }
21    }
22    parser = (message) => {
23      const lowercase = message.toLowerCase();
24      if(lowercase.includes("start")){
25        this.actionProvider.start_again();
26      }
27      else if(lowercase.includes("5000")){
28        this.actionProvider.locate_store(lowercase);
29      }
30      else if(lowercase.includes("1000")){
31        this.actionProvider.cancel_order(lowercase);
32      }
33      else if (lowercase.includes("yes")){
34        this.actionProvider.cancel_order_yes();
35      }
36      else if (lowercase.includes("no")){
37        this.actionProvider.cancel_order_no();
38      }
39    }
40  }
41  export default MessageParser;
```

Src/components/actionprovider.js

```
2  < import a from "../api/language"
3  import api from "../api/retriver"
4  import {useNavigate} from 'react-router-dom'
5  import firebase from 'firebase/compat/app';
6  import 'firebase/compat/firestore';
7  import 'firebase/compat/auth';
8  import 'firebase/compat/analytics';
9
10 < firebase.initializeApp({
11   apiKey: "AIzaSyCjw7uSk1JHtymgBPRDWJs2W0EB2vIaB0A",
12   authDomain: "project-7f27d.firebaseio.com",
13   projectId: "project-7f27d",
14   storageBucket: "project-7f27d.appspot.com",
15   messagingSenderId: "845652508520",
16   appId: "1:845652508520:web:f1dc91daf2cd3dd9f1e0dd",
17   measurementId: "G-9YLWFK2X5K"
18 })
19
20 const firestore = firebase.firestore();
21 const messagesRef = firestore.collection('messages');
22
23
24 < const text1 = [
25   "1. Steps to place an order online or offline stores",
26   "2. Finding nearest Lenskart Store in your surroundings",
27   "3. Cancellation of Ordered Items with order number",
28   "4. Making an appointment for a home eye test and try-on",
29   "5. Select different input and display language",
30   "6. Logout from your lenskart account"
31 ]
32
33 < class ActionProvider {
34   constructor(createChatBotMessage, setStateFunc ,createClientMessage) {
35     this.createChatBotMessage = createChatBotMessage;
36     this.createClientMessage = createClientMessage;
37     this.setState = setStateFunc;
38     this.ids = ["100011","100012","100013","100014","100015"]
39     this.order_id = ""
40   }
41 }
```

```

41   get_translation = async (message,language) => {
42     const response = await api.get("/translater",{
43       params : {
44         msg: message,
45         lang: language
46       }})
47       return response;
48     };
49
50   get_message = async (message,language) => {
51     const response = await api.get("/get_response",{
52       params : {
53         msg: message,
54         lang: language
55       }})
56       return response;
57     };
58
59   start_again = () => {
60     this.get_translation("Hello, I'm Sara. How may I help you today ?"+
61     "Do you need help with: ",a.getLanguage()).then(async (response) => {
62       const message = this.createChatBotMessage(response.data, {widget: "options",});
63       this.addMessageToState1(message)
64       await messagesRef.add({
65         text: "start again",
66         createdAt: firebase.firestore.FieldValue.serverTimestamp()
67       })
68     })
69   };
70
71   warning_1 = (lowercase) => {
72     this.get_message(lowercase,a.getLanguage()).then(async (response) => {
73       const message = this.createChatBotMessage(response.data);
74       this.addMessageToState(message)
75       await messagesRef.add({
76         text: lowercase,
77         createdAt: firebase.firestore.FieldValue.serverTimestamp()
78       })
79     })
80   };
81
82   starting_option_1 = () => {
83     this.get_translation(text1[0],a.getLanguage()).then(async (response) => {
84       const message = this.createClientMessage(response.data, {widget: "options1",});
85       this.addMessageToState(message);
86       await messagesRef.add({
87         text: response.data,
88         createdAt: firebase.firestore.FieldValue.serverTimestamp()
89       })
90     })
91   };

```

```

93  |     starting_option_2 = () => {
94  |       var lang_str = ''.concat([text1[1], "Please enter your area pincode number"])
95  |       this.get_translation(lang_str,a.getLanguage()).then(async (response) => {
96  |         const msg = response.data.split(",")
97  |         var message = this.createClientMessage(msg[0])
98  |         this.addMessageToState(message);
99  |         message = this.createChatBotMessage(msg[1]);
100 |         this.addMessageToState(message);
101 |         await messagesRef.add({
102 |           text: response.data,
103 |           createdAt: firebase.firestore.FieldValue.serverTimestamp()
104 |         })
105 |       })
106 |     );
107 |
108 |     starting_option_3 = () => {
109 |       var lang_str = ''.concat([text1[2], "Please enter the order-id for your order"])
110 |       this.get_translation(lang_str,a.getLanguage()).then(async (response) => {
111 |         const msg = response.data.split(",")
112 |         var message = this.createClientMessage(msg[0]);
113 |         this.addMessageToState(message);
114 |         message = this.createChatBotMessage(msg[1]);
115 |         this.addMessageToState(message);
116 |         await messagesRef.add({
117 |           text: response.data,
118 |           createdAt: firebase.firestore.FieldValue.serverTimestamp()
119 |         })
120 |       })
121 |     };
122 |
123 |     starting_option_4 = () => {
124 |       var lang_str = ''.concat([text1[3], "Please select one of 2 options"])
125 |       this.get_translation(lang_str,a.getLanguage()).then(async (response) => {
126 |         const msg = response.data.split(",")
127 |         var message = this.createClientMessage(msg[0]);
128 |         this.addMessageToState(message);
129 |         message = this.createChatBotMessage(msg[1], {widget: "options4",});
130 |         this.addMessageToState(message);
131 |         await messagesRef.add({
132 |           text: response.data,
133 |           createdAt: firebase.firestore.FieldValue.serverTimestamp()
134 |         })
135 |       })
136 |     };

```

```

138     starting_option_5 = () => {
139       var lang_str = ''.concat([text1[4], "Please Select any below Language"])
140       this.get_translation(lang_str,a.getLanguage()).then(async (response) => {
141         const msg = response.data.split(",")
142         var message = this.createClientMessage(msg[0]);
143         this.addMessageToState(message);
144         message = this.createChatBotMessage(msg[1], {widget: "options5",});
145         this.addMessageToState(message);
146         await messagesRef.add({
147           text: response.data,
148           createdAt: firebase.firestore.FieldValue.serverTimestamp()
149         })
150       })
151     };
152
153     starting_option_6 = () => {
154       this.get_translation(text1[5],a.getLanguage()).then(async (response) => {
155         const message = this.createChatBotMessage(response.data);
156         this.addMessageToState(message)
157         a.setLanguage(3);
158         await messagesRef.add({
159           text: response.data,
160           createdAt: firebase.firestore.FieldValue.serverTimestamp()
161         })
162         this.navigate('/')
163       })
164     };
165
166
167     place_an_order_online = () => {
168       let request_string = "1.Place an order online ; Please contact our nearby "+
169       "store and make an order or please contact to our customer care 2014587400 "+
170       "or visit store.lenskart.com for details "
171       this.get_translation(request_string,a.getLanguage()).then(async (response) => {
172         const msg = response.data.split(";");
173         var message = this.createClientMessage(msg[0]);
174         this.addMessageToState(message)
175         message = this.createChatBotMessage(msg[1]);
176         this.addMessageToState(message)
177         await messagesRef.add({
178           text: "Place an order online",
179           createdAt: firebase.firestore.FieldValue.serverTimestamp()
180         })
181       })
182     );
183   };

```

```

185   place_an_order_offline = () => {
186     let request_string = "2.Place an order offline ; Please contact our nearby store "+  

187     "and make an order or please contact to our customer care 2014587400 or visit"+  

188     " store.lenskart.com for details "
189     this.get_translation(request_string,a.getLanguage()).then(async (response) => {
190       const msg = response.data.split(";");
191       var message = this.createClientMessage(msg[0]);
192       this.addMessageToState(message)
193       message = this.createChatBotMessage(msg[1]);
194       this.addMessageToState(message)
195       await messagesRef.add({
196         text: "Place an order offline",
197         createdAt: firebase.firestore.FieldValue.serverTimestamp()
198       })
199     })
200   };
201
202
203
204   locate_store = (lowercase) => {
205     //const lowercase = getPincode();
206     const get_translation = async (message,language) => {
207       const response = await api.get("/address",{
208         params : {
209           pincode: message,
210           lang: language
211         })
212       return response
213     };
214     get_translation(lowercase,a.getLanguage()).then(async (response) => {
215       const message = this.createChatBotMessage(response.data)
216       this.addMessageToState(message)
217       await messagesRef.add({
218         text: lowercase,
219         createdAt: firebase.firestore.FieldValue.serverTimestamp()
220       })
221     })
222   }
223
224
225
226   cancel_order = (lowercase) => {
227     if (this.ids.includes(lowercase)) {
228       this.order_id = lowercase
229       this.get_translation("Do you want to cancel the order , "+  

230       "type yes or no",a.getLanguage()).then(async (response) => {
231         const message = this.createChatBotMessage(response.data,  

232           {widget: "options3",})
233         this.addMessageToState(message)
234         await messagesRef.add({
235           text: lowercase,
236           createdAt: firebase.firestore.FieldValue.serverTimestamp()
237         })
238       })
239     }

```

```

240     }
241     else{
242       this.get_translation("Please enter the correct order-id",
243       a.getLanguage()).then(async (response) => {
244         const message = this.createChatBotMessage(response.data);
245         this.addMessageToState(message);
246       })
247     }
248   cancel_order_yes = () => {
249     var index = this.ids.indexOf(this.order_id)
250     delete this.ids[index];
251     this.get_translation("Order deleted Sucessfully",
252     a.getLanguage()).then(async (response) => {
253       const message = this.createChatBotMessage(response.data)
254       this.addMessageToState(message)
255       await messagesRef.add({
256         text: "yes",
257         createdAt: firebase.firestore.FieldValue.serverTimestamp()
258       })
259     })
260   }
261   cancel_order_no = () => {
262     this.get_translation("Cancellation Process terminated Sucessfully",
263     a.getLanguage()).then(async (response) => {
264       const message = this.createChatBotMessage(response.data)
265       this.addMessageToState(message)
266       await messagesRef.add({
267         text: "no",
268         createdAt: firebase.firestore.FieldValue.serverTimestamp()
269       })
270     })
271   }
272 }
273 home_eye_test = () => {
274   var lang_str = ''.concat(["1. Home eye test", "Please contact our Customer "+
275   "service centre to get an appointment - contact 180-100-1111"])
276   this.get_translation(lang_str,a.getLanguage()).then(async (response) => {
277     const msg = response.data.split(",")
278     var message = this.createClientMessage(msg[0])
279     this.addMessageToState(message)
280     message = this.createChatBotMessage(msg[1])
281     this.addMessageToState(message)
282
283
284

```

```

285     |     |     await messagesRef.add({
286     |     |     |     text: "home eye test",
287     |     |     |     createdAt: firebase.firestore.FieldValue.serverTimestamp()
288     |     |     })
289     |     |
290     |     }
291     |
292     try_on = () => {
293     |     var lang_str = ''.concat(["2. Try on", "This feature is available in android "+
294     |     |     "or the ios app , please download the app to see the feature"])
295     |     this.get_translation(lang_str,a.getLanguage()).then(async (response) => {
296     |     |     const msg = response.data.split(",")
297     |     |     var message = this.createClientMessage(msg[0])
298     |     |     this.addMessageToState(message)
299     |     |     message = this.createChatBotMessage(msg[1])
300     |     |     this.addMessageToState(message)
301     |     |     await messagesRef.add({
302     |     |     |     text: "try on",
303     |     |     |     createdAt: firebase.firestore.FieldValue.serverTimestamp()
304     |     |     })
305     |     |     })
306     |     }

310     language_option_1 = () => {
311     |     a.setLanguage(1);
312     |     console.log(a.getLanguage())
313     |     this.get_translation("Language changed to telugu",a.getLanguage()).then(async (response) => {
314     |     |     const message = this.createChatBotMessage(response.data)
315     |     |     this.addMessageToState(message)
316     |     |     this.start_again()
317     |     |     await messagesRef.add({
318     |     |     |     text: "telugu",
319     |     |     |     createdAt: firebase.firestore.FieldValue.serverTimestamp()
320     |     |     })
321     |     |     })
322     |     }

323     language_option_2 = () => {
324     |     a.setLanguage(2);
325     |     console.log(a.getLanguage())
326     |     this.get_translation("Language changed to hindi",a.getLanguage()).then(async (response) => {
327     |     |     const message = this.createChatBotMessage(response.data)
328     |     |     this.addMessageToState(message)
329     |     |     this.start_again()
330     |     |     await messagesRef.add({
331     |     |     |     text: "hindi",
332     |     |     |     createdAt: firebase.firestore.FieldValue.serverTimestamp()
333     |     |     })
334     |     |     })
335     |     }
336   }

```

```
338     language_option_3 = () => {
339         a.setLanguage(3);
340         console.log(a.getLanguage())
341         this.get_translation("Language changed to english",
342             a.getLanguage()).then(async (response) => {
343                 const message = this.createChatBotMessage(response.data)
344                 this.addMessageToState(message)
345                 this.start_again()
346                 await messagesRef.add({
347                     text: "english",
348                     createdAt: firebase.firestore.FieldValue.serverTimestamp()
349                 })
350             })
351         }
352
353     language_option_4 = () => {
354         a.setLanguage(4);
355         console.log(a.getLanguage())
356         this.get_translation("Language changed to tamil",
357             a.getLanguage()).then(async (response) => {
358                 const message = this.createChatBotMessage(response.data)
359                 this.addMessageToState(message)
360                 this.start_again()
361                 await messagesRef.add({
362                     text: "tamil",
363                     createdAt: firebase.firestore.FieldValue.serverTimestamp()
364                 })
365             })
366         }
367
368     language_option_5 = () => {
369         a.setLanguage(5);
370         console.log(a.getLanguage())
371         this.get_translation("Language changed to kannada",
372             a.getLanguage()).then(async (response) => {
373                 const message = this.createChatBotMessage(response.data)
374                 this.addMessageToState(message)
375                 this.start_again()
376                 await messagesRef.add({
377                     text: "kanadda",
378                     createdAt: firebase.firestore.FieldValue.serverTimestamp()
379                 })
380             })
381         }
```

```
383     addMessageToState = (message) => {
384       this.setState((prevState) => ({
385         ...prevState,
386         messages: [...prevState.messages, message],
387       }));
388     };
389
390     addMessageToState1 = (message) => {
391       this.setState((prevState) => ({
392         ...prevState,
393         messages: [message],
394       }));
395     };
396   }
397
398   export default ActionProvider;
399
```

5.3 Results and Output Screens

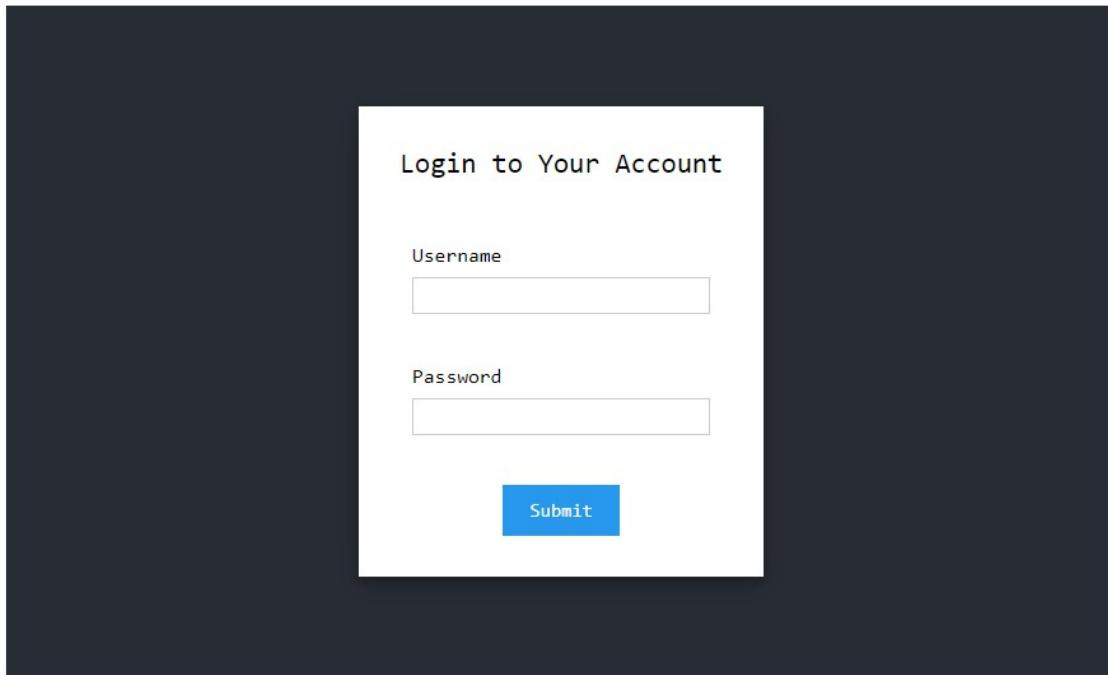


Fig 5.3.1 : login page

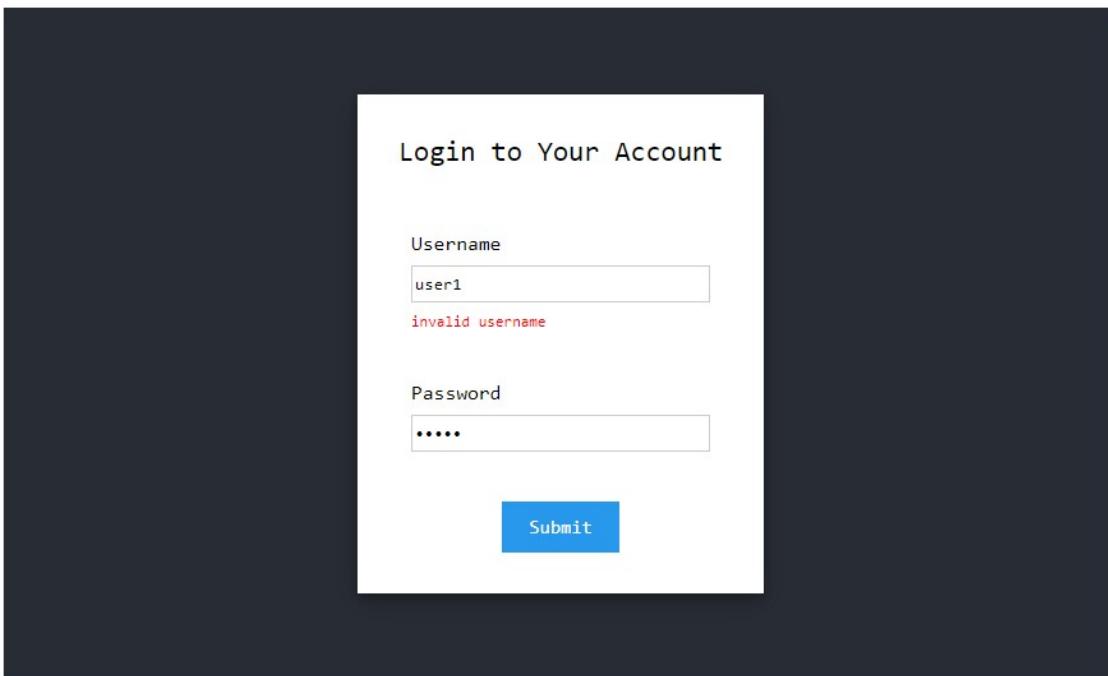


Fig 5.3.2 : invalid username

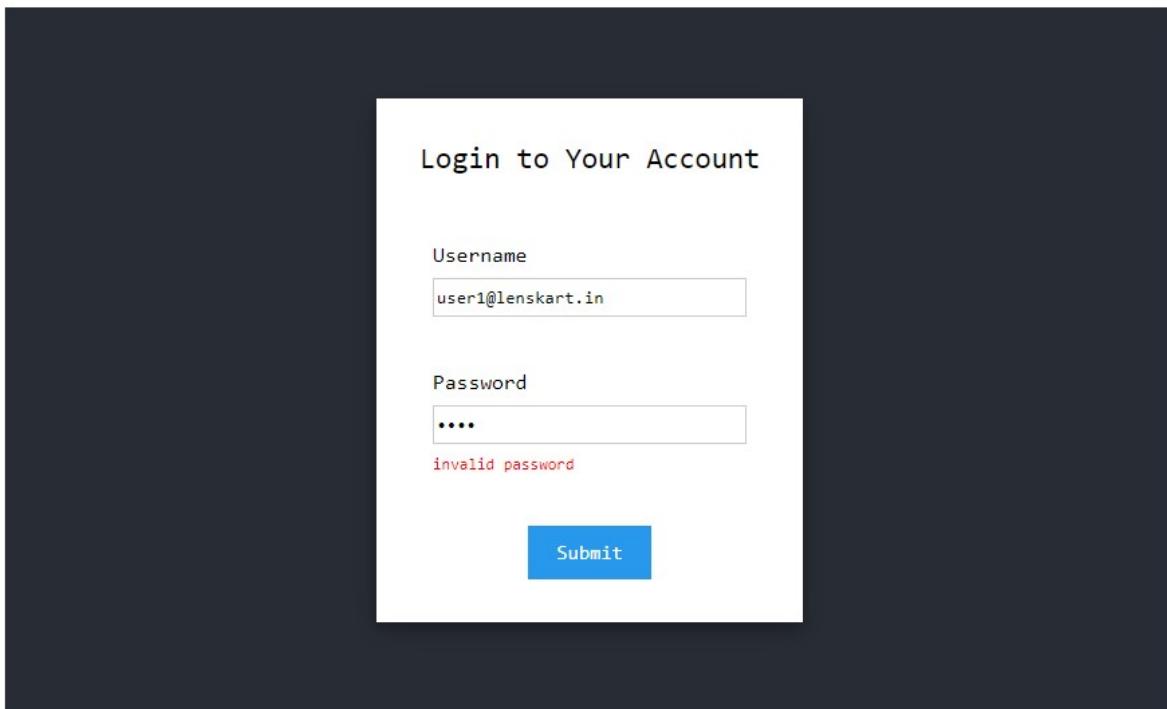


Fig 5.3.3 : invalid password

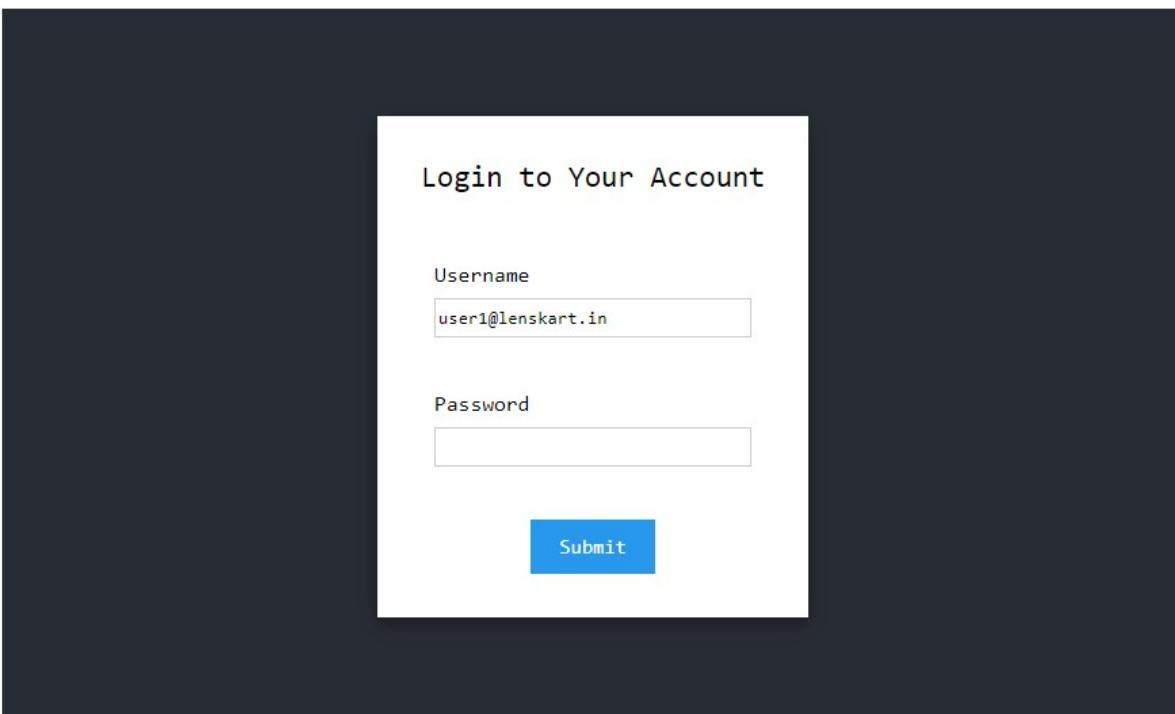


Fig 5.3.4 : after logging out from chatbot

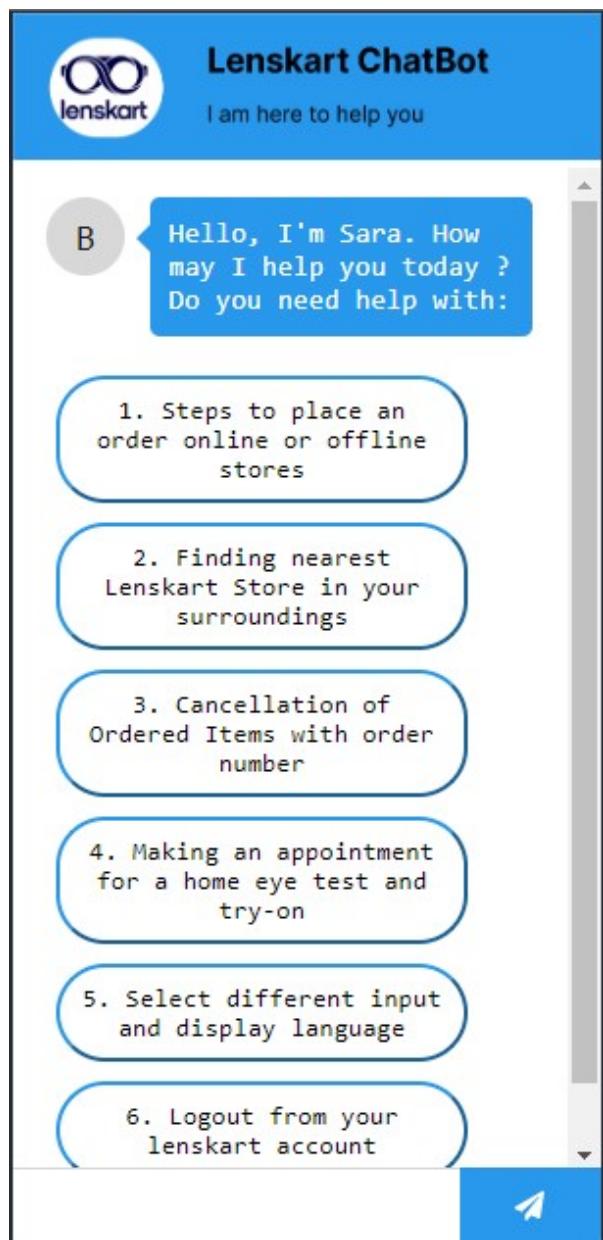


Fig 5.3.5 : chatbot initial page

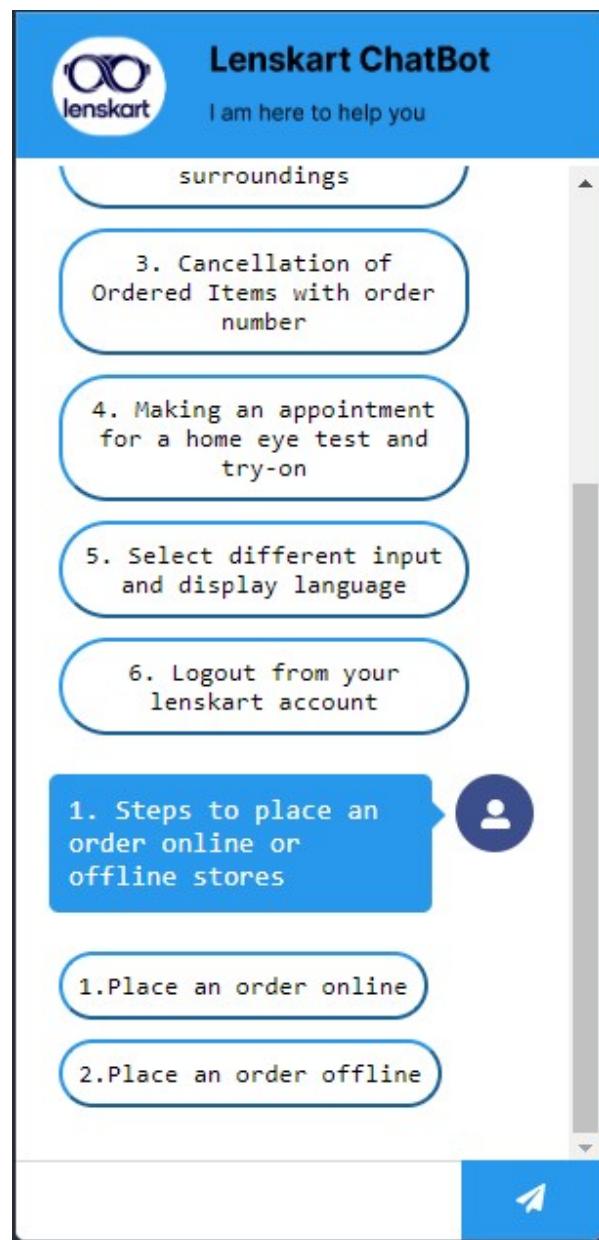


Fig 5.3.6 : after clicking the first option

This is the first initial page of the chatbot that is shown in the figure 5.3.5 , it consists of 6 different options and depending on the different options different menu and options popup and requests the user.if we click on the first option we will get the other two options as place an order online and place an order offline.

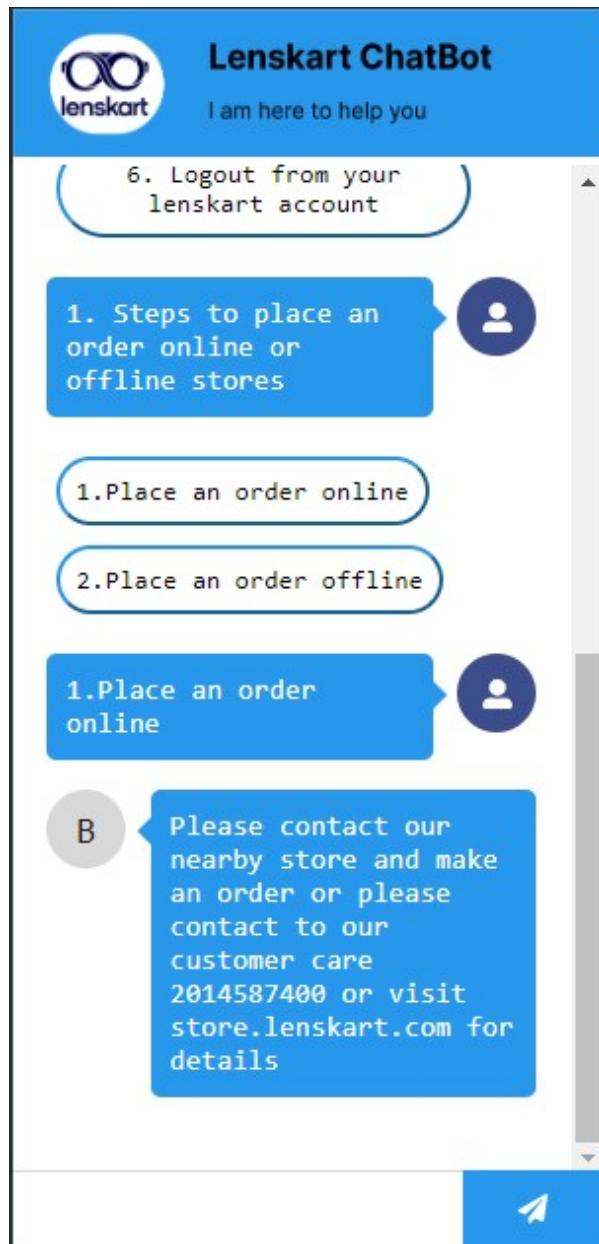


Fig 5.3.7 : place an order online

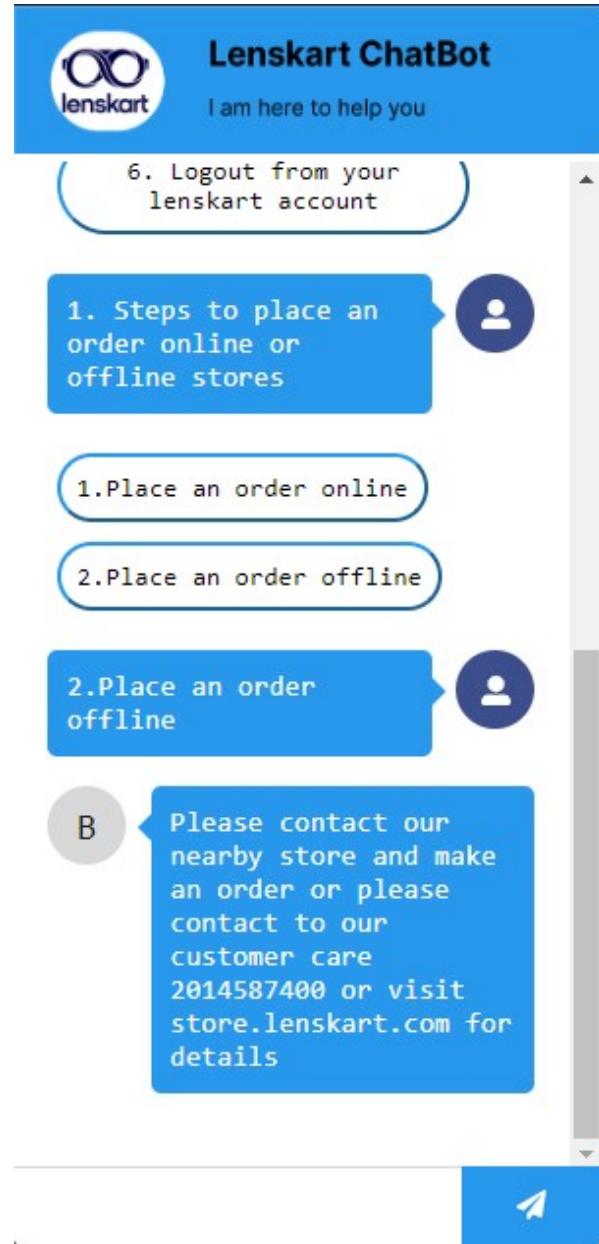


Fig 5.3.8 : place and order offline

After clicking on the two different options we get the respective results

As this project is mainly focussed on the language translation, the result for the both of them is as follows : “Please contact our nearby store and make an order or please contact to our customer care 2014587400 or visit store.lenskart.com for details”

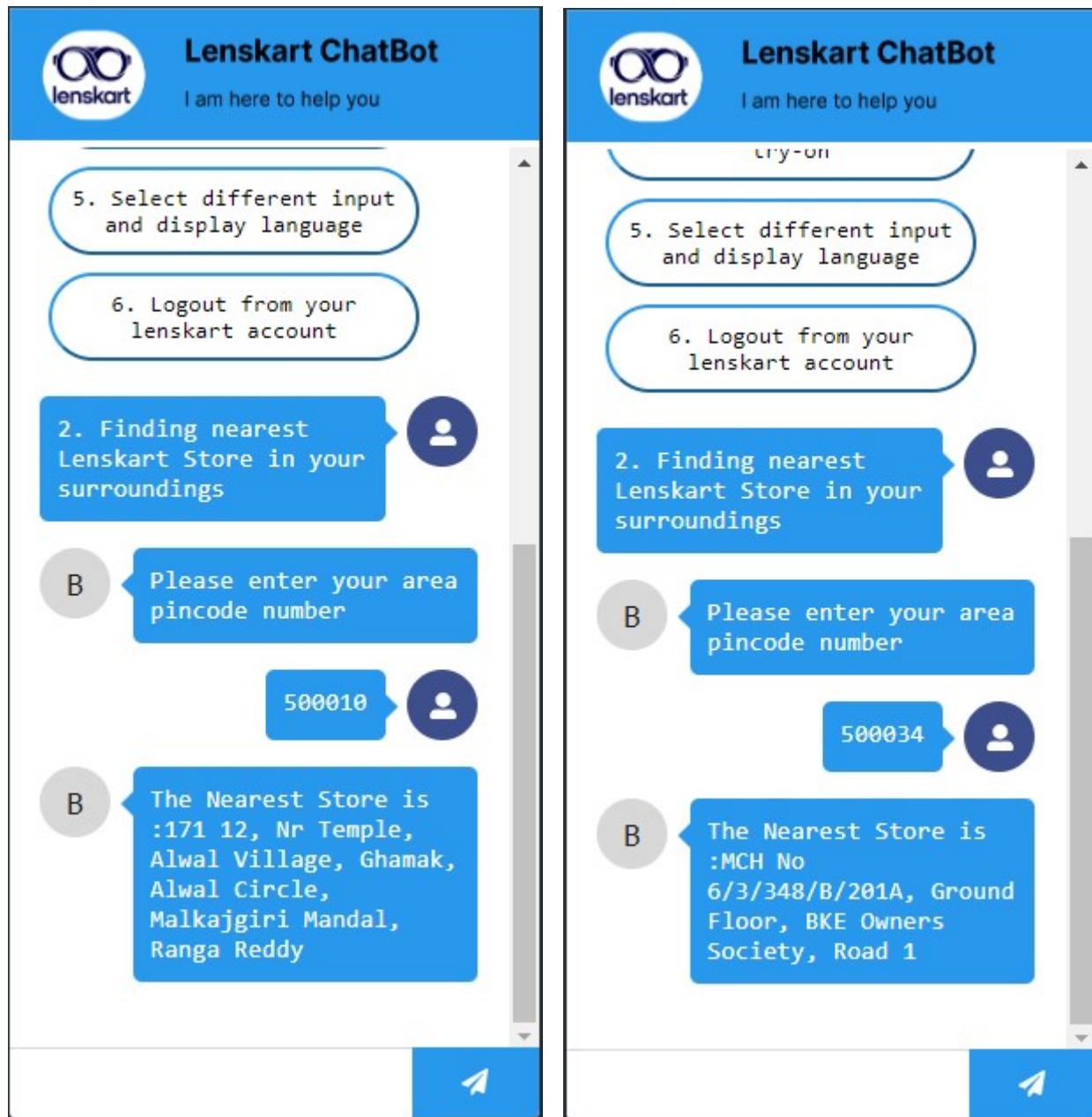


Fig 5.3.9 : find the nearest lenskart store-1

Fig 5.3.10 : find the nearest lenskart store-2

The next option is to find the nearest lenskart store and as the name suggests , It gives the address of the nearest lenskart store according to the pincode. As shown in figures 5.3.9 and 5.3.10 it gives the addresses of stores near pincodes , 500010 and 500034.

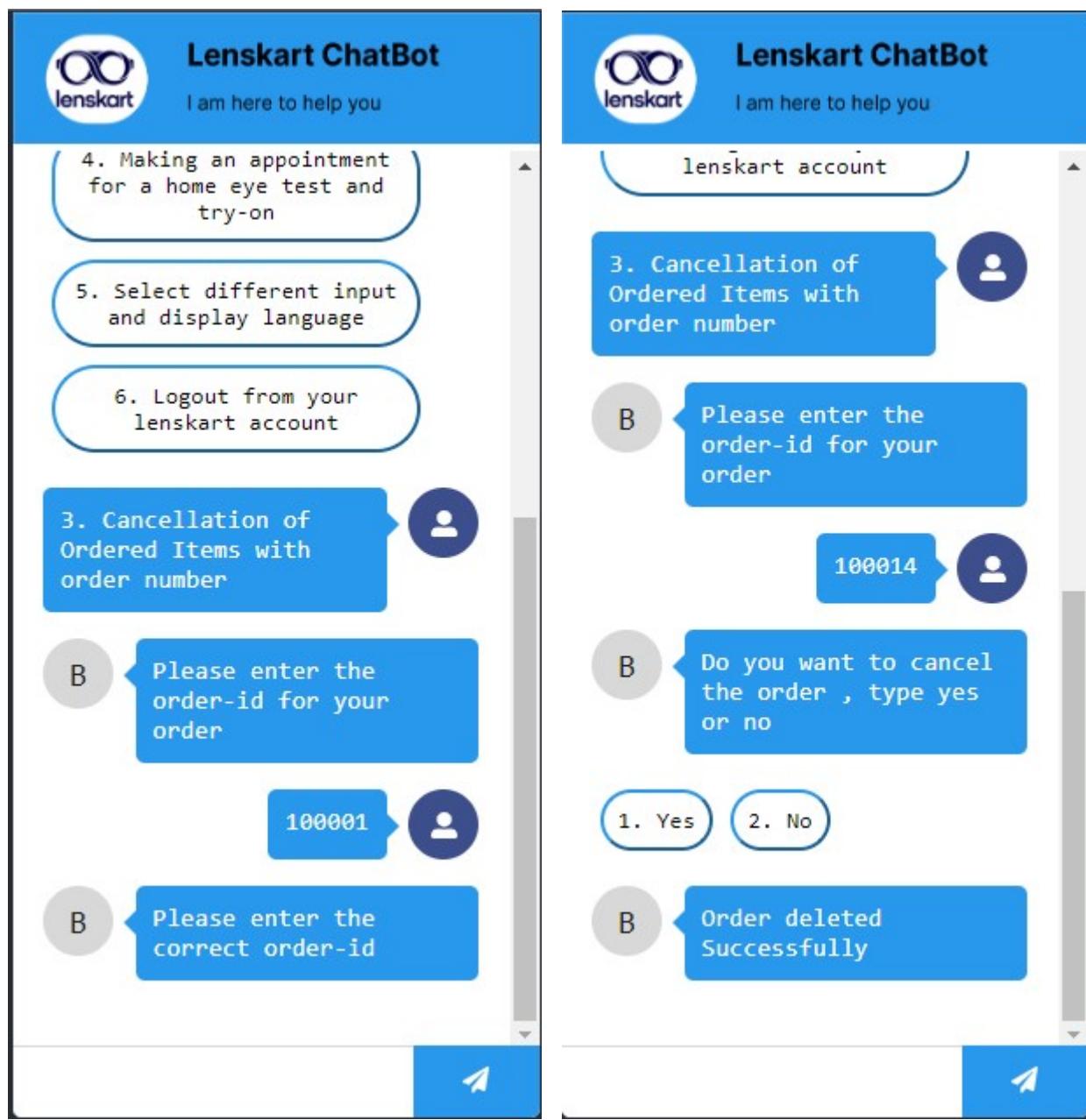


Fig 5.3.11 : Cancellation of ordered items -1 Fig 5.3.12 : Cancellation of ordered items2

the above screenshots shows us about the order cancellations and gives the appropriate results for the both of the cases where the order id exists and does not exists.

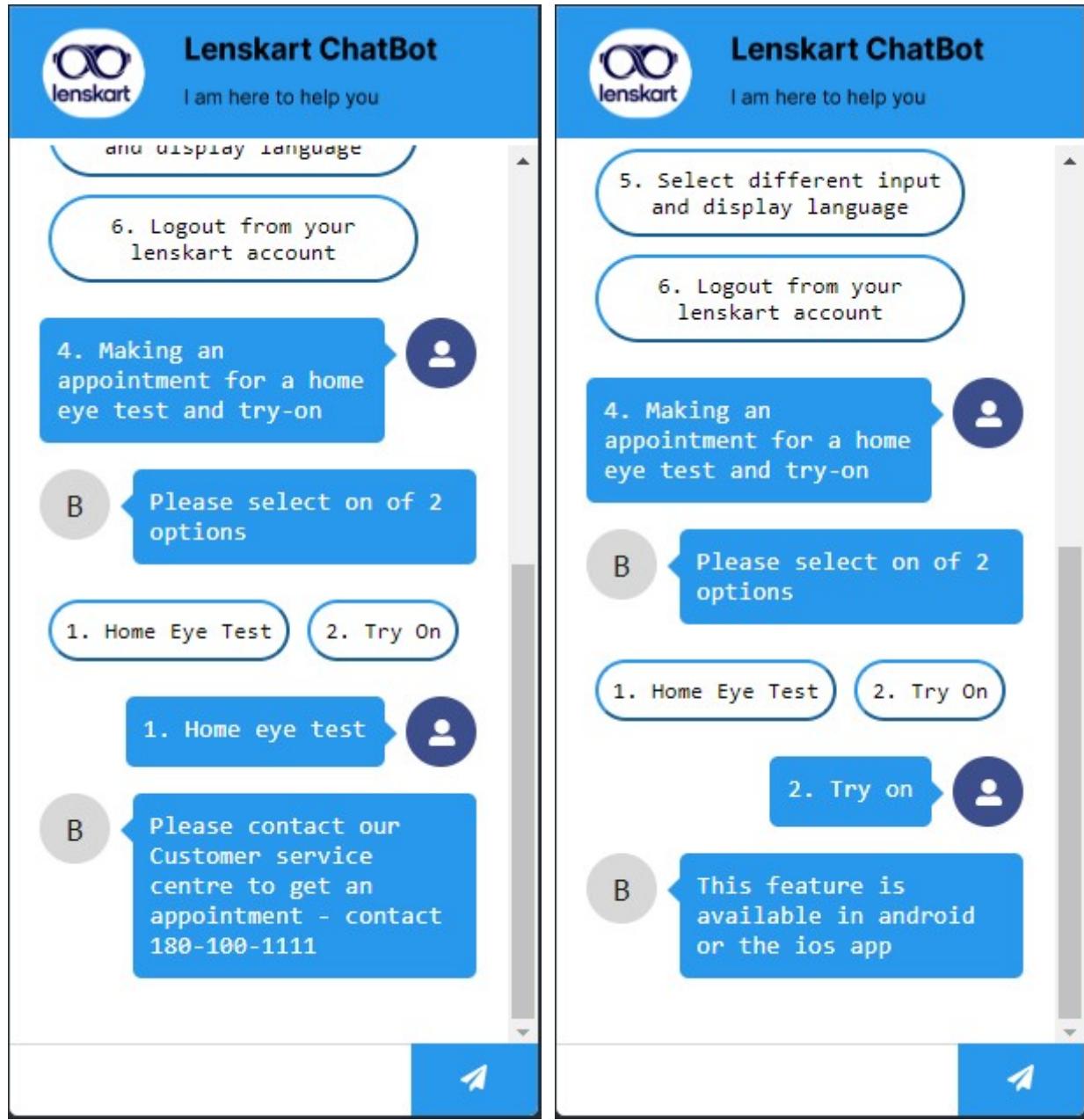


Fig 5.3.13 : Making home appointment -1

Fig 5.3.14 : Making home appointment -2

the above screenshots shows us about the making an home appointment for the eye test and it shows the two options – Home eye test and Try on
the home eye test and try on shows the respective outputs as shown in the figures.

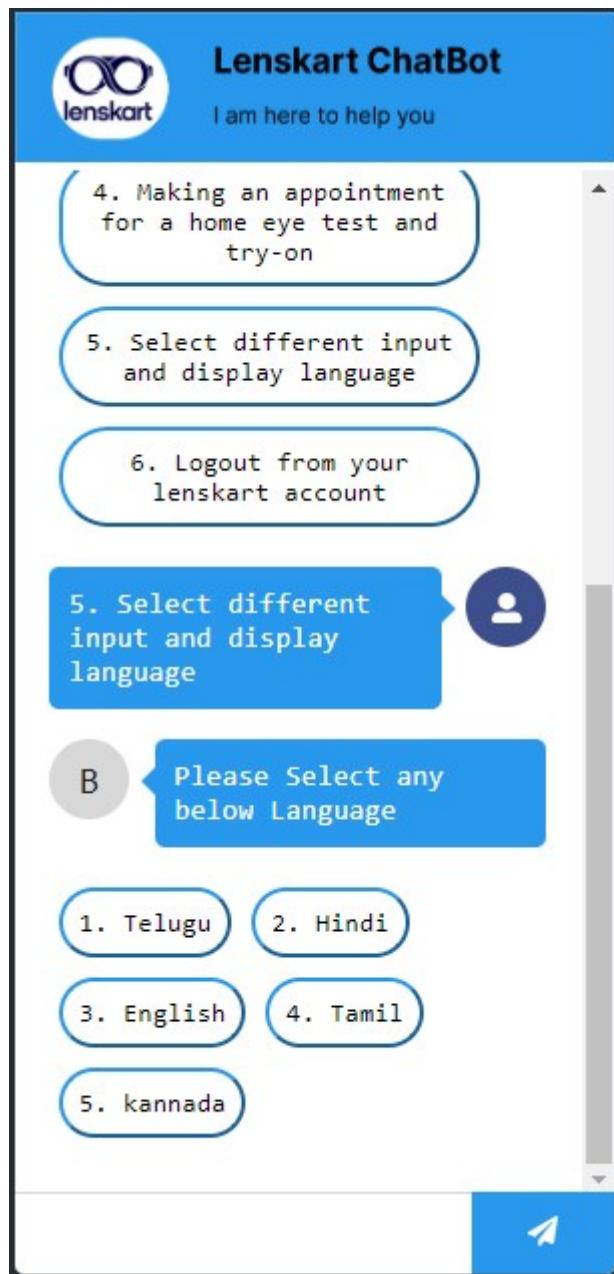


Fig 5.3.15 : language translation options-1

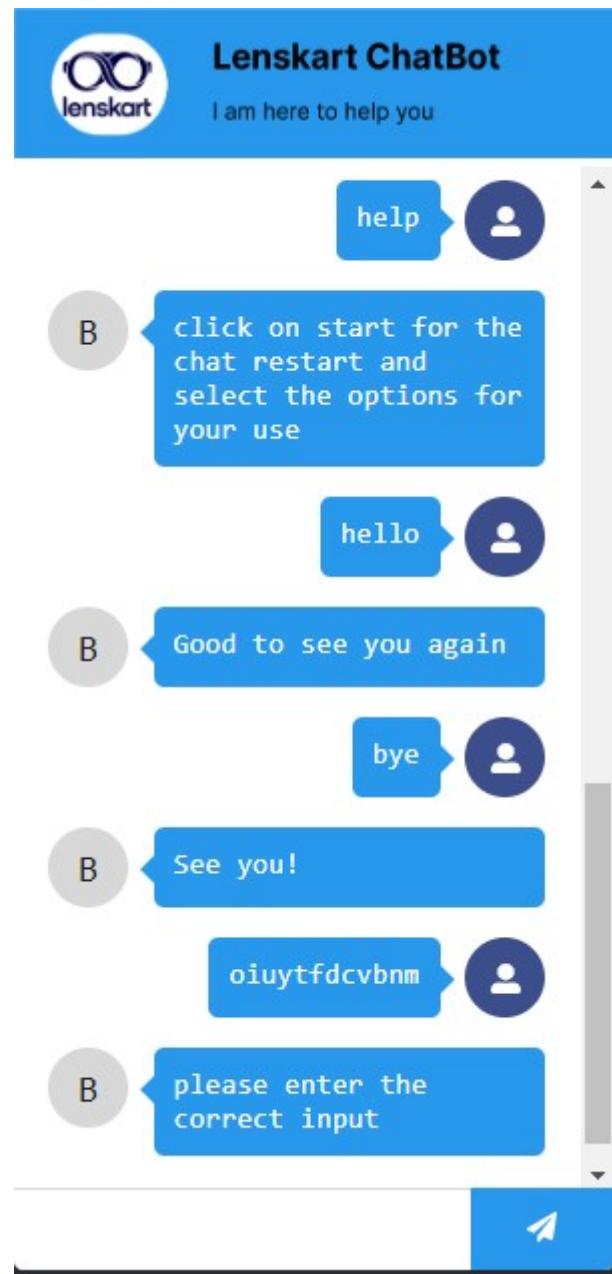


Fig 5.3.16 other chatbot messages

Fig 5.3.15 represents the main feature and the purpose of this project .i.e language translation depending on which we select , the options will be changed into that respective language . Fig 5.3.16 represents the random messages and its respective outputs given by the to the user messages . these responses are from the model hence we can get the particular output depending on the message entered.

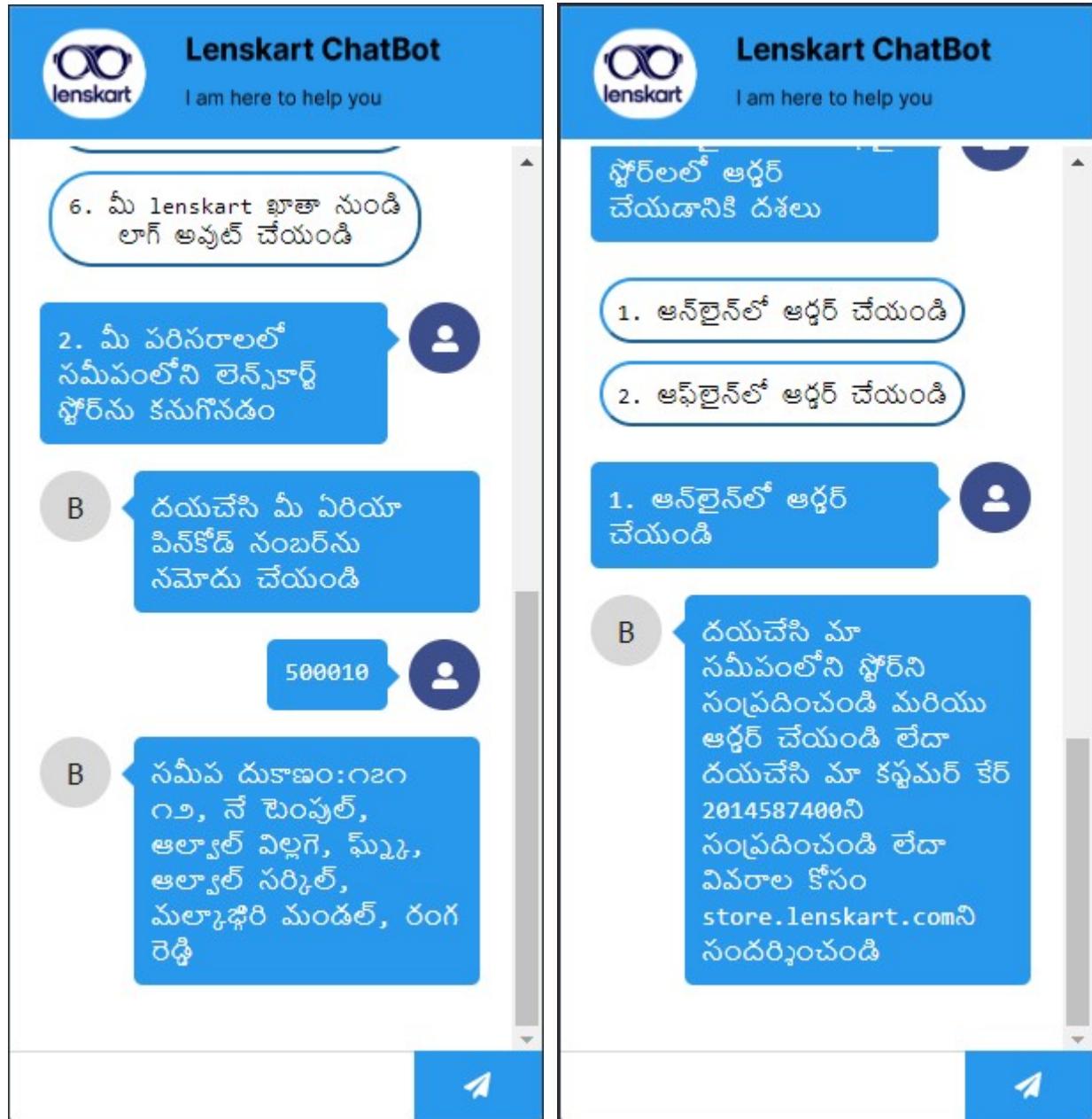


Fig 5.3.17 : telugu translated options-2

Fig 5.3.18 telugu translated text-2

As shown in Fig 5.3.16 if we select the option telugu from there we can get then respective output from there and Fig 5.3.20 is the resultant output of it.

The Figure 5.3.19 shows the “finding the nearest lenskart Store “ Option in telugu.

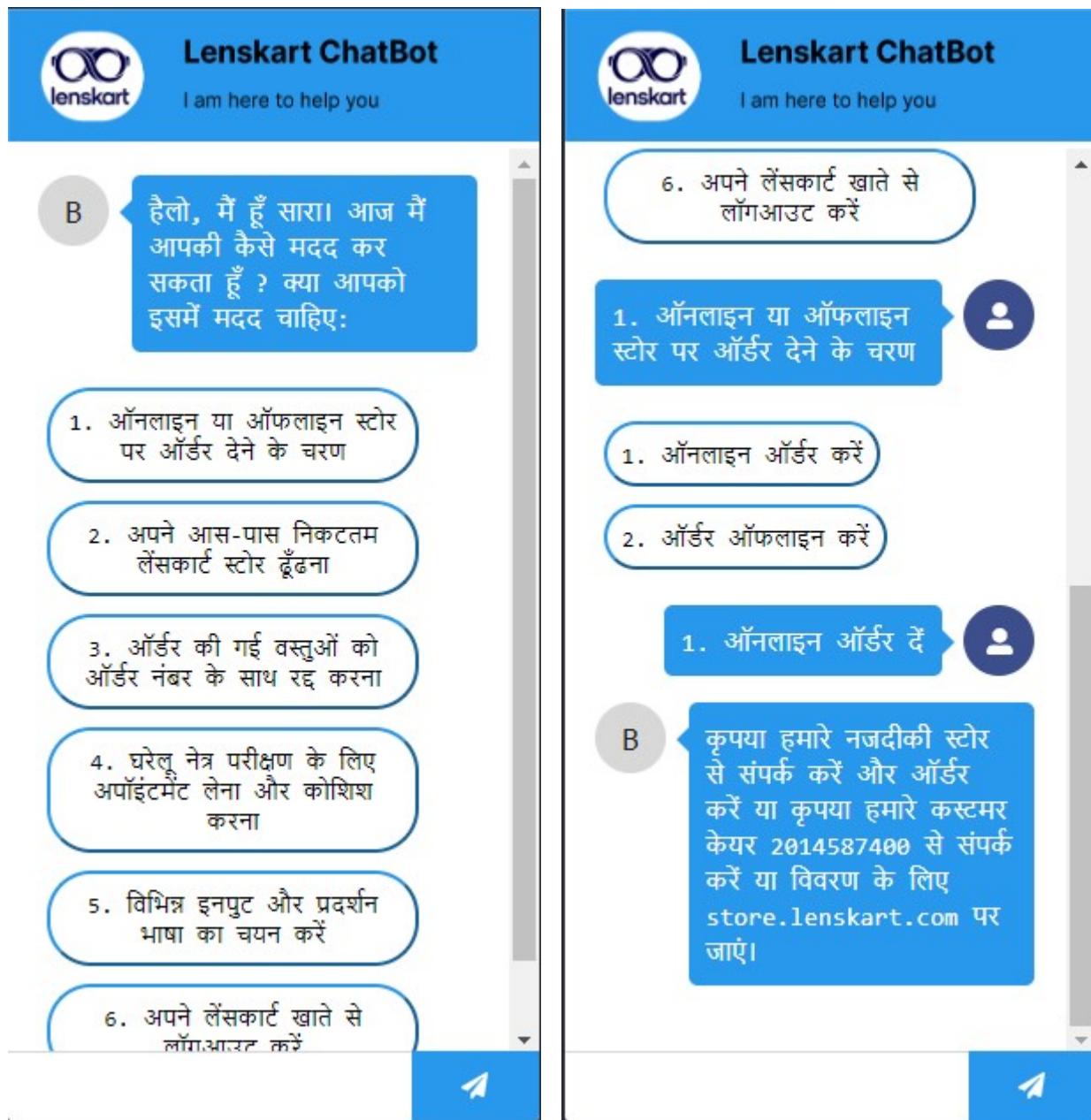


Fig 5.3.19 : hindi translated options-1

Fig 5.3.20 hindi translated text-1

As shown in Fig 5.3.16 if we select the option hindi from there we can get then respective output from there and Fig 5.3.22 is the resultant output of it.

Fig 5.3.22 represents the first options “Steps to place order in online or offline” which shows in hindi language.

Lenskart ChatBot
I am here to help you

B வணக்கம், நான் சாரா. இன்று நான் உங்களுக்கு எப்படி உதவ முடியும்? உங்களுக்கு உதவி தேவையா:

1. ஆன்லைன் அல்லது ஆஃப்லைன் கடைகளில் ஆர்டர் செய்வதற்கான படிகள்
2. உங்கள் சுற்றுப்புறத்தில் அருகிலுள்ள லென்ஸ்கார்ட் கடையைக் கண்டறிதல்
3. ஆர்டர் என் கொண்ட ஆர்டர் செய்யப்பட்ட பொருட்களை ரத்து செய்தல்
4. வீட்டிக் கண்

Lenskart ChatBot
I am here to help you

1. ஆன்லைனில் ஆர்டர் செய்யுங்கள்

2. ஆஃப்லைனில் ஆர்டர் செய்யுங்கள்

1. ஆன்லைனில் ஆர்டர் செய்யுங்கள்

B அருகிலுள்ள எங்கள் கடையைத் தொடர்புகொண்டு ஆர்டர் செய்யுங்கள் அல்லது எங்கள் வாடிக்கையாளர் சேவை 2014587400ஜத் தொடர்பு கொள்ளவும் அல்லது விவரங்களுக்கு store.lenskart.com ஜப் பார்வையிடவும்

Fig 5.3.21 : tamil translated options-1

Fig 5.3.22 tamil translated text-1

As shown in Fig 5.3.16 if we select the option tamil from there we can get then respective output from there and Fig 5.3.25 and Fig 5.3.26 is the resultant output of it.

Fig 5.3.26 represents the first option “Steps to take the order online or offline” in online order in tamil.

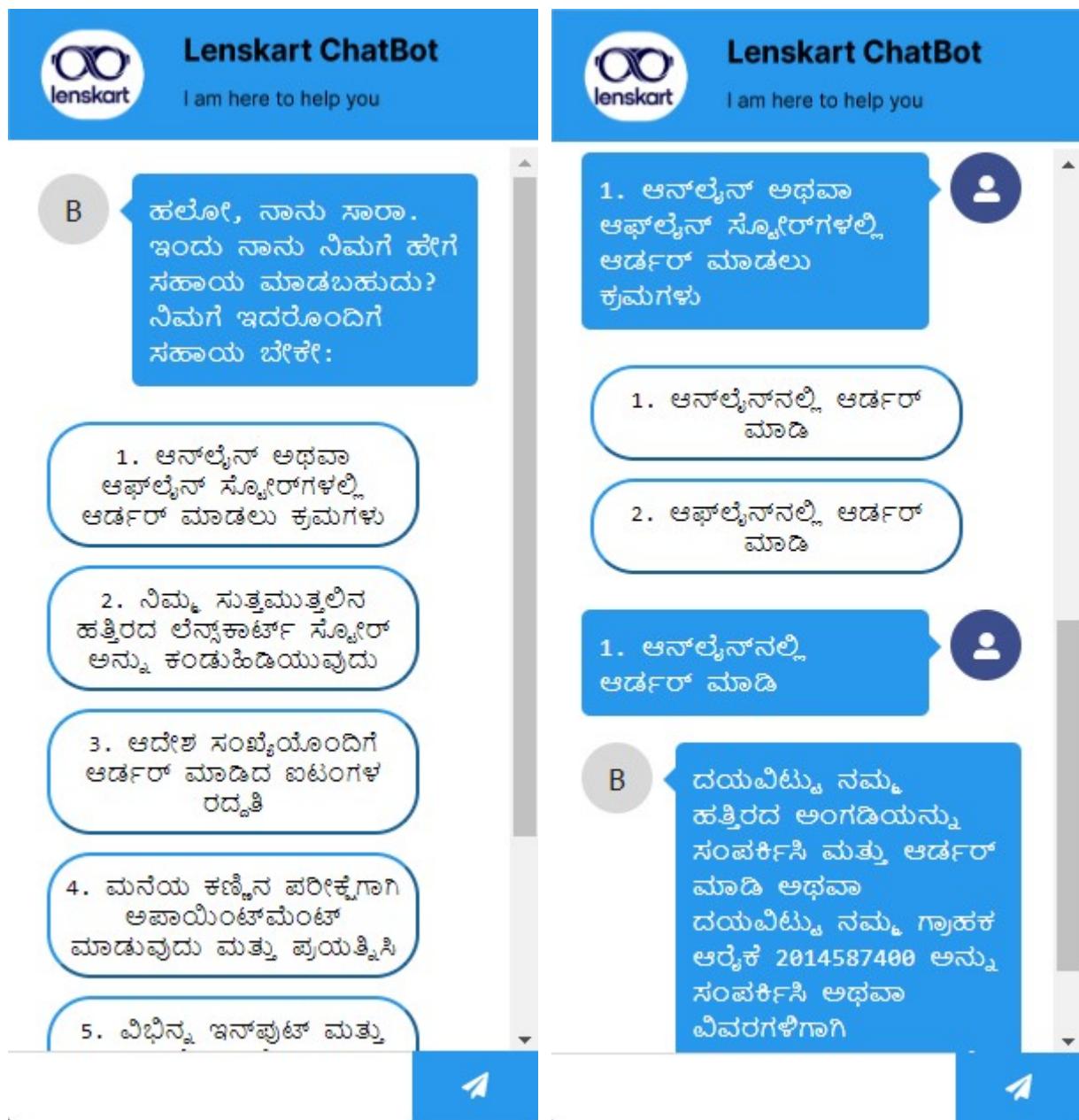


Fig 5.3.23 : kannada translated options-1

Fig 5.3.24 kannada translated text-1

As shown in Fig 5.3.16 if we select the option kannada from there we can get then respective output from there and Fig 5.3.27 and Fig 5.3.28 is the resultant output of it.

Fig 5.3.26 represents the first option “Steps to take the order online or offline” in online order in kannada.

6. TESTING

6.1 Types of Testing

There are a lot of different ways of testing an application, including testing tools such as Bugzilla, JMeter and Selenium. But the testing we performed was a type of white box testing, where we as the developers performed testing by providing a multitude of inputs, validating the outputs and checking if they are up to the mark or not. The test cases we used to test our web application are given below in the next section.

6.2 List of Test Cases

Given below in Table VI, are the test cases which include various test cases that we have used to test the various features of our web application:

TEST SCENARIO	TEST CASE ID	CATEGORY	FEATURE DESCRIPTION	PREREQUISITE	TEST DESCRIPTION	INPUT DATA	EXPECTED RESULT	ACTUAL RESULT	STATUS
Validation of Automated chatbot Authentication	TC_001	Functional	Validation of Sign in Page	1.The website link should be opened in a browser window 2.internet connectivity should present	1.Fill in user-id and password 2.Click Sign In	1.userid: user15@lenskart.in 2.password: John@123	Invalid user show come as output	Invalid user	Pass
Validation of Automated chatbot Authentication	TC_002	Functional	Validation of Sign in Page	1.The website link should be opened in a browser window 2.internet connectivity should present	1.Fill in user-id and password 2.Click Sign In	1.userid: user1@lenskart.in 2.password: John@123	Should show the wrong password	Wrong password	Pass
Validation of Automated chatbot Authentication	TC_003	Functional	Validation of Sign in Page	1.The website link should be opened in a browser window 2.internet connectivity should present	1.Fill in user-id and password 2.Click Sign In	1.userid: user1@lenskart.in 2.password: pass1	User should be redirected to the chatbot page	User is redirected to the chatbot page	Pass
Validation of Automated chatbot chatting page	TC_004	Functional	Validation of chatbot page	1.The website link should be opened in a browser window 2.internet connectivity should present 3.The user should be registered and logged in as a lenskart User.	Checking whether the option "Steps to place an order online or offline "is working or not	-	Should show the options 1. Place an order online 2. Place an order offline	Showed the respective options for the result	Pass
Validation of Automated chatbot chatting page	TC_005	Functional	Validation of chatbot page	1.The website link should be opened in a browser window 2.internet connectivity should present 3.The user should be registered and logged in as a lenskart User.	Checking whether the option "Find the nearest lenskart store "is working or not	500010	The nearest store is 171 12, nr temple, Alwal, Malkangiri, Secunderabad, 500010.	The nearest store is 171 12, nr temple, Alwal, Malkangiri, Secunderabad, 500010.	Pass
Validation of Automated chatbot chatting page	TC_006	Functional	Validation of chatbot page	1.The website link should be opened in a browser window 2.internet connectivity should present 3.The user should be registered and logged in as a lenskart User.	Checking whether the option "Cancellation of order items with order no "is working or not	500010	Please enter the correct order number	Please enter the correct number	Pass

Validation of Automated chatbot chatting page	TC_007	Functional	Validation of chatbot page	1.The website link should be opened in a browser window 2.internet connectivity should present 3.The user should be registered and logged in as a lenskart User.	Checking whether the option "Cancellation of order items with order no "is working or not	100014	The order with that order no is cancelled	The order with that order no is cancelled	Pass
Validation of Automated chatbot chatting page	TC_008	Functional	Validation of chatbot page	1.The website link should be opened in a browser window 2.internet connectivity should present 3.The user should be registered and logged in as a lenskart User.	Checking whether the option "Select a different language output "is working or not	telugu	The chatbot restarted with language telugu	The chatbot restarted with language telugu	Pass
Validation of Automated chatbot chatting page	TC_009	Functional	Validation of chatbot page	1.The website link should be opened in a browser window 2.internet connectivity should present 3.The user should be registered and logged in as a lenskart User.	Checking whether the option "Select a different language output "is working or not	hindi	The chatbot restarted with language hindi	The chatbot restarted with language hindi	Pass
Validation of Automated chatbot chatting page	TC_010	Functional	Validation of chatbot page	1.The website link should be opened in a browser window 2.internet connectivity should present 3.The user should be registered and logged in as a lenskart User.	Checking whether the option "Select a different language output "is working or not	tamil	The chatbot restarted with language tamil	The chatbot restarted with language tamil	Pass
Validation of Automated chatbot chatting page	TC_011	Functional	Validation of chatbot page	1.The website link should be opened in a browser window 2.internet connectivity should present 3.The user should be registered and logged in as a lenskart User.	Checking whether the option "Select a different language output "is working or not	kannada	The chatbot restarted with language kannada	The chatbot restarted with language kannada	Pass
Validation of Automated chatbot chatting page	TC_012	Functional	Validation of chatbot page	1.The website link should be opened in a browser window 2.internet connectivity should present 3.The user should be registered and logged in as a lenskart User.	Checking the response of the chatbot for the particular messages	hello	Hi, nice to meet you	Hi, nice to meet you	Pass
Validation of Automated chatbot chatting page	TC_013	Functional	Validation of chatbot page	1.The website link should be opened in a browser window 2.internet connectivity should present 3.The user should be registered and logged in as a lenskart User.	Checking the response of the chatbot for the particular messages	help	Please enter start for to start the chatbot and select different options to get your requirement satisfied.	Please enter start for to start the chatbot and select different options to get your requirement satisfied	Pass

7. CONCLUSION AND FUTURE SCOPE

Conclusion

Our system was designed in order to increase the interactions and circulation of more lenskart products to different regions in the country. We expect to see a lot of customers and use our system. We would like to help out a wide range of customers and specks manufacturers with our system. With the implementation of this system, we expect to see increase in more and more regional and rural customers who can interact with their regional language. This would help out villagers and rural people by providing a trust-worthy medium to verify the order and the specks they are purchasing. Also, the manufacturers can benefit from our system as only the verified manufacturers would be able to sell their products at lenskart.

Future Scope

It is difficult to design a system that satisfies all of the user's requirements. Users' requirements alter as the system is under use. The following are some of the prospective upgrades to this system:

- We can add a feature of speech recognition to detect the speech and add the message to the chatbot which helps in easier communication.
- We can add a feature of face recognition in chatbot which helps in detecting the face and add an AI implemented feature of adding specks to the face in chatbot.

8. BIBLIOGRAPHY

1. <https://fastapi.tiangolo.com/tutorial/>
2. <https://reactjs.org/tutorial/tutorial.html>
3. <https://www.tensorflow.org/guide/keras/rnn>
4. International Journal of Recent Technology and Engineering (IJRTE) – chat bot and its practical applications in the materialistic world.
5. <https://huggingface.co/docs/transformers/index>
6. <https://devcenter.heroku.com/categories/reference>
7. https://docs.netlify.com/?_ga=2.50570025.1403717473.1655133651-380874290.1655133651
8. <https://firebase.google.com/docs/release>
9. <https://lenskart-chatbot.netlify.app/> (deployed app)
10. <https://hf.space/embed/nitishkumargundapu793/chat-bot-response/+> (deployed model)
11. <https://chatbot-fast-server.herokuapp.com> (deployed backend server)
12. [https://console.firebaseio.google.com/project/project7f27d.firebaseio.firebaseio.com/~2Fmessages~2F0sSOiRh6hMWUlhCVZo7](https://console.firebaseio.google.com/project/project7f27d.firebaseio/data/~2Fmessages~2F0sSOiRh6hMWUlhCVZo7) (firebase datastore)

plag1.pdf

ORIGINALITY REPORT



PRIMARY SOURCES

dl.ucsc.cmb.ac.lk
1 Internet Source **1%**

Submitted to University of Warwick
2 Student Paper **1%**

Submitted to University of Surrey
3 Student Paper **1%**

documents.ucsc.lk
4 Internet Source **1%**

Submitted to University of Wales central institutions
5 Student Paper **<1%**

ukdiss.com
6 Internet Source **<1%**

7daixie.com

7 Internet Source

<1%

eprints.utm.my

8 Internet Source

<1%

www.ijgser.com

9 Internet Source

<1%

Exclude quotes

On

Exclude matches

< 5 words

Exclude bibliography

On

Chatbot and its Practical Applications in the Materialistic World

Gundapu Nitish Kumar, Devavarapu Sreenivasarao, Shaik Khasim Saheb

Abstract: These days Chat has become the new way of conversation and changed the way of life and the view that the world used to see before and due to Industrial revolution 4.0, the gradual increase in machine learning and artificial intelligence fields has gone to higher and many companies are reaching customers to get their products with more ease . This is where chatbots are used. It all started with one question! can machines think? The concept of chatbots came into existence to check whether the machines could fool users and make them think that they are actually talking to humans and not robots. On the Other hand, with the Successes Rate of Chat bots, Different companies Started using machines for having conversations with their customers about everything which made their work simpler and reduced the need of man power. There are many different types of building a chatbot but this paper will mainly concentrate on building a Chatbot using TensorFlow API in python

Keywords : Chatbot, Industrial Revolution, Machine Learning, Artificial Intelligence, TensorFlow, API, Python.

I. INTRODUCTION

A Chat Bot is a computer program which helps develop conversations with humans, deploying Natural Language Processing in terms of textual, audible and interactive messages. In today's era, many Chat Bots are seen to be revolutionizing the way machines talk to humans, such as Siri, Alexa and Google's Voice Assistant. In simpler terms, a Chat Bot is just a piece of code which takes an input from the user and tries to match the input with similar statements existing in a database comprising of a multitude of such statements, and delivers an output which relies on the model containing a pattern of such statements, usually referred to as the 'brain' of the Chat Bot. The present times have seen the evolution of various kinds of Chat Bots, depending on their means of usage, and so, this paper explains various facets involved in the building of a Chat Bot which uses a data set to train itself and responds accordingly. This paper hence primarily focuses on the development of a user-friendly Chat Bot practically using Google's Deep Learning Framework, TensorFlow in

Revised Manuscript Received on November 15, 2019

* Correspondence Author

Gundapu Nitish Kumar is currently pursuing B.Tech Degree program in Computer Science & Engineering in Sreenidhi Institute of Science and Technology, Affiliated to Jawaharlal Nehru Technical University Hyderabad, Telangana, India, PH-7032580182.

Devavarapu Sreenivasarao is currently working an Assistant Professor in Computer Science & Engineering Department in Sreenidhi Institute of Science and Technology and his area research includes Medical Image Processing, Machine Learning. PH-9866014581.

Shaik Khasim Saheb is currently working as Assistant Professor in Computer Science & Engineering Department in Sreenidhi Institute of Science and Technology, and his area research includes Medical Image Processing, Machine Learning. PH-9642097865.

Python.

A. Introduction to Chatbot

Conversation has turned an essential aspect people's lives, and so, many researchers are working today in order to construct a perfect interactive conversation between humans and machines. However, the implementation of conversations in an AI Dependent Systems is a tedious task, owing to the complexity of its design. In the initial stages of development of Chat Bots, patterns were written in a language known as AIML (Artificial Intelligence Markup Language) [1], in which the patterns are written into a file and adopted by a code to construct a conversation. This is a pattern oriented method, since the pattern taken as input is directly searched for in the AIML File.

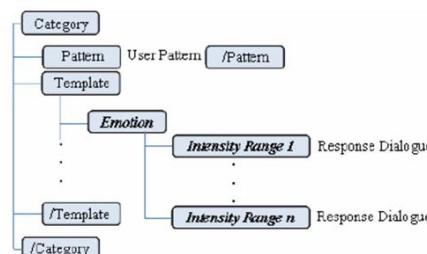


Fig. 1. Composition of an AIML Pattern

An AIML file is a collection of category tags. Each category tag is considered a query and there are numerous category tags depending on the number of queries. These category tags comprise of two main portions – the pattern and the template. The input taken is matched with the pattern, and the template is returned as the answer to the query as shown in Figure 1. The augmentation of Chat Bots by AIML patterns made their modeling simpler. However, they lacked efficiency because of the method of implementation. In most cases the AIML powered Chat Bots were able to answer only those questions to which relevant patterns were mentioned in the AIML File. For instance, if a pattern for the query, "What is your name?" is written in the AIML File, the Chat Bot would only be able to answer that question, but not some other question with the same meaning, such as "Tell me your name."

Here is where the concept of bigram calculations [2] came into existence. For the given input, probability calculations of other sentences are made in order to find out similar sentences in the database, i.e. the response returned is of that sentence which has the highest probability. The probability of all the sentences is computed using the below equation.

$$p(\varphi) = \prod_{i=0}^n p(\varphi_i | \varphi_0, \dots, \varphi_{i-1})$$



However, in the present times, Chat Bots are extensively being used in industries and companies to enhance communication with their customers and improve the accessibility of their products. A multitude of websites on the internet are shifting to Chat Bots from the conventional, archaic techniques of customer care, just like industries which rely on machines to get the task done swiftly and efficiently. Nevertheless, the conventional techniques of creating a Chat Bot fail to work in such instances, owing to the complexity of writing patterns for millions of queries, which is certainly a herculean task. Hence, the only solution to this issue is to build a model which takes an input and returns an output by training itself on datasets and learns to construct replies by itself. This means, we require a deep learning model which learns by itself instead of being explicitly programmed to reply. The following sections elaborate the construction of a Chat Bot, relying on deep learning frameworks. Chat Bots are mainly of two types

1. Generative based (from a Dataset).
2. Retrieval Based (from a Database).

A Retrieval Chatbot need to have a large amount of data in order to train itself and respond to the user's input, hence it requires database in order to retrieve data. but a Generative Based Chat Bot trains itself on a dataset to give the response to the user. This paper will show how to create a chat bot of generative type using seq2seq model which works on recurrent neural networks. but both types of chat bots require substantial amount of resources to get trained and executed. Even after the training process is done, the outputs are not perfect. Hence the inputs are normalized to get the accurate output.

B. Introduction to TensorFlow

TensorFlow is a free open source library designed in order to manage the work flow in different tasks. It was created by Google's Brain team for its internal usage. It is made open source in the year 2015. TensorFlow is also used in differentiable programming which is widely used in AI. TensorFlow was created initially to control the data flow due to its extensibility over large data. TensorFlow uses a computational graph for executing its tasks. For any TensorFlow model, its computational graph is made in order to execute the model. This paper will show the execution of seq2seq model to build a chatbot which is available in TensorFlow library.

II. METHODOLOGY

The following steps [3] are adopted in order to build a Generative Chat Bot.

- Step 1: Choosing a Data set.
- Step 2: Data Preparation and pre-processing.
- Step 3: Graph Building.
- Step 4: Neural Network.
- Step 5: Model Selection.
- Step 6: Optimizers.
- Step 7: Training the Model.
- Step 8: Executing.
- Step 9: Testing and Results.

A. Choosing a Data set

This paper demonstrates the usage of the Twitter dataset for the purpose of completing this task. This data set consists of lots of useful Twitter Conversations which can be useful for the analysis and the training of the Chatbot. This data set is not especially designed for training chat bots, but it can be used to make simple conversations between humans and machines.

B. Data Preparation and Pre-processing

Data preparation in the context of building a Chat Bot refers to the conversion of data which is the form of text present in datasets to utterance pairs – question and answer pairs used to train the Chat Bot. As mentioned above, the demonstration shown in this paper will use Python. For the Data preprocessing, analyzing and interpretation, nltk and re modules of Python are made use of. The input data must initially be converted either into lowercase or uppercase, in order to be preprocessed. Firstly, unwanted special characters, white spaces are removed from the input. If the input is still found to contain too many numbers and additional, insignificant words, such words too are removed depending upon the input to get the generalized input form. These generalizations are derived using regular expressions and most optimizations are done using the module nltk.

C. Graph Building

For training a model, it must be defined. In the process of graph building, the hidden layers are defined to give the accurate output. The demonstration shown in this paper uses a predefined model named seq2seq for natural machine translation. Using the Dataset API, the data preparation pipeline can be defined and batched. All that needs to be done is wrapping the data using the Dataset API. Now, some data operations should be done on the dataset like Tokenization, Word to Index, Shuffling, and Batching, which convert the entire dataset into utterance-pairs, thereby simplifying processing.

D. Neural Network

The Neural Network used in this paper is an RNN. A Recurrent Neural Network (RNN) [4] is a type of a neural network that can take a sequence $D = (d_1, \dots, d_n)$ as an input and produces a sequence of encoded states $L = (L_1, \dots, L_n)$ by using the process of recurrence . It is also called as unrolling as shown in Figure 2 at every stage the front part of the network takes x_i and h_{i-1} as input and returns an encoded state h_i as output. At every stage i , the hidden layer h_i is updated by below Equation.

$$l_i = f(Al_{i-1} + Bd_i)$$

Where A and B are the vectors which contains the weights of the neural network, where f is a nonlinear activation function which represents a hyperbolic tangent function. LSTM's [6] were designed to improve the performance and to resolve the problems faced by the Vanilla version of RNN. The Vanilla version of RNN is not used because of its problem with Vanishing gradient [5]. As the steps involved in the unrolling process increase it becomes harder for a simple recurrence neural network to learn and to remember information.



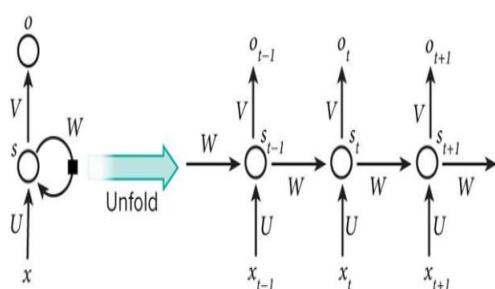


Fig. 2. Unfolding of an RNN over 3 time-steps

In the above mentioned figure, x is the input sequence, o is the output sequence. LSTMs are enabled with gates in order to regulate the dataflow present in the neural network. These gates consist of information related to the operations done on the input and the previous encoded output such as the matrix multiplications, non-linear functions etc. GRUs are mainly used for language modeling, because of its ability of preserving information for different and more than one sentence when the network is being unrolled for every word in the provided input. An important property of an RNN is that the components relates to the activation function remains constant when the network is being unrolled. This makes the model eligible for its usage in Language Modeling. When this model is trained to learn the probability Distribution of vocabulary v it will be well suited for the natural language processing. The probability distribution function that can be used to predict the next word in the sentence is given by below equation,

$$\rho(x_{ij}|x_{i-1} \dots x_i) = \frac{\exp(v_j h_i)}{\sum_{j=1}^k \exp(v_j h_i)}$$

Hence for each value of j from $1, \dots, k$, v_j represents the rows of the weighted matrix and the sequence (x_{i-1}, \dots, x_1) represents the input given to the model and h_i represents the encoded state of the neural network for every value i . so the probability function above results the accurate words for the corresponding input. The training part of the neural network is done using truncated back propagation [8] algorithm. Essentially the error is back propagated for each step. The error can be calculated by using a loss function, which calculates how the calculated values would be when compared with the true labels.

$$\text{Loss}(y_j, \hat{y}_j) = -y_j \log(\hat{y}_j)$$

After calculating the loss, the training process is improved by adding optimization techniques such that the neural network gives the output having maximum accuracy. The final outcome from the decoder will be a vector consisting of 1's and 0's, which will be further processed to get the accurate output which resembles required output for the corresponding input. The Seq2seq is a very useful model and these days many of the natural language processing systems uses this model to get the accurate output.

E. Model Selection

The model that will be used is seq2seq. Seq2seq [10] was the first model introduced for the implementation of machine translation using neural networks. Before the model was introduced, the machine translation was done in a different way. Each sentence that was given as input was splitted into

words and converted to its outcome expected language giving no regards to the respective language's grammatical pattern. Seq2seq has radically changed this process by using concepts of deep learning. This model takes the input splitted words and input neighborhood for the training process while translating to other language. Now a days, this model is used for a many different useful applications like conversational models, image captioning, text summarization etc.

Seq2seq Model Working

The Seq2seq model takes a sentence (The flow of Characters) as an input and generates an output sentence. The process is done by using RNNs (recurrent neural networks). The RNN (Vanilla version) is not used widely because of its problem with vanishing gradient. Instead, more advanced version GRU (or LSTM) is used. The Model is developed with an idea of talking two inputs to get the output where one of them is from the sentence provided by the user and other from the previous given output. The seq2seq model consists of two major RNNs - The Encoder and The Decoder, the working is shown in the figure 3.

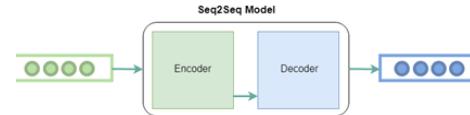


Fig. 3. The Encoder and Decoder Model

Encoder

It is an RNN which uses deep neural networks to convert the given input sequence of words into the sequence of encoded vectors where each hidden state represents the depended factors of the given word.

Decoder

It is an RNN similar to the encoder where it takes the encoded vectors generated by encoder as an input, its own encoded states and present word in order to produce the next encoded vector which is used to predict the next word for the corresponding input. These are the Additional Components of seq2seq model.

Attention

If a hidden state is given to the decoder as a single vector the computation will be easy but the computation becomes difficult if the input is given as the sequence of encoded vectors. Here, the attention mechanism is adopted in order to select the input sequence relatively and selectively.

Beam Search

Always a word which is considered as the output will have the highest probability compared to other output words. but this way of taking output does not always gives best results because of greedy algorithms. To rectify this, Beam search mechanism is applied which suggests the possible translated output at each and every step. This process is done by making a tree of k-results.

Bucketing

Bucketing is done in order to reduce the memory usage, for example, if the maximum length for the output is assigned as 1000 and the output sentence is just 40 words it leads to the enormous memory usage. To reduce this bucketing is used. We make ordered pairs like (4,6), (7,15) etc which 7 is the input length and 15 will be maximum length of the output that will be generated.

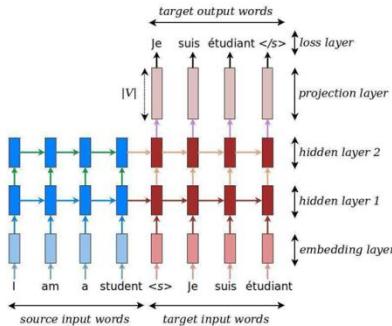


Fig. 4.The Working of the Seq2seq model using RNN

F. Optimizers

Using the loss obtained, gradients can be computed, which train the model using an optimizer. We define an operation known as optimizer which calculates gradients and uses them to train the model. We also add to more components to the optimizer like cosine decay and gradient clipping to operate the training process of the model. Cosine decay is used to analyze the growth rate mechanism and gradient clipping is used to ignore exploding gradients created.

G. Training the Model

Training the model involves initializing all variables and starting to train the model. We have to save the state of the model for every epoch and calculate average loss for each and every epoch to obtain accurate results. This Process may take a lot of time, depending on the hardware configuration of the machine used for the training process.

H. Executing

The Execution part of the Chatbot involves creating a seq2seq model for NLP. But the python's Tensor Flow module consists the model predefined. Hence it made the task easier and more efficient compared to other methods. The number of epochs is 50 and the accuracy is nearly 95 % which turns out to be pretty accurate compared to other models.

I. Testing and Results

Training process of the model took more time than expected but the results were accurate not exact. To test the resulting model different inputs were given, and as expected some of the answers were good and some of them were not that good. This model was made in python and the results were demonstrated using nltk, TensorFlow, NumPy libraries.

III. DISCUSSIONS AND RESULTS

After taking the input from the user, the input is generalized in to such a way that the outcome after the generalizations would be a string with no special characters and exaggerated words. First the input is taken as a string and the white spaces from first and last are removed and converted into lowercase to make the data processing easier. Given table 1 is some of the examples of the processed strings

Table-I: Eliminating white spaces and converting the strings into lower case.

Input	Output
Hi HOW aRe YOu	hi how are you
HeLlOwOrlD	hello world
I'M FIne and Good	i'm fine and good
I WILL eat Rice	i will eat rice
ILikeCrICKet	i like cricket

After the input is converted into lower case, all the Special Characters such as !, @, = etc are replaced with the white spaces in String Which Makes the input to be more readable and easier to process than compared with the sentence with the special Characters. The respective output for the Corresponding inputs is shown in the Table 2.

Table-II: Taking out the Special Characters and replacing them with the white spaces

Input	Output
hi@how(are you	hi how are you
Hello34World	hello world
i'm fine-and good	im fine and good
i=will eat*rice	i will eat rice
i!like)cricket	i like cricket

Then after the punctuations and the grammatical errors in the Sentences are removed using the nltk.stem() function which helps correct sentences and convert them into grammatically accurate ones.

Table-III: After the Correcting the input grammatically

Input	Output
hii how are u	hi how are you
Hello. , world	hello world
i'mm fine and goood	im fine and good
i will eatt rice	i will eat rice
iilkecricket	i like cricket

After the input normalization is done, the input is split into a group of words and these words are passed into the seq2seq model. The seq2seq2 model is made of two recurrent neural networks named encoder and decoder. The encoder takes the input and converts the input list of words into an encoded vector using the concept of recurrence as shown in the figure 5.

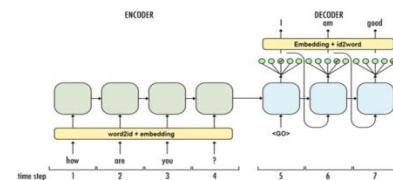


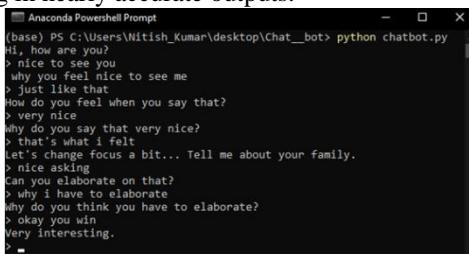
Fig. 5. Sequence of Networks in RNN

After this, the decoder decodes the encoded vector taking two inputs one from the vector and another from the previous input, which forms a grammatically accurate output from the model. In the entire process of building the generative Chat Bot, it is observed that the most time-consuming part is the training part, as the time taken for training depends not just on the model, but also on the architecture of the machine. When the Chat Bot is ready to be asked questions, outputs are delivered by it as shown in the table 4.

Table-IV: Outputs after the model is rendered

Input	Output
hi , how are you	I am fine
Do you know me	How can I?
will you marry me	Can you elaborate on that?
What is your Gender	i prefer not to say
bye	have a great time

It is observed that reasonably precise outputs are given for the inputs delivered to the Chat Bot. The following figures depict the model demonstrated in this paper. The training part of this model took 50 epochs and in due course of advancement of training, the outputs obtained to the questions asked improve, resulting in nearly accurate outputs.



```
(base) PS C:\Users\Nitish_Kumar\Desktop\Chat_Bot> python chatbot.py
Hi, how are you?
> nice to see you
why you feel nice to see me
> just like that
How do you feel when you say that?
> very nice
why do you say that very nice?
> that's what i felt
Let's change focus a bit... Tell me about your family.
> nice asking
Can you elaborate on that?
> why i have to elaborate
why do you think you have to elaborate?
> okay you win
Very interesting.
>
```

Fig. 6. The practical implementation of a Generative chat bot using TensorFlow

When the training process was undergoing for each iteration (epoch), the accuracy was increasing and the loss was decreasing. At First, the results were pretty poor but after some epochs the outputs produced were relevant to the corresponding input given. There are many algorithms and models that can be tried on to create conversational software but, the seq2seq model had given very good results compared to other models.



```
(base) PS C:\Users\Nitish_Kumar\Desktop\Chat_Bot> python chatbot.py
Hi, how are you?
> do you know me
why you feel me know you
> i know that you know me
How does that make you feel?
> nicer
Please tell me more.
> i won't tell you
Can you elaborate on that?
> i won't
Very interesting.
> nice very interesting
How does that make you feel?
> worse
Please tell me more.
> ?
Please tell me more.
> no
Very interesting.
>
```

Fig.7. The practical implementation of a Generative chatbot using TensorFlow

IV. CONCLUSION

As demonstrated in this paper, developing Chat Bots using frameworks of deep learning make them powerful, efficient, well-structured and accurate. Though this paper describes the development of a Generative Chat Bot using RNNs, many other techniques can be used to do the same. A point worth noting is that frameworks of Deep Learning have revolutionized the ways in which Chat Bots are built and work, and hence, the world would look up to the making of Chat Bots working on more efficient algorithms, to enhance communication between a human and a machine.

REFERENCES

1. Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*,3(Feb):11371155.
2. Bottou, 2010, Large-scale machine learning with stochastic gradient descent. In proceedings of COMPSTAT2010, pages 177186. Springer.
3. Cho et al., 2014, Cho, K., VanMerriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.
4. G.Neubig," NLP Programming Tutorial 2 - Bi- gram Language Models", Presentation Moduleof Nara Institute of Science andTechnology.
5. Hochreiter, S. (1998). The van-ishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge- Based Systems*,6(02):107116.
6. Hochreiter,S. AndSchmidhuber, J.(1997). Long short termmemory. Neural computation, 9(8):17351780.
7. <Https://omarito.me/building-a-seq2seq-conversational-chat-bot-using-tensorflow/>, 2007
8. NanoDano,<https://www.devdungeon.com/content/ai-chat-bot-python-a-iml>
9. Rumelhart, D. E.,Hinton, G. E., Williams, R. J., et al. (1988). Learning representations by back-propagating errors. Cognitive modelling.
10. Werbos, P. J. (1990).Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*,78(10):15501560.

AUTHORS PROFILE



Mr. Gundapu Nitish Kumar is currently pursuing B.Tech Degree program in Computer Science & Engineering in Sreenidhi Institute of Science and Technology, Affiliated to Jawaharlal Nehru Technical University Hyderabad, Telangana, India. His main research work focuses on Machine Learning and Neural Networks



Mr. Devevarapu Sreenivasarao, currently working as an Assistant Professor in the department of CSE in Sreenidhi Institute of Science and Technology since 2014. He did Master of Technology from JNT University Hyderabad, India in year 2012. He is a research scholar in Annamalai University which was located in Chidambaram, Tamilnadu, India He has published more than 15 research papers in various peer reputed international journals..His main research work focuses on Medical Image Processing, Machine Learning.



Mr. Shaik Khasim Saheb currently working as an assistant Professor in the department of CSE in Sreenidhi Institute of Science and Technology since 2014. He did masters from VIT University, Tamilnadu, India. 2014. He is a research scholar in Annamalai University which was located in Chidambaram, Tamilnadu, India. He has published more than 10 research papers in various peer reputed international journals. His main research work focuses on Medical Image Processing, Machine Learning.

Sreenidhi Institute of Science and Technology
Department of Computer Science and Engineering
Project - II

Batch No: A2		Title
Roll No	Name	
18311A0519	GUNDAPU NITISH KUMAR	AUTOMATED CHATBOT FOR LANGUAGE TRANSLATION AND RECOGNITION

ABSTRACT

These days Chat has become the new way of conversation and changed the way of life and the view that the world used to see before and due to Industrial revolution 4.0, the gradual increase in machine learning and artificial intelligence fields has gone to higher and many companies are reaching customers to get their products with more ease.

This is where chatbots are used. It all started with one question! can machines think? The concept of chatbots came into existence to check whether the machines could fool users and make them think that they are actually talking to humans and not robots. On the Other hand, with the Successes Rate of Chat bots, Different companies Started using machines for having conversations with their customers about everything which made their work simpler and reduced the need of man power.

There are many different types of chatbots which can be used for different purposes but in this project, we will mainly focus on the chatbot used for the language detection and language translation and we will try to automate that chatbot and make it into a full stack application.

Student 1

Internal Guide

HOD

Dr. Halavath Balaji

Batch No: A2		Title
Roll No	Name	
18311A0519	GUNDAPU NITISH KUMAR	AUTOMATED CHATBOT FOR LANGUAGE TRANSLATION AND RECOGNITION

Table 1: Project/Internship correlation with appropriate POs/PSOs (Please specify level of Correlation, H/M/L against POs/PSOs)

H	High	M	Moderate	L	Low
---	------	---	----------	---	-----

SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING Projects Correlation with POs/PSOs														
PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
H	H	M	H	H	M	L	H	H	H	M	M	H	M	M

Student 1

Internal Guide

HOD

Dr. Halavath Balaji

Batch No: A2		Title
Roll No	Name	
18311A0519	GUNDAPU NITISH KUMAR	AUTOMATED CHATBOT FOR LANGUAGE TRANSLATION AND RECOGNITION

Table 2: Nature of the Project/Internship work (Please tick √ Appropriate for your project)

Batch No.	Title	Nature of Project			
		Product	Application	Research	Others (please specify)
A2	AUTOMATED CHATBOT FOR LANGUAGE TRANSLATION AND RECOGNITION		√		

Student 1

Internal Guide

HOD

Dr. Halavath Balaji

Table 3: Domain of the Project/ Internship work (Please tick √ Appropriate for your project)

Batch No.	Title	Domain of the Project				
		ARTIFICIAL INTELLIGENCE, MACHINE LEARNING AND DEEP LEARNING	COMPUTER NETWORKS, INFORMATION SECURITY, CYBER SECURITY	DATA WAREHOUSING, DATA MINING, BIG DATA ANALYTICS	CLOUD COMPUTING, INTERNET OF THINGS	SOFTWARE ENGINEERING, IMAGE PROCESSING
A2	AUTOMATED CHATBOT FOR LANGUAGE TRANSLATION AND RECOGNITION					√

Student 1

Internal Guide

HOD

Dr. Halavath Balaji