

A project report on

CUSTOMER SERVICE PORTAL AUTOMATION

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

MASTER OF COMPUTER APPLICATIONS Of



Visvesvaraya Technological University
Belgaum, Karnataka

By

ANKIT SINGH

1CR17MCA02



CMR INSTITUTE OF TECHNOLOGY
132, IT Park Road, Kundalahalli, Bengaluru -560037
2019-2020

A project report on

CUSTOMER SERVICE PORTAL AUTOMATION

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

MASTER OF COMPUTER APPLICATIONS

Of



Visvesvaraya Technological University
Belgaum, Karnataka

By

ANKIT SINGH

1CR17MCA02



CMR INSTITUTE OF TECHNOLOGY
132, IT Park Road, Kundalahalli, Bangalore-560037
2019-2020

A project report on

**CUSTOMER SERVICE PORTAL
AUTOMATION**

Submitted in partial fulfillment of the requirement
for the award of the degree

MASTER OF COMPUTER APPLICATIONS

of

Visvesvaraya Technological University
Belgaum, Karnataka

By

**ANKIT SINGH
1CR17MCA02**

Under the guidance of

Internal Guide

Ms. Gomathi T

Asst.Prof & HoD

Department of MCA,
CMRIT, Bangalore.

External Guide

Mr. Ashok Kumar

Solution Architect

Careerlabs Technologies Pvt. Ltd.,
Bangalore.



CMR INSTITUTE OF TECHNOLOGY

#132, IT Park Road, Kundalahalli, Bangalore-560037

2019-2020

CMR INSTITUTE OF TECHNOLOGY

Department of Master of Computer Applications
Bangalore - 560037



CERTIFICATE

This is to certify that the project work entitled

CUSTOMER SERVICE PORTAL AUTOMATION

Submitted in partial fulfillment of the requirement

for the award of the degree of

Master of Computer Applications of the

Visvesvaraya Technological University, Belgaum, Karnataka

bonafide work carried out by

**ANKIT SINGH
1CR17MCA02**

during the academic year 2019-2020

Signature of the Guide

Ms. Gomathi T
Asst.Prof & HoD, MCA

Signature of the HOD

Ms. Gomathi T
Asst.Prof & HoD, MCA

Signature of the Principal

Dr. Sanjay Jain
PRINCIPAL, CMRIT

External Viva

Name of the Examiners

Signature with date

1.

2.

DECLARATION

I, **ANKIT SINGH**, student of 6th MCA, **CMR Institute of Technology**, bearing the USN **1CR17MCA02**, hereby declare that the project entitled “**CUSTOMER SERVICE PORTAL AUTOMATION**” has been carried out by me under the supervision of External Guide **Mr. Ashok Kumar**, Solution Architect, and Internal Guide **Ms. Gomathi T, Asst.Prof & HoD**, **Department of Master of Computer Applications** and submitted in the partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications by the **Visvesvaraya Technological University** during the academic year 2019-2020. The reports have not been submitted to any other University or Institute for the award of any degree or certificate.

Place: Bangalore

ANKIT SINGH

Date:

(1CR17MCA02)

ACKNOWLEDGMENT

I would like to thank all those who are involved in this endeavor for their kind cooperation for its successful completion. At the outset, I wish to express my sincere gratitude to all those people who have helped me to complete this project in an efficient manner.

I offer my special thanks to my external project guide Mr. Ashok Kumar Solution architect, Careerlabs Technologies Pvt. Ltd., Bangalore, and to my Internal Project guide Ms. Gomathi T, Asst.Prof & HoD, Department of MCA, CMRIT, Bangalore without whose help and support throughout this project would not have been this successful.

I am thankful to Dr. SANJAY JAIN, Principal, CMRIT, Bangalore for his kind support in all respects during my study. I would like to thank Mr. Ashok Kumar Solution architect, Careerlabs Technologies Pvt. Ltd., Bangalore, who gave an opportunity to do this project at an extreme organization Most of all and more than ever, I would like to thanks my family members for their warmness, support, encouragement, kindness, and patience. I am really thankful to all my friends who always advised and motivated me throughout the course.

ANKIT SINGH

(1CR17MCA02)



 www.TheCareerLabs.com
 support@TheCareerLabs.com

Date: 25th May 2020

PROJECT COMPLETION LETTER

We are glad to inform you that **Mr. Ankit Singh** from **CMR Institute of Technology** has successfully completed his project work at **Careerlabs Technologies Pvt Ltd** from 24-Dec-2019 from to 20-May-2020.

He has worked on a project titled "**Customer Service Portal Automation**".

We found him extremely inquisitive and was very much interested to learn different functions of Automation. He was also willing to learn the subject matter in depth and understand it better.

His association with us was very fruitful and we wish him all the best for his future endeavors.

For Careerlabs Technologies Pvt Ltd



Authorized Signatory

S.NO.	Contents	Page No.
1.	Introduction	
	1.1 Project Description	1
	1.2 Company Profile	3
2.	Literature Survey	
	2.1 Existing System and Proposed System	6
	2.2 Feasibility Study	7
	2.3 Hardware and Software Requirements	8
3.	Software Requirement Specification	
	3.1 Functional Requirements	09
	3.2 Non-Functional Requirements	10
4.	System Design	
	4.1 Bot Perspective	11
	4.2 Context Diagram	
	4.2.1 Data flow Diagram	13
5.	Detailed Design	
	5.1 Use Case Diagrams	
	5.1.1 Use Case Diagram for RASA Chatbot	16
	5.2 Sequence Diagrams	
	5.2.1 Sequence Diagram for admin	17
	5.2.2 Sequence Diagram for user	17
	5.3 Activity Diagrams	
	5.3.1 Activity Diagram for Career Domain	18
	5.3.2 Activity Diagram of Personal Query Response	18
	5.3.3 Normal Conversation Response Activity	18

	5.4 ER Diagrams	
	5.4.1 ER Diagram for RASA Chatbot	19
6.	Implementation	20
	6.1 Screenshots	21
	6.2 Source Code	29
7.	Software Testing	37
8.	Conclusion	40
9.	Future Enhancements	41
10.	Bibliography	42
11.	User Manual	43

INTRODUCTION

1.1 Project Description:

We introduce a CUSTOMER SERVICE as a Chatbot, Rasa X is open-source python libraries for building conversational Chatbot. The part of ML-based chatbot flow management and able to understand language accessible to those who are not specialist software developers. In terms of the design of chatbot, we provide a simple use and bootstrapping from the initial training responses dataset. The purpose of this Chatbot is to showcase a contextual AI assistant built with the open-source Rasa framework.

Careerguru RASA CHATBOT is a beta version it is helping CUSTOMER SERVICE to more reliable. It has the following user goals:

- Discover Career Tracks test
- Answering some FAQs around CareerLabs
- Directing technical questions to specific documentation

What is the RASA AI chatbot?

A RASA AI chatbot is made with pre-programmed responses and it can also build with AI to answer the questions. RASA Chatbot did not require any human support for responses. RASA artificial intelligence chatbot can be built with voice assistance. The RASA AI chatbot able to understand the natural language of humans, based on the response it will modifying the NLU dataset or we can it will generate a new data model after training. The RASA AI NLU is an open platform to work. In rasa chatbot, we can work with different chatbots or voice assistance. RASA AI Chatbots are basically two types Rule-based, Self-learning. The first one is some time to fail when users ask complex queries. The second one is able to learn by itself. It is mainly programmed by LSTM (Time series). RASA framework gives highly sophisticated NLP solutions. Here come RASA AI Chatbot and Dialogflow, here Dialogflow is a chatbot platform without any programming language. For full control and modifiable RASA AI Chatbot, we use RASA AI chatbot that why RASA AI Chatbot came into this picture.

Why Should we choose the RASA Platform?

RASA AI Chatbots are basically two types:

1. **Rule-based**
2. **Self-learning**

The first one is some time to fail when users ask complex queries. The second one is able to learn by itself. It is mainly programmed by LSTM (Time series).

RASA framework gives highly sophisticated NLP solutions. Here come RASA AI Chatbot and Dialogflow, here Dialogflow is a chatbot platform without any programming language. For full control and modifiable RASA AI Chatbot, we use RASA AI chatbot that why RASA AI Chatbot came into this picture.

RASA AI Chatbot has two components –

- **RASA chatbot NLU (Natural Language Understanding):** The RASA AI chatbot able to understand the natural language of humans, based on the response it will modify the NLU dataset or we can it will generate a new data model after training. The RASA AI NLU is an open platform to work. In RASA chatbot, we can work with different chatbots or voice assistance. NLU has many number of different pipeline components Once is training data have done after that we feed to NLU pipeline.
- **The RASA AI Core:** The core of the RASA chatbot is built with AI features that are helping the developers. The Core is using ML(Machine learning) to make a model trained. The RASA AI core based on NLU and LSTM neural network. The RASA AI chatbot is open source platform based on python programming language.

1.2 Company Profile:

In the **CareerLabs**, in Partnership with **BYJU'S-The Learning App**, we help Graduate Students build a strong profile in line with their Career Ambitions, be it for Higher Studies or Placements.

Headquartered at Bangalore, CareerLabs is associated with over 150 Engineering Colleges in Bangalore, Chennai, Hyderabad, Pune, and many more cities to help graduate students with Placements, GMAT, GRE, CAT, UPSC, and GATE. We help students choose the right paths through Career Tracks selection, Industry Mentorship Integration, and Higher Completion Rates for Upskilling Course Completions through our best-in-class Knowledge Partners who bring on-board years of experience in delivering value to students in upskilling.

The platform on CareerLabs brings in an amalgamation of content pertaining to Aptitude Skills required to ace the entry tests, Soft Skills required to ace the Personal Interviews and Group Discussions, and Technical Content that will keep students on top of the applicant list.

Be it the concept of Trigonometry or the concept of speech anxiety, the platform combines multiple sources of content into one so that the students don't have to refer to many materials to make them employment-ready.

The aim is one: To make more Students Industry Ready, Employable, and to Empower them with all possible skills in that pursuit through our platform. If pursuing a master's degree is the catalyst in taking the student's profile to the next dimension, then our world-class content with tested strategies on GMAT, GRE, GATE, CAT, and UPSC are suited to help the student top these entrance tests.

CareerLabs is the go-to place for all the career needs of students and young graduates. We offer a wide range of preparatory programs for national and international entrance exams, as well as Admissions Consulting Services for Graduate and Business Schools, both abroad and in India.

We are a diverse team of highly-experienced professionals and educators with a passion for helping you discover your full potential as you prepare for a life beyond college.

With each passing year, we in CareerLabs, strive to become more committed to the cause of helping you succeed in life.

CareerLabs is focused on student career building and helps to get placed. CareerLabs has already helped 1000's of students craft their dream careers be it for placements or higher studies. Headquartered in Bangalore, CareerLabs is impacting the lives of 1000's of students across India.

At the CareerLabs, in Partnership with **BYJU'S-The Learning App**, we help Graduate Students build a strong profile in line with their Career Ambitions, be it for Higher Studies or Placements.

LITERATURE SURVEY

2.1 Existing System and Proposed System:

Existing System

There are many different types of chatbots available for customer service, example simple talking chatbot or survey filling chatbot so on but they have some drawbacks that are they all have static functionalities. This type of chatbot response is always the same, particularly what users asking or some time bot not able to understand user responses.

Proposed System

This chatbot we can embed on any website. This chatbot is using neural network to build the RASA chatbot. The career guru chatbot is using ML features to predict the career track of a student based on some random question around student interest.

The career guru chatbot is using LSTM neural network algorithm features to help action in the chatbot. Others chatbot using a bunch of if/else in their chatbot, but RASA chatbot using ML and AI features to help in build the chatbot.

2.2 Feasibility Study:

We introduce a CUSTOMER SERVICE as a Chatbot, Rasa X is open-source python libraries for building conversational Chatbot. The part of ML-based chatbot flow management and able to understand language accessible to those who are not specialist software developers.

In terms of the design of chatbot, we provide a simple use and bootstrapping from the initial training responses dataset.

The purpose of this Chatbot is to showcase a contextual AI assistant built with the open-source Rasa framework.

Careerguru RASA CHATBOT is a beta version it is helping CUSTOMER SERVICE to more reliable. It have the following user goals:

- Discover Career Tracks test
- Answering some FAQs around CareerLabs
- Directing technical questions to specific documentation

2.3 Hardware and Software Requirements:

Hardware Requirements

HDD: 40 GB

RAM: 2 GB

Processor: Pentium 4 CPU 3.06 GHz

Software Requirements

Operating System: Windows 10, Linux, and Mac OS

Database: MongoDB

Additional: RASA X, Chrome Browser

Language: Python 3.6 + , HTML, CSS, JavaScript

SOFTWARE REQUIREMENT SPECIFICATION

3.1 Functional Requirements:

- The RASA AI chatbot should respond to student queries.

All use cases

- If rasa chatbot does not understand the student input then it should display a message (ex." I didn't get you, can you type again").
 - If rasa chatbot understand the student input then it should display a current message.
-
- The RASA chatbot able to send user or student details in the following ways:
 - Student detail should be in string format.
 - The RASA chatbot should work fine with MSG91 API.
 - The RASA chatbot should be able to create user data and store in cloud database. Here we are using MongoDB database.

3.2 Non-Functional Requirements:

- The RASA chatbot have to response within 25-30 second.
- When you talk with RASA chatbot, it should feel like a you are talking with human not a chatbot so the response should be always 1-2 second delay.
- There are kinds of bot users
 - Student who are logged in careerlabs website. They can query about their career track domain.
 - Developers can modify and update the query if bot will get any errors.
 - The tester can test the chatbot a student or a user they see all scenarios.
- The RASA chatbot should authenticate students before starting a career track test.
- If students do not authenticate then chatbot should ask for authentication.
- The RASA chatbot main logic file should be separate from other files of the chatbot.

SYSTEM DESIGN

4.1 Bot Perspective

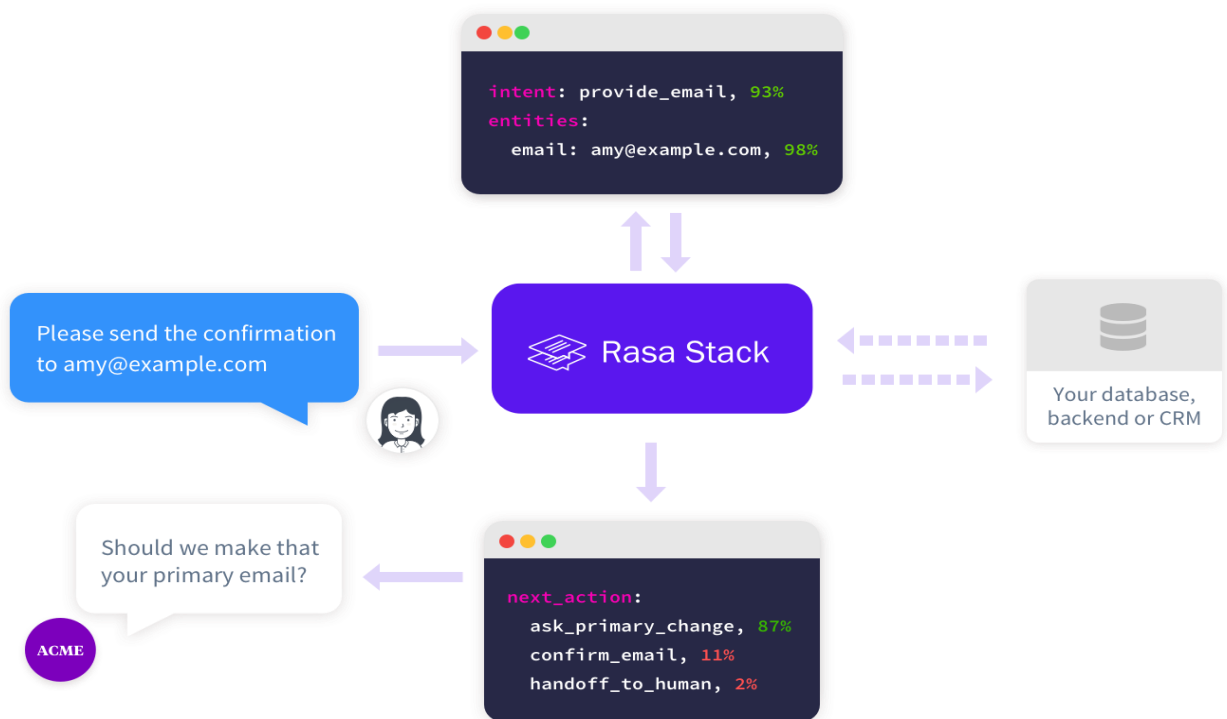
A RASA AI chatbot is made with pre-programmed responses and it can also build with AI features to answer the questions. RASA Chatbot did not require any human support for responses. RASA artificial intelligence chatbot can be built with voice assistance.

The RASA AI chatbot able to understand the natural language of humans, based on the response it will modifying the NLU dataset or we can it will generate a new data model after training. The RASA AI NLU is an open platform to work. In rasa chatbot, we can work with different chatbots or voice assistance. RASA AI Chatbots are basically two types Rule-based, Self-learning. The first one is some time to fail when users ask complex queries. The second one is able to learn by itself. It is mainly programmed by LSTM (Time series). RASA framework gives highly sophisticated NLP solutions. Here come RASA AI Chatbot and Dialogflow, here Dialogflow is a chatbot platform without any programming language. For full control and modifiable RASA AI Chatbot, we use RASA AI chatbot that why RASA AI Chatbot came into this picture.

RASA chatbot NLU (Natural Language Understanding): The RASA AI chatbot able to understand the natural language of humans, based on the response it will modifying the NLU dataset or we can it will generate a new data model after training.

The RASA AI NLU is an open platform to work. In rasa chatbot, we can work with different chatbots or voice assistance. NLU has many number of different pipeline components Once is training data have done after that we feed to NLU pipeline.

RASA Core: The core of the rasa chatbot is built with AI features that are helping the developers. The Core is using ML(Machine learning) to make a model trained. The RASA AI core based on NLU and LSTM neural network. The RASA AI chatbot is open source platform based on python programming language.



The Architecture of RASA Stack

Allowing users to access the information regarding Career Domain.

- Allowing Users to query regarding Their Domain.
- Users are able to see more details about domains.
- Users should be able to know the specific domain
- User can have a general conversation like how he/she interacts with his/her friends.
- Providing a responsive user interface for the application.

4.2 Context Data Flow Diagram

Data flow diagram of User

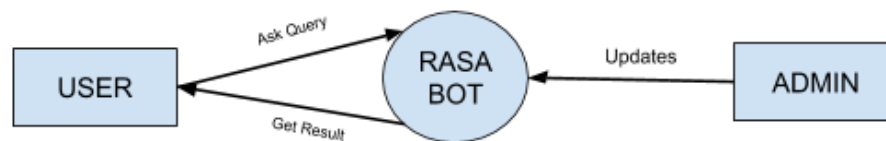


Fig. User Data Flow

Data flow diagram of Sign in and Authentication

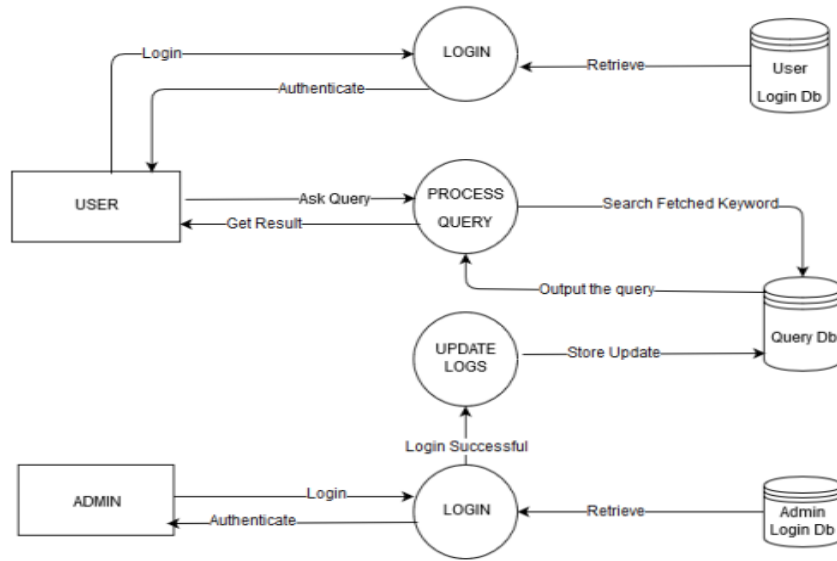


Fig. Sign in and Admin

Data flow diagram of User input in detail

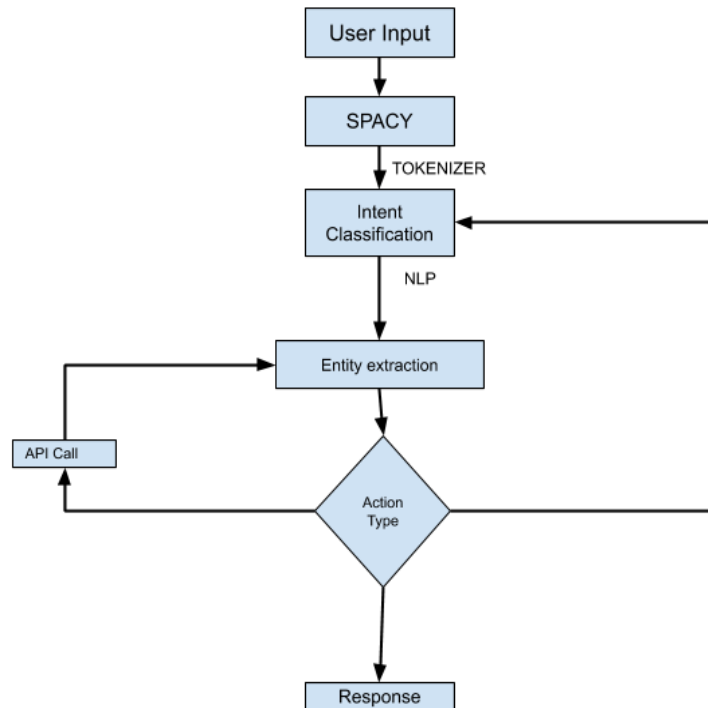
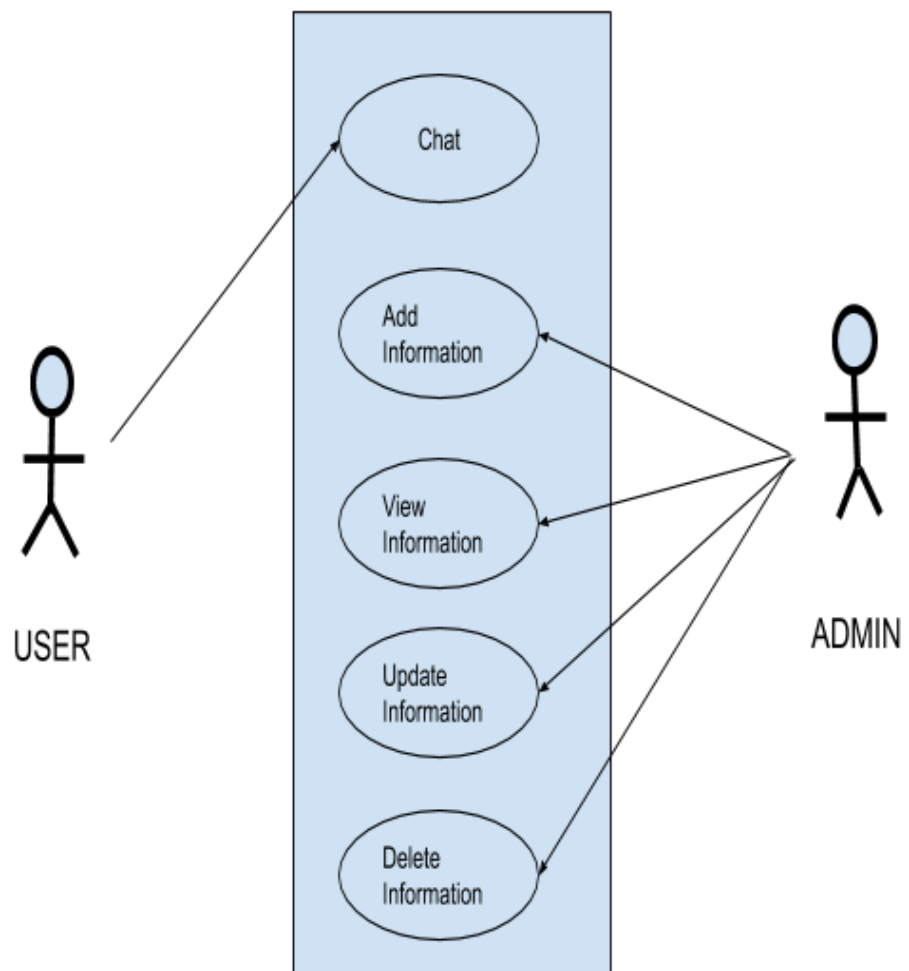


Fig. Data flow in Detail with NLP model

5. DETAILED DESIGN

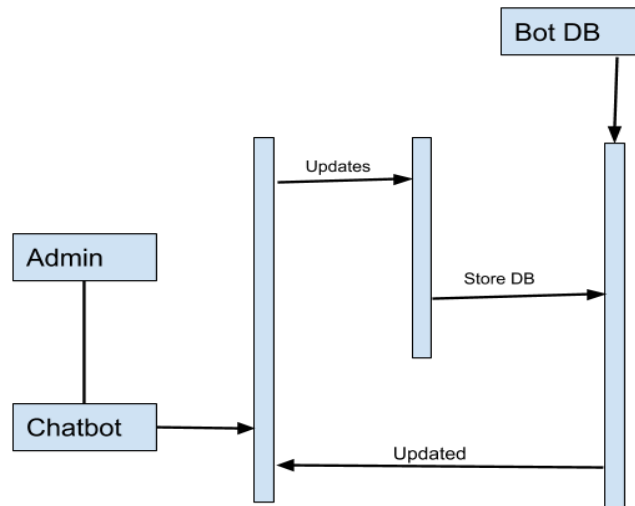
5.1 Use Case Diagrams

5.1.1 Use Case Diagram for RASA Chatbot

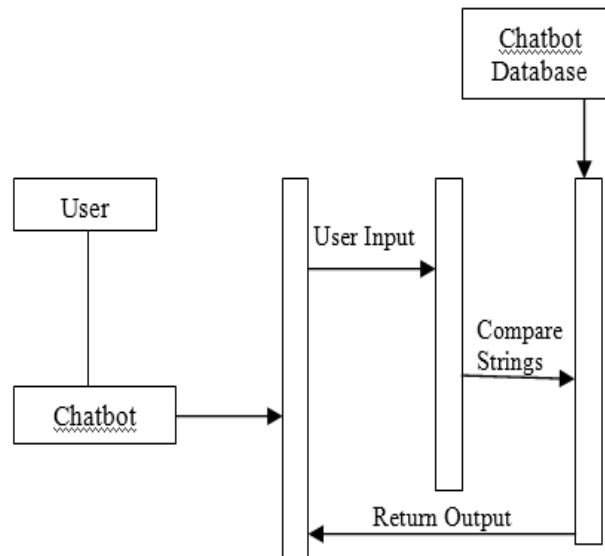


5.2 SEQUENCE DIAGRAM

5.2.1 SEQUENCE DIAGRAM FOR ADMIN

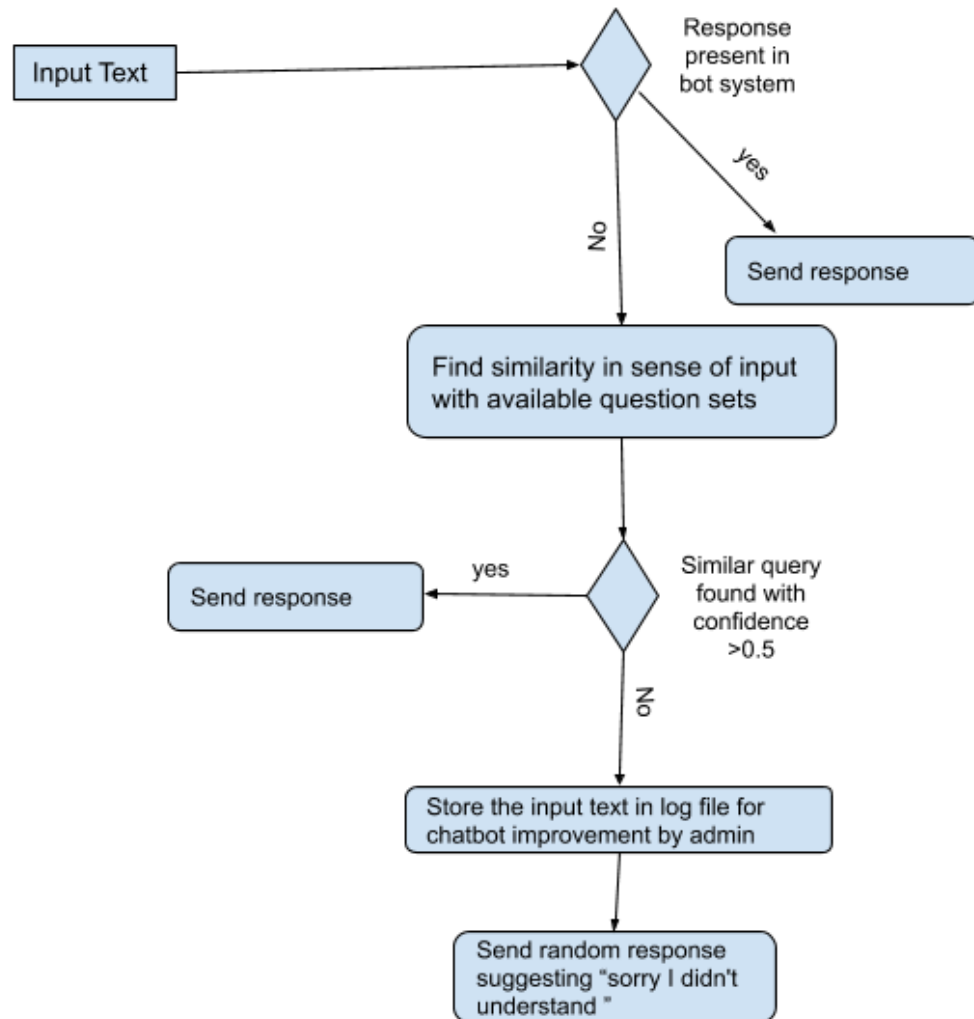


5.2.2 SEQUENCE DIAGRAM FOR USER

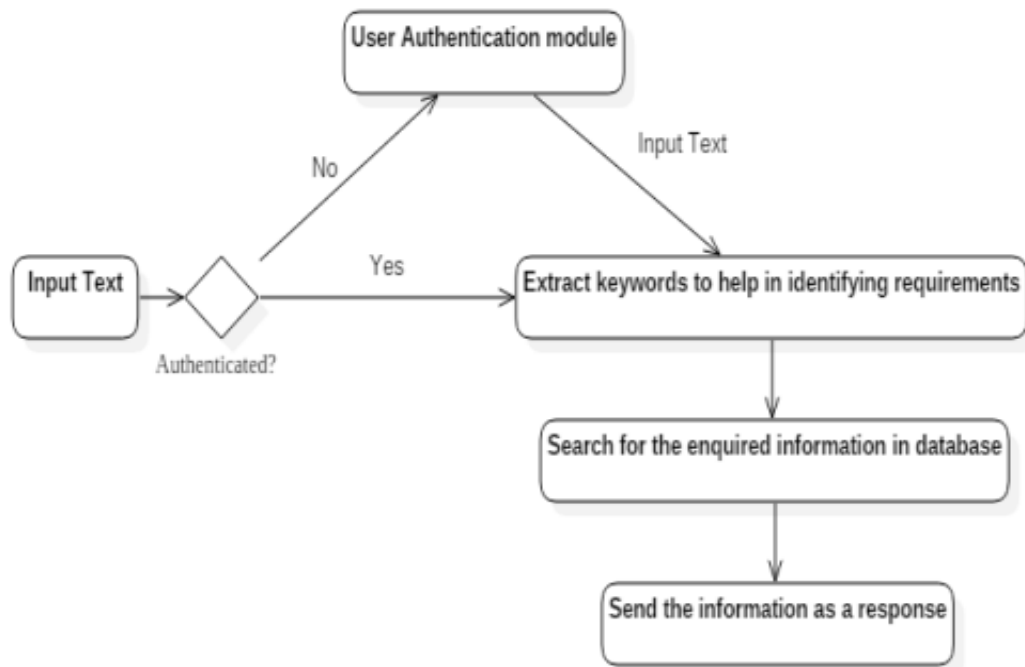


5.3 ACTIVITY DIAGRAM

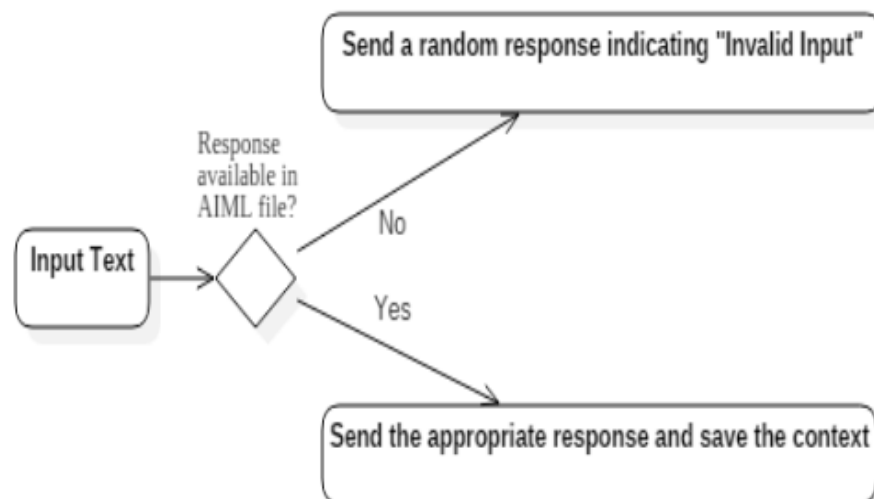
5.3.1 ACTIVITY DIAGRAM FOR CAREER DOAMIN



5.3.2 ACTIVITY DIAGRAM OF PERSONAL QUERY RESPONSE

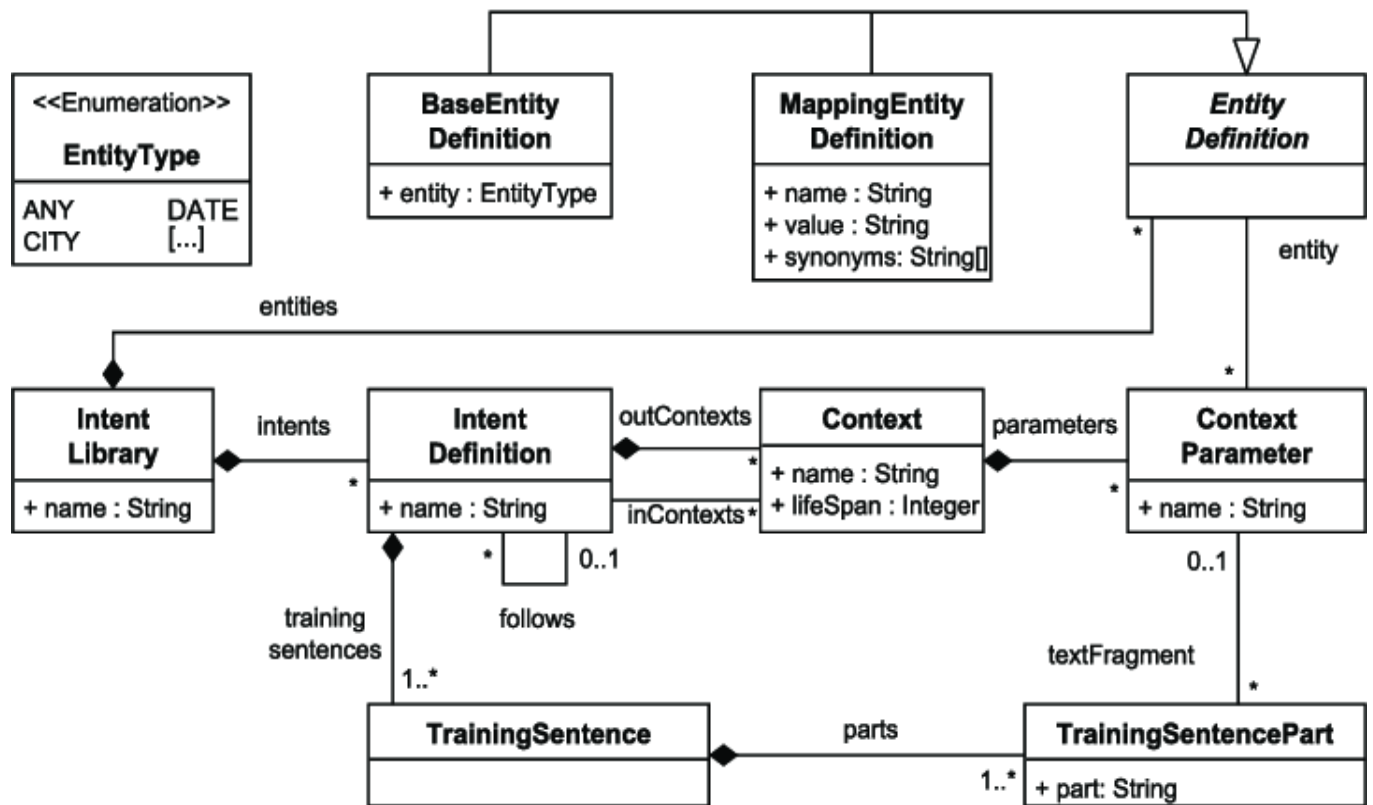


5.3.3 Normal Conversation Response Activity



5.4 ER Diagrams

5.4.1 ER Diagram for RASA Chatbot

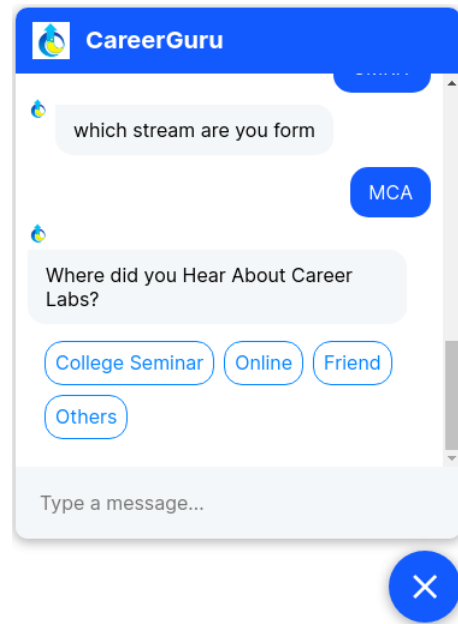
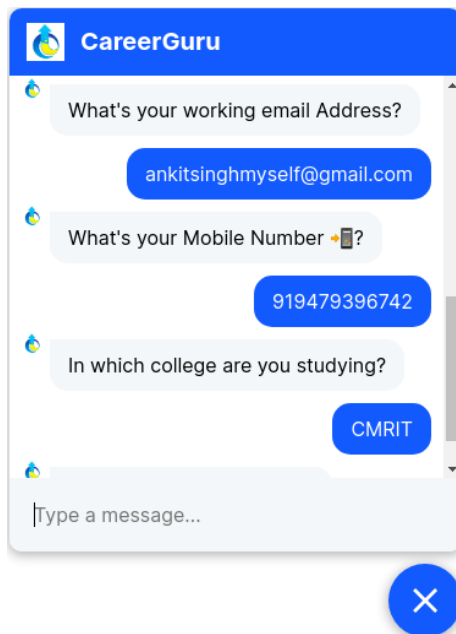
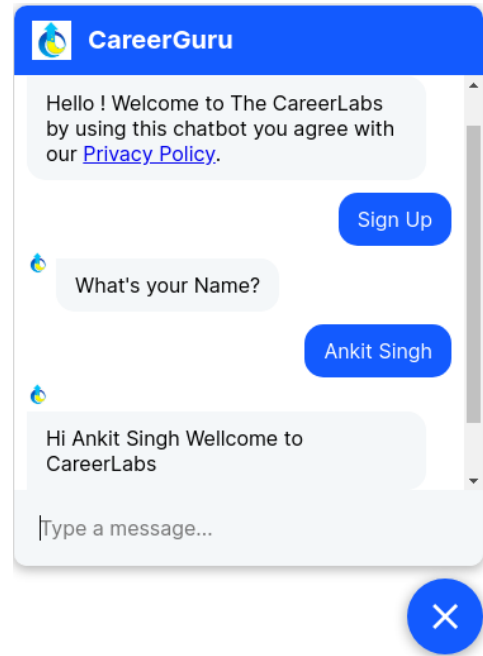
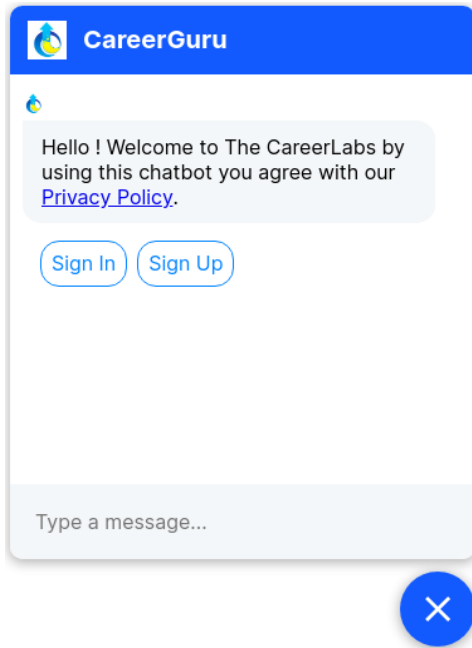


IMPLEMENTATION

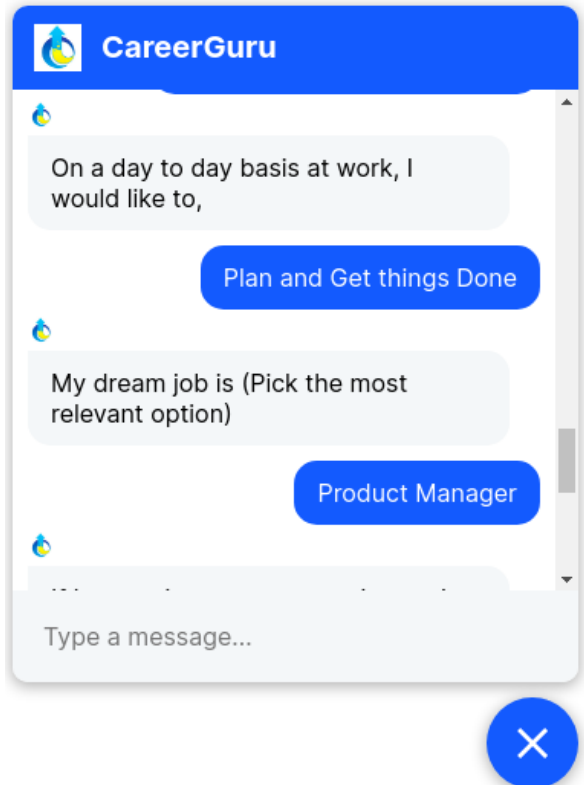
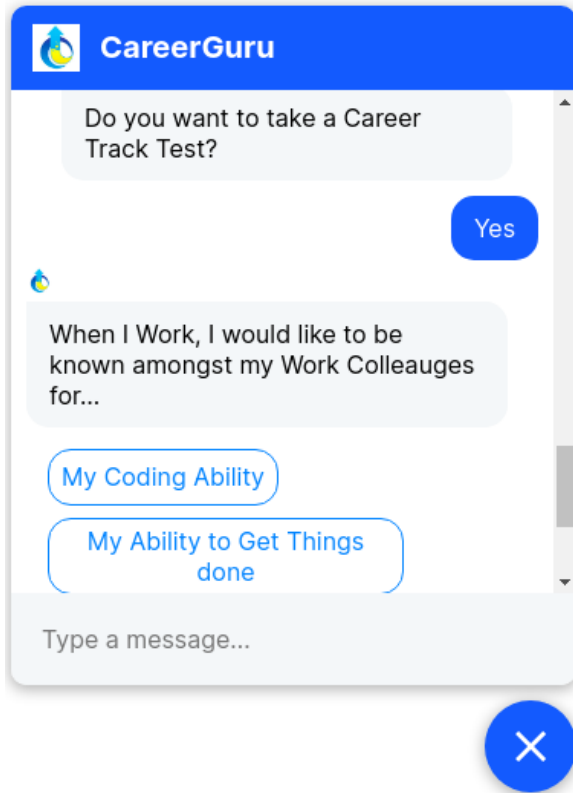
The implementation is a realization of the technical specification as a program, software component, and deployment. Some of the implementations may exist for a given specification or standard. This project is implemented using PYTHON and RASA. Installing Anaconda means you are getting a version of Python Installed (3.6 currently). The Conda distro provides, first and foremost, a Python distribution outfitted with easy access to the packages often used in data science: NumPy, Pandas, Matplotlib, and so on. They're not simply bundled with Anaconda, but available via a custom package management system called Conda. Conda Installed packages can include tricky external binary dependencies that couldn't be managed through Python's own Pip. Each package is kept up-to-date by Anaconda, and many of them are compiled with the Intel MKL extensions for speed. The other major advantage Anaconda confers is a graphical environment, the Anaconda Navigator. The Navigator isn't an IDE, but rather a convenient GUI front end for Anaconda features including the Conda package manager and user-configured virtual environments. You can also use Navigator to manage third-party applications such as Jupyter notebooks and the Visual Studio Code IDE.

6.1 Screenshots

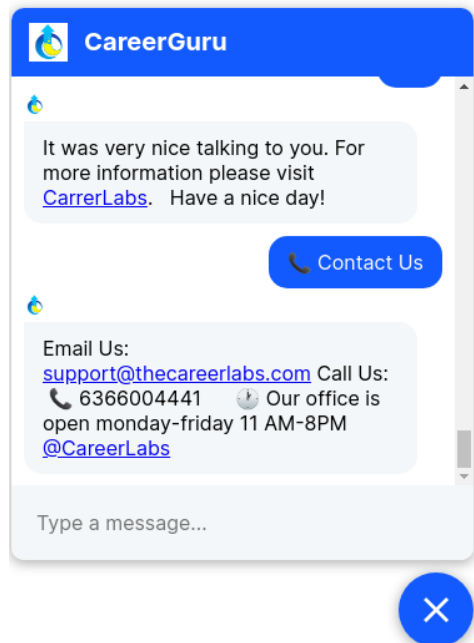
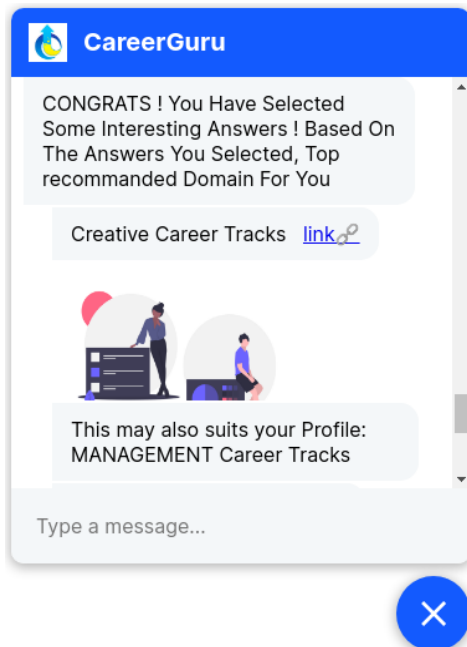
- Welcome Screen Sign in and Sign Up



- **Career Domain test**



- **Career domain Result**



● Conversations

The screenshot shows a 'Conversations (96)' interface. On the left is a list of messages from 'Tester' with timestamps ranging from 11:54 am to 2:55 pm on Feb 20, 2020. The main area displays a conversation with 'questions_form2'. The messages include:

- CONGRATS ! You Have Selected Some Interesting Answers ! Based On The Answers You Selected, Top recommended Domain For You
- An illustration of a person at a laptop interacting with a robot.
- Core Engineering Career Tracks
- This may also suits your Profile: CODING Career Tracks

On the right, a summary indicates 'Conversation between Demobot and Tester'.

● RASA Models

The screenshot shows the 'Models' interface with an 'Upload model' button. It contains a table with the following data:

Model name	Created on	State
<input type="checkbox"/> 20200320-115734	Fri Mar 20 2020 11:57:32 GMT+0530	Active
<input type="checkbox"/> 20200306-132730	Fri Mar 06 2020 13:27:30 GMT+0530	
<input type="checkbox"/> 20200306-132124	Fri Mar 06 2020 13:21:24 GMT+0530	
<input type="checkbox"/> 20200306-131557	Fri Mar 06 2020 13:15:57 GMT+0530	
<input type="checkbox"/> 20200219-113736	Wed Feb 19 2020 11:36:21 GMT+0530	

- **NLU training (training Data)**

NLU Training interface showing training data. The 'Training data' tab is active. The table displays the following data:

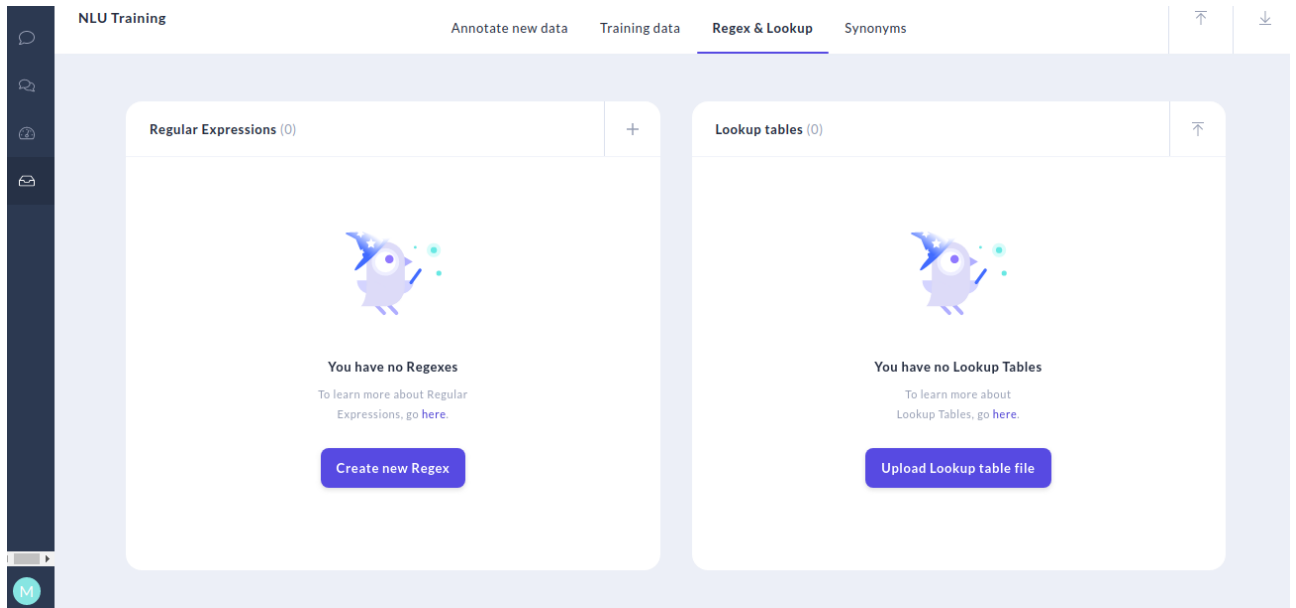
Sentence	Intent
/restart	restart_bot
RESTART	restart_bot
Restart	restart_bot
Resend OTP	reSend_otp
Resend	reSend_otp
4556	otp_codes

- **NLU Training (Annotate new data)**

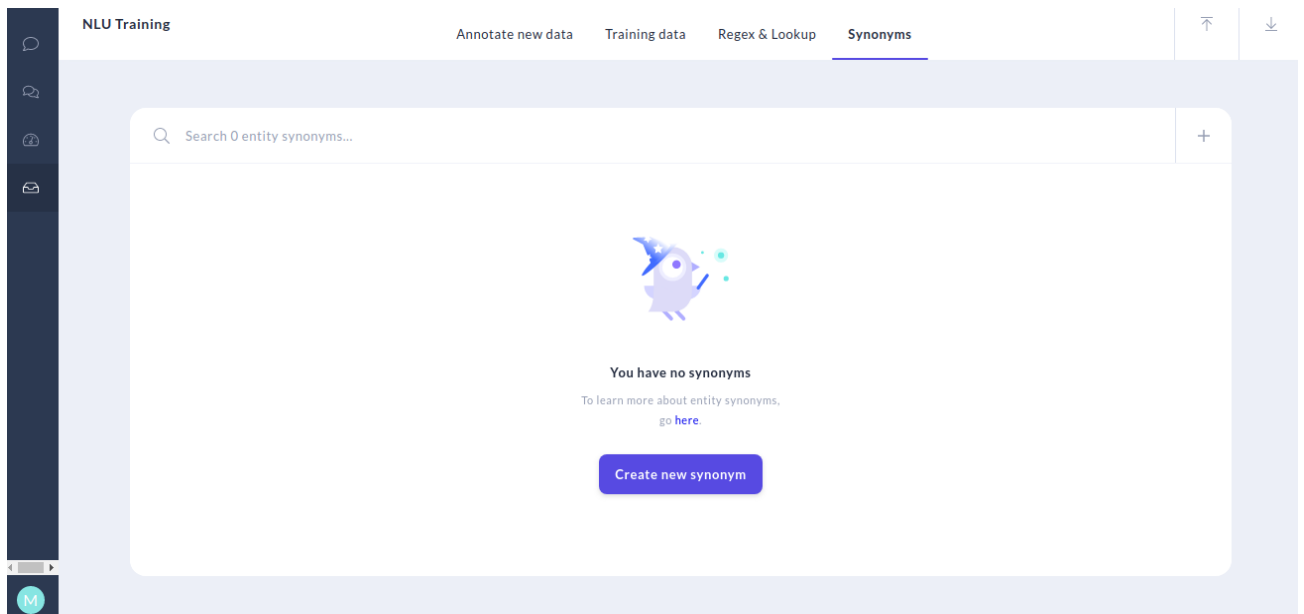
NLU Training interface showing 'Annotate new data'. The 'Annotate new data' tab is active. The table displays the following data:

Sentence	Predicted Intent and Confidence
<input type="checkbox"/> Hi_welcome_to_career_lab	hearAbout (0.96)
<input type="checkbox"/> anki32	user_names (0.67)
<input type="checkbox"/> ankit32	user_names (0.31)
<input type="checkbox"/> anskn334	user_names (0.81)
<input type="checkbox"/> asm@gmail.com	emails (1.0)

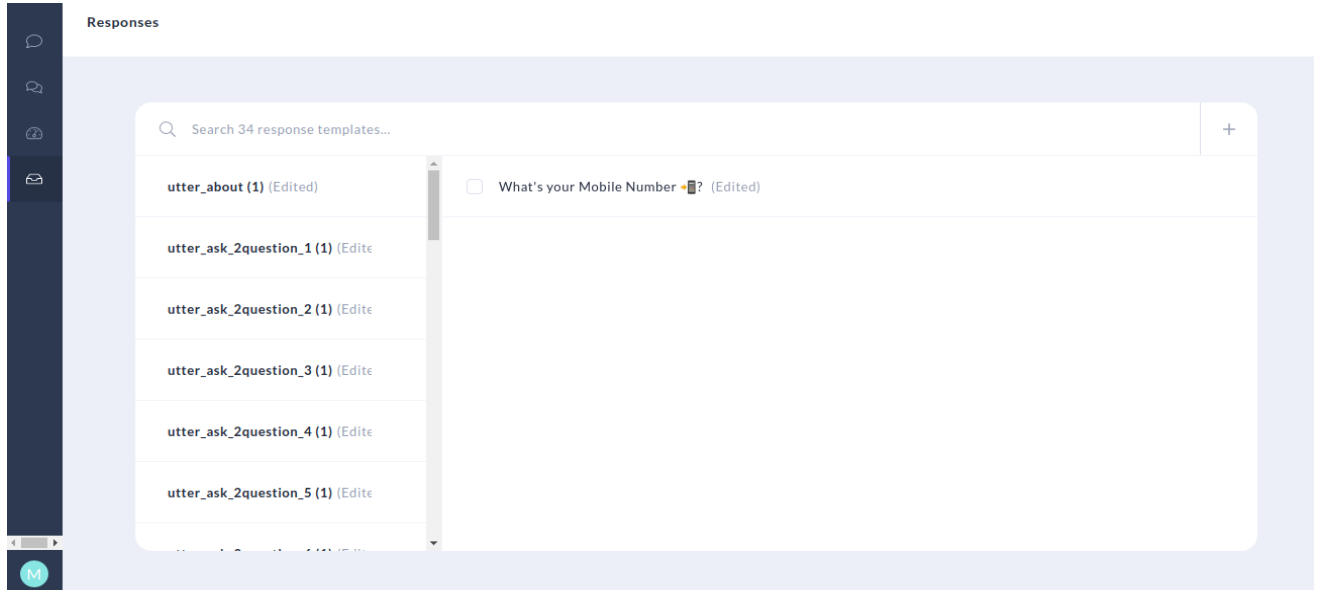
- **NLU Training (Regex & Lookup)**



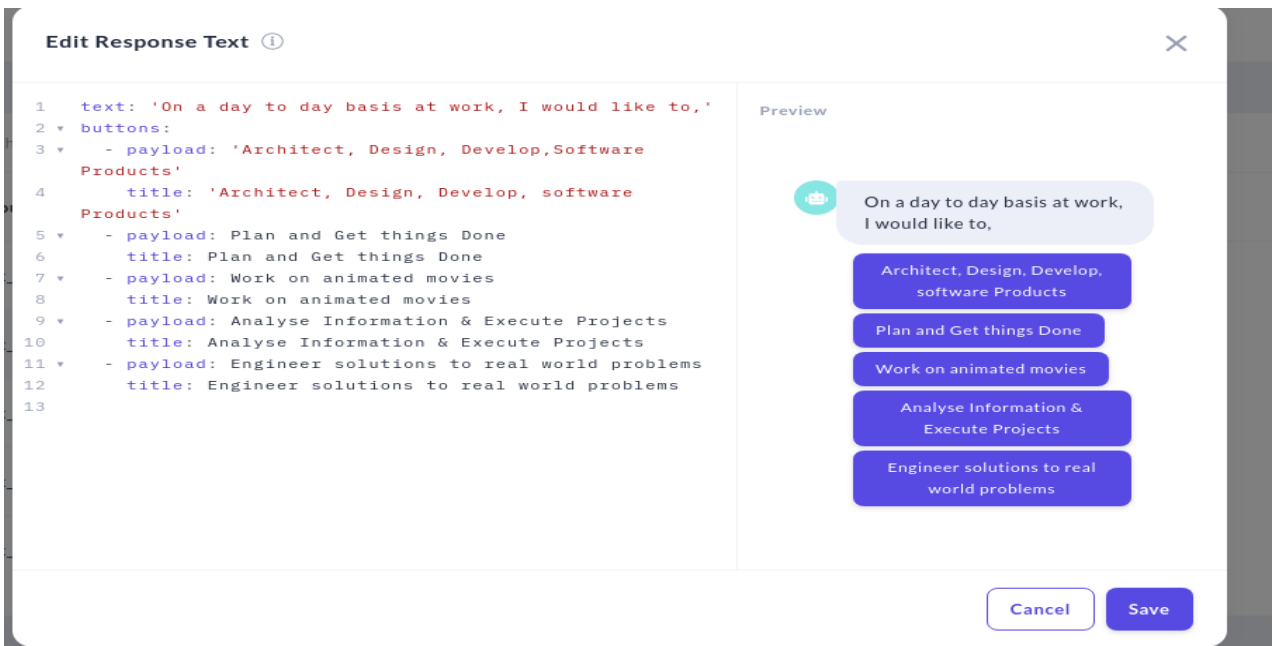
- **NLU Training (Synonyms)**



- **Responses**



- **Edit Response Text**



● Stories

Stories
Search 41 stories...
+
↓

- New Story
- New Story
- New Story
- New Story
- New Story
- New Story
- New Story
- New Story
- New Story
- test1
- New Story
- New Story
- New Story

New Story
🗑️

```

1 ## New Story
2
3 * greet
4   - utter_greet
5 * Sign_Up
6   - sign_up_form
7     - slot{"requested_slot":"password"}
8 * password
9   - sign_up_form
10  - slot{"password":"123546"}
11  - slot{"requested_slot":"email"}
12 * emails
13   - sign_up_form
14   - slot{"email":"anki@gmail.com"}
15   - slot{"requested_slot":"phno"}
16 * password
17   - sign_up_form
18   - slot{"phno":"9856742310"}
19 * colleges
20   - sign_up_form
21   - slot{"collegeName":"cmr"}
22   - slot{"requested_slot":"stream"}
23 * stream
24   - sign_up_form
25   - slot{"stream":"mca"}
26   - slot{"requested_slot":"hearAbout"}
                    
```

Flow
⚙️

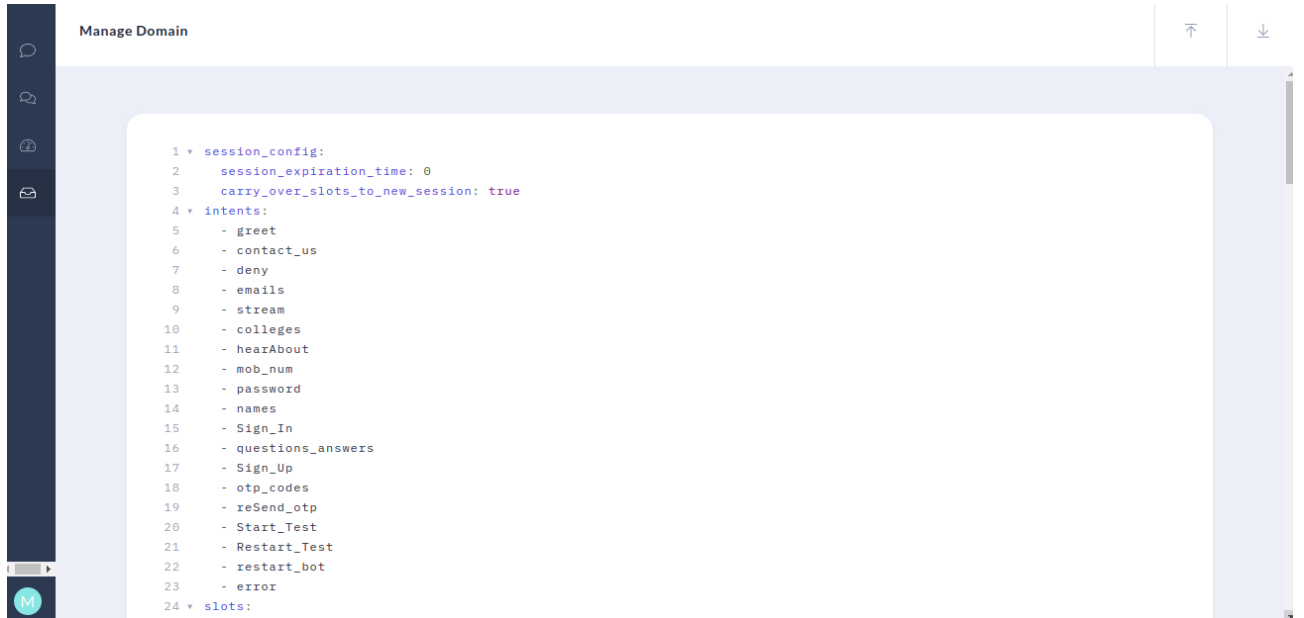
● Model Configuration

Model Configuration ⓘ
Discard
Save changes

```

1 language: en
2 pipeline: supervised_embeddings
3 policies:
4   - name: MemoizationPolicy
5   - name: KerasPolicy
6   - name: MappingPolicy
7   - name: FormPolicy
8
                    
```

- **Manage Domain**



The screenshot shows a web interface titled "Manage Domain". On the left is a dark sidebar with navigation icons. The main content area displays a JSON configuration for a domain. The configuration includes session settings and a list of intents.

```
1 * session_config:
2   session_expiration_time: 0
3   carry_over_slots_to_new_session: true
4 * intents:
5   - greet
6   - contact_us
7   - deny
8   - emails
9   - stream
10  - colleges
11  - hearAbout
12  - mob_num
13  - password
14  - names
15  - Sign_In
16  - questions_answers
17  - Sign_Up
18  - otp_codes
19  - reSend_otp
20  - Start_Test
21  - Restart_Test
22  - restart_bot
23  - error
24 * slots:
```

6.2 Source Code:

Code for Login and Registration:

```
class SignUpForm(FormAction):

    def name(self):
        return 'sign_up_form'
    @staticmethod
    def required_slots(tracker):
        return ["full_name","password","email","phno","collegeName","stream","hereAbout"]
    def slot_mappings(self) -> Dict[Text, Union[Dict, List[Dict]]]:
        return {
            "full_name": [
                self.from_text(),
            ],
            "password": [
                self.from_text(),
            ],
            "email": [
                self.from_text(),
            ],
            "phno": [
                self.from_text(),
            ],
            "collegeName": [
                self.from_text(),
            ],
            "stream": [
                self.from_text(),
            ],
            "hereAbout": [
                self.from_text(),],
        }

def validate(self, dispatcher, tracker, domain):
    slot_values = self.extract_other_slots(dispatcher, tracker, domain)
    slot_values.update(self.extract_requested_slot(dispatcher, tracker, domain))
    hearAbout=["College Seminar","Online","Friend","Others"]
    #myclient = pymongo.MongoClient("localhost", 27017)
    #mydb=myclient["rasa"]#database name
    #mycol=mydb["user_details"] #it is a collection name in the db rasa .
    for slot, value in slot_values.items():
        if slot == 'full_name':
```

```

#x = mycol.insert_one({"Name":value})
#print(x)
dispatcher.utter_message("Hi {} Wellcome to CareerLabs ".format(value))
#dispatcher.utter_message("Please Enter username ")
dispatcher.utter_message("Please type The Password")
elif slot == 'password':
    SpecialSym =['$', '@', '#', '%']
    if len(value) < 6:
        dispatcher.utter_message("length should be at least 6")
        slot_values[slot] = None
    elif len(value) > 20:
        dispatcher.utter_message('length should be not be greater than 20')
        slot_values[slot] = None

    elif not any(char.isdigit() for char in value):
        dispatcher.utter_message('Password should have at least one numeral')
        slot_values[slot] = None

    elif not any(char.isupper() for char in value):
        dispatcher.utter_message('Password should have at least one uppercase letter')
        slot_values[slot] = None

    elif not any(char.islower() for char in value):
        dispatcher.utter_message('Password should have at least one lowercase letter')
        slot_values[slot] = None

    elif not any(char in SpecialSym for char in value):
        dispatcher.utter_message('Password should have at least one of the symbols $#@#')
        slot_values[slot] = None
    else:
        print("give password")
elif slot == 'email':
    email = re.compile(r'([a-zA-Z0-9_+]+@[a-zA-Z0-9]+\.[a-zA-Z0-9-]+)$')
    mo = email.search(value)
    if mo == None:
        dispatcher.utter_message("Please Enter valid email Address! ")
        slot_values[slot] = None

    else:
        #x = mycol.insert_one({"Email_Address":value})
        print("x")
        #dispatcher.utter_message("Please Enter Working Mobile Number")
elif slot == 'phno':
    if len(value) == 12 and value.isdigit() == True:
        if value.startswith('91'):
            #dispatcher.utter_message("Please Enter The Varfication OTP")
            print(value)
            #print("above")
        else:
            dispatcher.utter_message("Please include country code 91")
            slot_values[slot] = None

```

```

else:
    dispatcher.utter_message("Please provide valid mobile number including country code 91")
    slot_values[slot] = None
elif slot == 'collegeName':
    if value.replace(" ", "").isalpha() == False:
        dispatcher.utter_message("Please Enter valid College Name! ")
        slot_values[slot] = None
    else:
        #x = mycol.insert_one({"College_Name":value})
        print("x")
        #dispatcher.utter_message("Which Stream are You From?")
elif slot == 'stream':
    if value.replace(" ", "").isalpha() == False:
        dispatcher.utter_message("Please Enter valid Stream! ")
        slot_values[slot] = None
    else:
        #x = mycol.insert_one({"Stream":value})
        print("x")

elif slot == 'hereAbout':
    if value not in hearAbout:
        dispatcher.utter_message("Please Choose from Options! ")
        slot_values[slot] = None
    else:
        #x = mycol.insert_one({"Hear_About":value})
        print("x")

'''
elif slot == 'gender':
    if value.lower() not in ["male", "female"]:
        dispatcher.utter_template('utter_ask_question', tracker)
        slot_values[slot] = None
elif slot == 'dob':
    try:
        datetime.strptime(value, '%d-%m-%Y')
    except ValueError:
        dispatcher.utter_template('utter_ask_question', tracker)
        slot_values[slot] = None'''
return [SlotSet(slot, value) for slot, value in slot_values.items()]

```



```

def submit(self,dispatcher: CollectingDispatcher,tracker: Tracker,domain: Dict[Text, Any],) -> List[Dict]:
    firstname=tracker.get_slot("full_name")

    #gender=tracker.get_slot("gender")
    #dob=tracker.get_slot("dob")
    email=tracker.get_slot("email")
    contact=tracker.get_slot("phno")
    collegeName=tracker.get_slot("collegeName")
    stream=tracker.get_slot("stream")
    hereAbout=tracker.get_slot("hereAbout")
    #state=tracker.get_slot("state")
    #zipcode=tracker.get_slot("zipcode")
    #ssn=tracker.get_slot("ssn")
    #SlotSet('zipcode', None)
    #SlotSet('zipcode', "11111")
    dispatcher.utter_message("Here is the details you have entered.Please review")
    dispatcher.utter_message("Name : {} \nemail : {} \ncontact : {} | \nCollege Name : {}
(firstname,email,contact,collegeName,stream,hereAbout))
    dispatcher.utter_message("Thank You for Sign Up @careerLabs")
    dispatcher.utter_template("utter_ask_start_test",tracker)
    return []

#####
class ActionCheckOtp(Action):
    def name(self) -> Text:
        return "action_check_otp"

    def run(self,
            dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        dispatcher.utter_message("OTP Successfully Send")
        otpobj.retry(tracker.get_slot("phno"))
        return []
class SignInForm(FormAction):#sign in form

    def name(self):
        return 'sign_in_form'
    @staticmethod
    def required_slots(tracker):
        return ["phno","otp"]
    def slot_mappings(self) -> Dict[Text, Union[Dict, List[Dict]]]:
        return {
            "phno": [
                self.from_text(),
            ],
            "otp": [
                self.from_text(),
            ]
        }
    def validate(self, dispatcher, tracker, domain):
        slot_values = self.extract_other_slots(dispatcher, tracker, domain)
        slot_values.update(self.extract_requested_slot(dispatcher, tracker, domain))
        #db connection
        #myclient = pymongo.MongoClient("localhost", 27017)
        #mydb=myclient["rasa"]#database name
        #mycol=mydb["conversations"] #it is a collection name in the db rasa .

```

```

for slot, value in slot_values.items():
    if slot == 'phno':
        if len(value) == 12 and value.isdigit() == True:
            if value.startswith('91'):
                print(otpobj.send(value,'msgind',sendotp.generateOtp(4)))
                #dispatcher.utter_message("Please Enter The Varfication OTP")
                print(value)
                #print("above")
            else:
                dispatcher.utter_message("Please include country code 91")
                slot_values[slot] = None
        else:
            dispatcher.utter_message("Please provide valid mobile number including country code 91")
            slot_values[slot] = None

return [SlotSet(slot, value) for slot, value in slot_values.items()]

def submit(self,dispatcher: CollectingDispatcher,tracker: Tracker,domain: Dict[Text, Any],) -> List[Dict]:
    mob=tracker.get_slot("phno")
    otp=tracker.get_slot("otp")
    """def retryOTP(mob):
        print(otpobj.retry(mob))
        dispatcher.utter_message("OTP successfully Send")"""
    slot_values = self.extract_other_slots(dispatcher, tracker, domain)
    slot_values.update(self.extract_requested_slot(dispatcher, tracker, domain))

    print("strated")
    print(mob)
    verfiyotp = otpobj.verify(mob,otp)
    out0 = str(verfiyotp)
    only_alpha = ""
    for char in out0:
        if ord(char) >= 65 and ord(char) <= 90:
            only_alpha += char
            ## checking for lower case
        elif ord(char) >= 97 and ord(char) <= 122:
            only_alpha += char
            ## printing the string which contains only alphabets
    print(only_alpha)
    str1=only_alpha.replace("message","")
    str2=str1.replace("typesuccess","")
    str3=str2.replace("typeerror","")
    outputdata=str3
    print(outputdata)
    for slot, value in slot_values.items():
        if slot == 'otp':
            if value.isdigit() == True and len(value) == 4:
                #print( tracker.latest_message.get('text')) to take last indent value

```

```
if "otpverified" == outputdata:
    dispatcher.utter_message("Successfully Logged In")
    dispatcher.utter_message("Welcome To The CareerLabs")
    dispatcher.utter_template("utter_ask_start_test",tracker)
if "alreadyverified" == outputdata:
    dispatcher.utter_message("Successfully Log In")
    dispatcher.utter_message("welcome To CareerLabs")
    dispatcher.utter_template("utter_ask_start_test",tracker)
elif "otpnotverified" == outputdata:
    print("otp_not_verified or error")
    slot_values[slot] = None
    dispatcher.utter_template("utter_ask_retry",tracker)

elif "otpoutoflimit" == outputdata:
    print("otp out of limit or error")
    slot_values[slot] = None
    dispatcher.utter_template("utter_ask_retry",tracker)

elif "invalidotp" == outputdata:
    print("invalid otp or error")
    slot_values[slot] = None
    dispatcher.utter_template("utter_ask_retry",tracker)
elif "otpexpired" == outputdata:
    print("otp expired or error")
    slot_values[slot] = None
    dispatcher.utter_template("utter_ask_retry",tracker)
elif "maxlimitreachedforthisotpverification" == outputdata:
    print("maxlimitreachedforthisotpverification or error")
    slot_values[slot] = None
    dispatcher.utter_template("utter_ask_retry",tracker)

else:
    slot_values[slot] = None
    dispatcher.utter_message(text="Please Type 4 Digit Valid OTP!")
    dispatcher.utter_message(text="You Can Choose Resend OTP!",buttons=[{"title": "Resend OTP", "payload": "Resending OTP"}])

return [SlotSet(slot, value) for slot, value in slot_values.items()]
```

Code for OTP:

```
class sendotp:

    def __init__(self, key, msg):

        self.baseUrl = "http://control.msg91.com"
        self.authkey = key

        try:
            msg
        except NameError:
            self.msg = "Your otp is {{otp}}. Please do not share it with anybody"
        else:
            self.msg = msg

    def actionURLBuilder(self, actionurl):
        # print self.baseUrl + '/api/' +str(actionurl)
        #print (actionurl)
        return self.baseUrl + '/api/' + str(actionurl)

    def generateOtp(self):
        return randint(1000, 9999)

    def send(self, contactNumber, senderId, otp):
        values = {
            'authkey': self.authkey,
            'mobile': contactNumber,
            'message': self.msg.replace("{{otp}}", str(otp)),
            'sender': senderId,
            'otp': otp
        }
        print (self.call('sendotp.php', values))
        return otp

    def retry(self, contactNumber, retrytype='text'):
        values = {
            'authkey': self.authkey,
            'mobile': contactNumber,
            'retrytype': retrytype
        }
        print (values)

response = self.call('retryotp.php', values)
```

```
return;

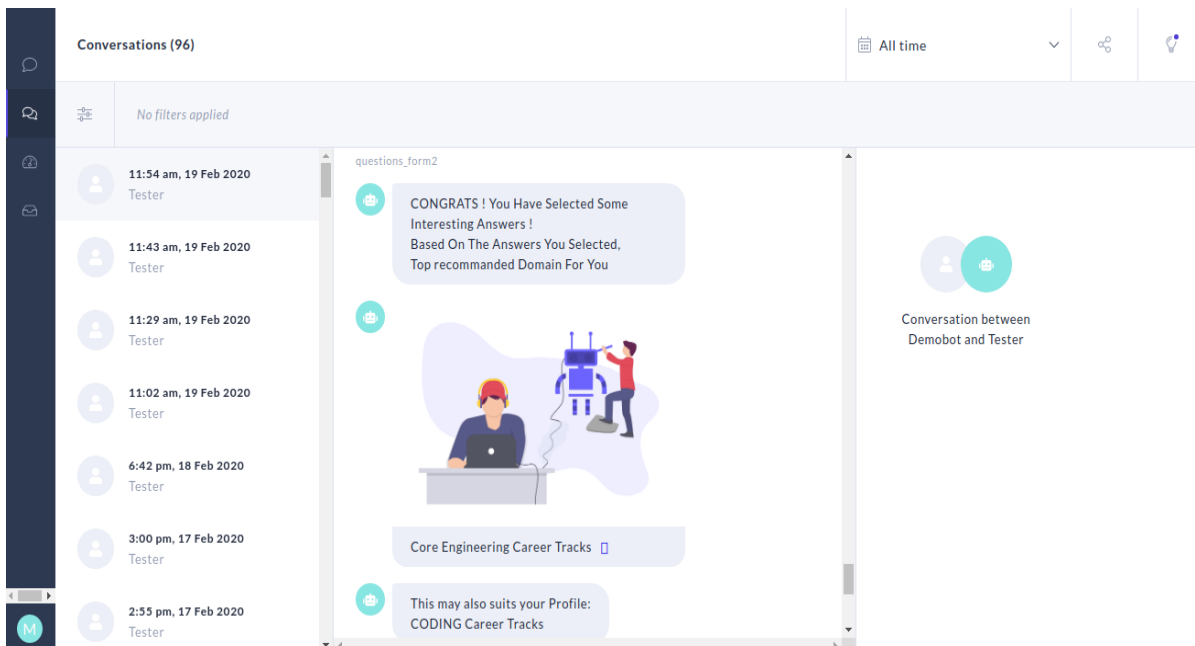
def verify(self, contactNumber, otp):
    values = {
        'authkey': self.authkey,
        'mobile': contactNumber,
        'otp': otp
    }
    response = self.call('verifyRequestOTP.php', values)
    return response

def call(self, actionurl, args):
    url = self.actionURLBuilder(actionurl)
    #print (url)
    payload = (args)

    response = requests.post(url, data=payload, verify=False)
    #print (response.text)
    return response.text#status_code
otpobj=sendotp('293971AJNhj1bUMHcC5e3bd0d9P1','Your OTP code is {{otp}}
keep otp with you. \nFrom CareerLabs PVT LTD.')
```

SOFTWARE TESTING

At RASA Chatbot we can do testing using inbuilt functionality Call as conversion with demo bot and tester. As in following figure:



As in the above figure you can see there is a list of all users who are using or used that chatbot as tester and we can improve chatbot.

Test Case:

- **TEST CASE I**

TEST CASE ID	TEST CASE I
Test Input	Username, password
Expected Result	Login Successful
Actual Result	Login Successful
Status	Successful Pass

- **TEST CASE II**

TEST CASE ID	TEST CASE II
Test Input	Student All Detail(Name, Password, College Name, Branch..etc)
Expected Result	Registration should successful
Actual Result	Registration is successful
Status	Pass

- **TEST CASE III**

TEST CASE ID	TEST CASE III
Test Input	Prediction of the domain(Coding, Tester, Manager.etc.)
Expected Result	One domain name output
Actual Result	One domain name output
Status	Pass

- **TEST CASE IV**

TEST CASE ID	TEST CASE IV
Test Input	Mobile Number
Expected Result	Should get OTP
Actual Result	Getting OTP on Number
Status	Pass

- **TEST CASE V**

TEST CASE ID	TEST CASE V
Test Input	Questions
Expected Result	Bot should ask questions
Actual Result	Bot is asking questions
Status	Pass

- **TEST CASE VI**

TEST CASE ID	TEST CASE VI
Test Input	Unknown words
Expected Result	Bot should response appropriate messages
Actual Result	Bot is responding appropriate messages
Status	Pass

CONCLUSION

A conversational careerguru agent is built as a helper bot in this project, which provides suitable answers to any questions asked by the users who want to know about their career domain . We've succeeded in building chatbot using the libraries since it is the most challenging factor for naïve users and non-professionals. As future work, we can make the chatbot which enables users to interact with helper bot in a more user-friendly way and enhance the discussion by including and changing patterns and templates. We can improve efficiency by training with more complex data sets and other enhancements like voice recognition which requires further training.

FUTURE ENHANCEMENTS

This project presents a career track Chatbot for students. The chatbot building with AI features. Careerguru chatbot can be enhanced and build with voice bot assistance. Future enhancement for careerguru chatbot is, it can book courses and register for webinars. This chatbot can be built with Ai and MI features.

In future, we can build this chatbot fully automated that not required and human interface.

This chatbot will be doing

1. Registration
2. Book online session
3. Buy online courses
4. Fixing a meeting
5. User friendly bot

and so on.

career guru chatbot is working with API and cloud data platform that make them faster and reliable. While talking to chatbot you feel like you are talking with a human.

BIBLIOGRAPHY

Websites:

- <https://rasa.com/>
- https://s3.amazonaws.com/assets.datacamp.com/production/course_3631/slides/chapter2.pdf
- <https://chatbotsmagazine.com/the-best-books-you-need-to-read-to-understand-chat-bot-ecosystem-a72e8a0bd2a>
- <https://medium.com/@itsromiljain/>
<https://www.datasciencelearner.com/guide/>
- <https://msg91.com/>
- <https://www.python.org/>
- <https://stackoverflow.com/>
- <https://www.mongodb.com/>

Books:

- **MongoDB Basics** - Book by David Hows, Eelco Plugge, and Peter Membrey
- **Python Programming: An Introduction to Computer Science**

USER MANUAL

This chatbot is intended for the following audience:

- Students
- Users
- Customers

Chatbot is a man-made artificial intelligence system that we interact over a messaging platform. It helps

Students can answer questions and fulfill an array of tasks.

Students need to be in regular touch with their web portal for various queries and daily updates.

Therefore Careerlabs needed a platform to answer students questions within the communication

mediums their students are already using. By using Careerguru Chatbots, we can alleviate a lot of the complexity for the user. To ensure security before providing any information about the domain etc, this interface authenticates the customer.

Features Supported through Chatbot App:

- Sign In
- Sign Up
- Take a test
- Prediction of Career Path

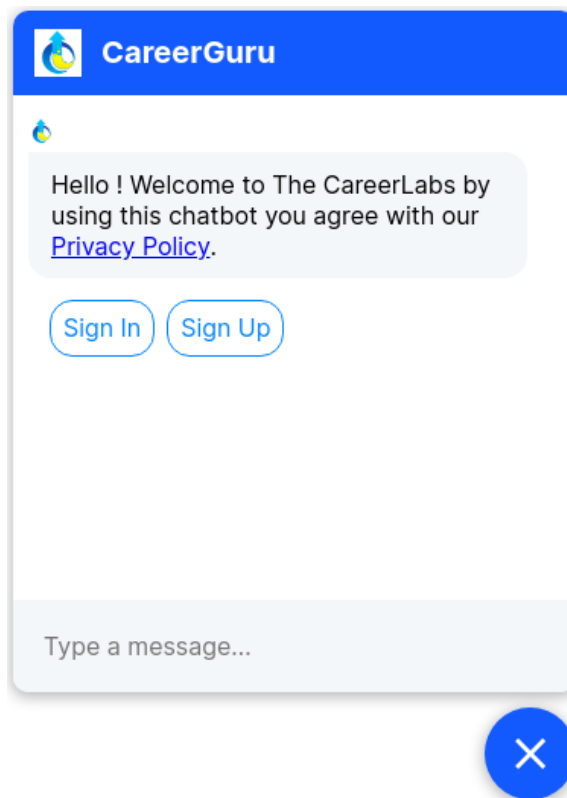
LOG-IN in to Chatbot App

The user requires authentic credentials to log in to the Chatbot App.

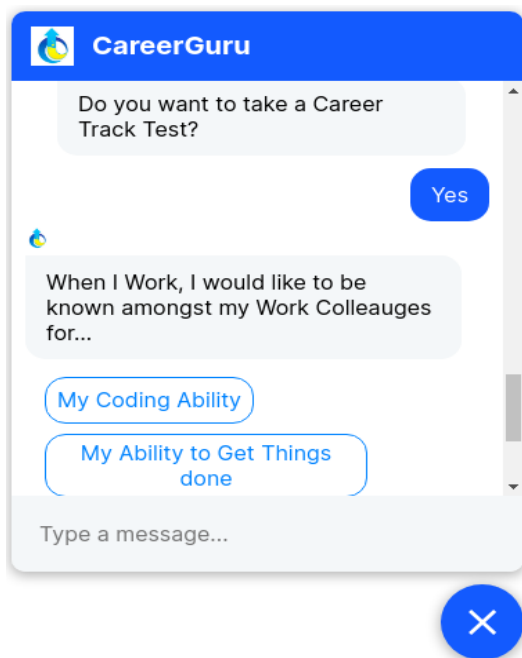
To log in to the application:

1. Open an internet browser
2. Login to Careerlabs with the user credentials.
3. Give answers to the chatbot. It will ask for all necessary information.

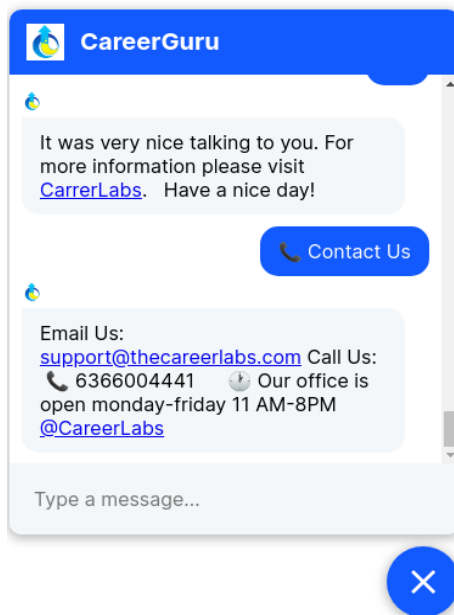
For the primary time login, Chatbot welcome message appears along side the OTP login.



4. Click Yes to Get Start Career domain Test which is available after login or sign up only.



5. Type Help Me or help in the message area and press Enter to interact with the Chatbot. It displays all inquiry that user can do using Chatbot.



LOG-Out from Chatbot App

Using this option users can log-out from Chatbot App.

To log out of the application:

1. In the top right corner of Careerlabs, click and then click the Log Out option.
2. The success message of logging out appears.

Career domain Result from Chatbot App

