

An Group Project Report on

## **BANK-AI-BOT**

Submitted to

The Department of Computer Science and Engineering

In 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

By

BIKUMALLA SAIKIRAN

17311A05Q2

MASHETTI PAVANKALYAN

17311A05W8

CHOWKULA KARTHIK

17311A05U5

Under the Guidance of

Mr. Kandula Damodhar Rao

Associate Professor



Sreenidhi Institute of Science and Technology

Yamnampet, Ghatkesar, R.R. District, Hyderabad - 501301

Affiliated to

Jawaharlal Nehru Technology University

Hyderabad - 500085

Department of Computer Science and Engineering

Sreenidhi Institute of Science and Technology



## CERTIFICATE

This is to certify that this Group Project report on “**BANK-AI-BOT**”, submitted by **BIKUMALLA SAIKIRAN(17311A05Q2)**, **MASHETTI PAVANKALYAN(17311A05W8)** and **CHOWKULA KARTHIK(17311A05U5)** in the year **2019** in 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 part of their Group Project under our guidance. This report has not been submitted to any other institute or university for the award of any degree.

Mr. Kandula Damodhar Rao  
Associate Professor  
Department of CSE  
Internal Guide

Dr . Aruna Varanasi  
Head, Department of CSE

Mr. A.SATYAM  
Assistant Professor  
Department of CSE  
Project Coordinator

External Examiner  
Date:-

## **DECLARATION**

We, **BIKUMALLA SAIKIRAN(17311A05Q2), CHOWKULA KARTHIK(17311A05U5), MASHETTI PAVANKALYAN(17311A05W8)** students of **SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY, YAMNAMPET, GHATKESAR, of COMPUTER SCIENCE AND ENGINEERING** solemnly declare that the Group project work, titled **“BANK-AI-BOT”** 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

STUDENT’S SIGNATURE

BIKUMALLA SAIKIRAN  
(17311A05Q2)

MASHETTI PAVANKALYAN  
(17311A05W8)

CHOWKULA KARTHIK  
(17311A05U5)

## **ACKNOWLEDGMENT**

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 run 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 my internal guide **Mr. Kandula Damodhar Rao** for his technical guidance, constant encouragement and support in carrying out my project 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 thank all the other staff members, both teaching and non-teaching, who have extended their timely help and eased my task.

<b>BIKUMALLA SAIKIRAN</b>	<b>17311A05Q2</b>
<b>MASHETTI PAVANKALYAN</b>	<b>17311A05W8</b>
<b>CHOWKULA KARTHIK</b>	<b>17311A05U5</b>

## List of Figures

S no	Figure no	Figure title	Page no
1	3.1	Architectural Design.	8
2	3.2	Project Architecture	9
3	3.3	Data Flow Diagram	10
4	3.4	Use Case Diagram.	11
5	3.5	Class Diagram.	12
6	3.6	Activity Diagram.	13
7	5.1	Run the Chatbot	28
8	5.2	Rasa X login	28
9	5.3	Basic chat - greet	29
10	5.4	Basic chat - talk	29
11	5.5	Basic chat 3	30
12	5.6	Basic chat 4	30
13	5.7	Basic chat 5	31
14	5.8	Banking chat - DD	31
15	5.9	Banking chat – DD cancel	32
16	5.10	Banking chat - loans	32
17	5.11	Banking chat - accounts	33
18	5.12	Banking chat – banking cards	33

# **BANK-AI-BOT**

## **Abstract**

The BANK-AI-BOT project is built using artificial algorithms that analyses user's queries and understand user's message. This System is a application which provides answer to the query of the user. Users just have to query through the bot which is used for chatting. Users can chat using any format there is no specific format the user has to follow. The System uses built in artificial intelligence to answer the query. The answers are appropriate what the user queries. The User can query any bank related activities through the system. The user does not have to personally go to the bank for enquiry. The System analyses the question and then answers to the user. The system answers to the query as if it is answered by the person. With the help of artificial intelligence, the system answers the query asked by the users. Various helping pages has the bot through which the user can chat by asking queries related to banking activities. The System analyses the question and then answers to the user. The system answers to the query as if it is answered by the person. With the help of artificial intelligence, the system answers the query asked by the users. The system replies to the user with the help of effective graphical user interface. The user can query about the bank related activities with the help of this application. The user can query banking related activities such as procedures of debiting, crediting or taking a demand draft or a loan etc. This system helps the user to be updated about the bank related activities.

**Key words :** Artificial Intelligence , bank enquiry , query , algorithms.

# INDEX

<b>List of Figures</b>	i
<b>Abstract</b>	ii
<b>1.INTRODUCTION</b>	1
1.1 Scope	2
1.2 Literature Survey	2
1.3 Proposed System	2
<b>2.SYSTEM ANALYSIS</b>	4
2.1 Functional Requirement Specifications	4
2.2 Performance Requirements	5
2.3 Software Requirements	6
2.4 Hardware Requirements	7
<b>3.SYSTEM DESIGN</b>	8
3.1 Architectural Design	8
3.2 UML Diagrams	10
3.2.1 Data Flow Diagram	10
3.2.2 Use Case Diagram	11
3.2.3 Class Diagram	12
3.2.4 Activity Diagram	13
3.3 Modules	14

<b>4.SYSTEM IMPLEMENTATION</b>	<b>15</b>
4.1 Code	15
4.2 Tools and Technology	27
<b>5. RESLULTS AND OUTPUT SCREENS</b>	<b>28</b>
<b>6.CONCLUSION AND FUTURE SCOPE</b>	<b>34</b>
<b>BIBLIOGRAPHY</b>	<b>35</b>
<b>APPENDIX-A : SKELETON OF RASA</b>	<b>36</b>
<b>APPENDIX-B : RASA X</b>	<b>41</b>
<b>APPENDIX-C : UNDERSTANDING THE CONVERSATIONAL CHATBOT ARCHITECTURE</b>	<b>45</b>



# **1. INTRODUCTION**

The User can query any bank related activities through the system. The user does not have to personally go to the bank for enquiry. The System analyses the question and then answers to the user. The system answers to the query as if it is answered by the person. With the help of artificial intelligence, the system answers the query asked by the users. Various helping pages has the bot through which the user can chat by asking queries related to banking activities.

The system replies to the user with the help of effective graphical user interface. The user can query about the bank related activities with the help of this application. The user can query banking related activities such as procedures of debiting, crediting or taking a demand draft or a loan etc. This system helps the user to be updated about the bank related activities.

Human language is special for several reasons. Understanding human language is considered a difficult task due to its complexity, ambiguity, variability and unexpected input like typos, grammar, spelling mistakes and faulty pronunciations can make natural language understanding and processing even more difficult. Chatbots can be used in many different ways, which is the reason why it's difficult to define exactly what they are.

It is actually possible to come up with a chatbot use case for every single business or industry, in the same way, that every business or industry can use a website or app. Literally any task you can do delegate to Chatbot. A couple of things you can definitely do is training chatbot to handle FAQs, Basic flow handling where it can help the user to view, purchase, or track something, feedback collection or notification. chatbots could allow human agents to get involved in the conversation if necessary, perhaps as a premium service.

## **1.1 Scope**

The objective of the paper is to successfully build an conversational AI (Artificial Intelligence) Chatbot which is used to query all the bank related enquiries. The engine developed must deal with different types of messages or queries the user gives and respond to them accurately. For each approach the engine is to be developed in such a way that it classifies the scenario of the message and analyze its semantics , give a reply. Commercial product survey is beyond scope of this paper as sometimes these products claim more accuracy than actual for promotional purposes. The scope of the system may vary from situation to situation and time to time.

## **1.2 Literature Survey**

Customers interact with their banks today through multiple channels. Branches, ATM, telephone banking, internet banking, and m-banking are all efficient ways of selling products and services to banking customers. The evolution from a focus on local-centric (branches and ATM) to place-centric (internet banking) and then to equipment-centric (accessible anywhere, 24 hours per day and 7 days a week) has yielded benefits in the form of time savings and shorter customer queues. The idea of this project is to make the customer satisfiable even when the servers are down by making an AI embedded chatbot.

## **1.3 Proposed System**

This System is a application which provides answer to the query of the user. Users just have to query through the bot which is used for chatting. Users can chat using any format there is no specific format the user has to follow. The System uses built in artificial intelligence to answer the query. The answers are appropriate what the user queries. The User can query any bank related activities through the system. The idea of this project is to make the customer satisfiable

even when the servers are down by making an AI embedded chatbot without any human interference.

The proposed system uses RASA frame work to create the chatbot.

[Rasa](#) is an open-source machine learning framework for building contextual AI assistants and chatbots. To make complete AI-based chatbot it has to handle two things:

- Understanding the user's message which is done through Rasa [NLU](#).
- The bot should be able to carry dialogue with context which is done by Rasa [Core](#).

As planned before we need to be very thorough with the scope of the bot. Hence we need to define its own universe (domain) in which our bot operates. It has the intents which it should classify to, entities which it should extract, slots which it should remember to maintain context, and actions which it should perform to complete the task. And response templates which bot should utter back (templates).

The basic workflow of our proposed system is:

Understand the chatbot's conversation flow again and create an NLU and story training data based on that. Assume that you are the first user who is talking to the bot and thinks of all-natural and quirky conversation you can have :P and prepare NLU and stories training data. Don't forget to include small talk(greetings, deny etc) in the training data.

Keep the all intents mutually exclusive and diversity in its utterance.

Set appropriate NLU pipeline and policy configuration. For NLU we will use the English language and default spacy pipeline (pretrained\_embeddings\_spacy). For Core, we will use MemoizationPolicy and KerasPolicy. As it's ML-based bot.

Explore other ways to build the bot. Write a domain.yml file. Write an action.py file - action.py file is self-explanatory as all classes and functions are well commented. Write an endpoint.yml file - this file contains the different endpoints which your bot can connect to. Like tracker store, events, actions, etc.

## **2. SYSTEM ANALYSIS**

This System Analysis is closely related to requirements analysis. It is also an explicit formal inquiry carried out to help someone (referred to as the decision maker) identify a better course of action and make a better decision than he might otherwise have made. This step involves breaking down the system in different pieces to analyze the situation, analyzing project goals, breaking down what needs to be created and attempting to engage users so that definite requirements can be defined.

### **2.1 Functional Requirement Specification**

The System after careful analysis has been identified to be present with the following modules.

#### **1. Development module**

This module deals with development of chatbot using RASA framework. This module includes development using two sub-modules: RASA CORE and RASA NLU.

##### **1.1. RASA Core module:**

core module deals with the collections of different pathways of the chatbot to create a response template action pair. It also deals with the domains, responses, actions and policies that the chatbot has to follow.

##### **1.2 RASA NLU module:**

This module deals with the brain of chatbot. Brain in the sense all the computing informational responses and replies are taken care by this module. It has components such as Tokenization, Featurizer, Entity extraction, classifier.

#### **2. Training module**

This module includes creation of training data for chatbot, creation of sample responses for every intent and reply of each action. This module deals with analyzing the activities of the chatbot and creating the training data according to the functional requirements. This module even uses the help of development module (NLU) to create training data

### **3. Testing module**

This module deals with the testing of the bot and even correction of the error responses. This module includes a special browser application which is used to see, test, talk evaluate and even correct the responses of the bot. The responses can even be classified for each intent and action manually and trained. This module involves training and even testing.

### **4. Deployment module**

This module deals with deployment of the chatbot engine to any application. This module uses rasa X an inbuilt deployment engine which is graphically designed web browser to test, train and even analyze the chatbot data. Rasa X is a tool that helps you build, improve, and deploy AI Assistants that are powered by the Rasa framework. Rasa X runs on your own computer and you can deploy it to your own server. None of your conversations or training data are ever sent anywhere.

## **2.2 Performance Requirements**

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely with the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

- The system should be able to interface with the existing system
- The system should be accurate

The existing system is completely dependent on the user to perform all the duties.

## 2.3 Software Requirements:

- **Operating System:** Microsoft Windows / Linux / MacOS
- **Technology:** Artificial Intelligence
- **Tools :** RASA , RASA x
- **Platform:** RASA engine, Anaconda distribution

To run a software it should satisfy minimum system requirements. For accomplishment of this project we use ANACONDA . **ANACONDA DISTRIBUTION** is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system [conda](#). The Anaconda distribution is used by over 15 million users and includes more than 1500 popular data-science packages suitable for Windows, Linux, and MacOS.

## RASA

Rasa is an open source machine learning framework for automated text and voice-based conversations. Understand messages, hold conversations, and connect to messaging channels and APIs. Understanding the user's message which is done through Rasa [NLU](#). The bot should be able to carry dialogue with context which is done by Rasa [Core](#).

## RASA X

Rasa X is a tool that helps you build, improve, and deploy AI Assistants that are powered by the Rasa framework. Rasa X runs on your own computer and you can deploy it to your own server. None of your conversations or training data are ever sent anywhere. Some things, like inspecting and annotating conversations, are much easier with a UI. Rasa X focuses on those use cases and not on replacing things that are easier to do in code.

## 2.4 Hardware Requirements:

### **Processor:**

**Minimum:** Any Intel or AMD x86-64 processor

**Recommended:** Any Intel or AMD x86-64 processor with four logical cores and AVX2 instruction set support

### **Disk:**

**Minimum:** 3 GB of HDD space for ANACONDA DISTRIBUTION

**Recommended:** An SSD is recommended

### **RAM :**

**Minimum:** 4 GB

**Recommended:** 8 GB

### **Graphics:**

No specific graphics card is required. GPU acceleration can make the model building fast.

### 3. SYSTEM DESIGN

Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. Object-oriented analysis and design methods are becoming the most widely used methods for computer systems design.

#### 3.1 Architectural Design

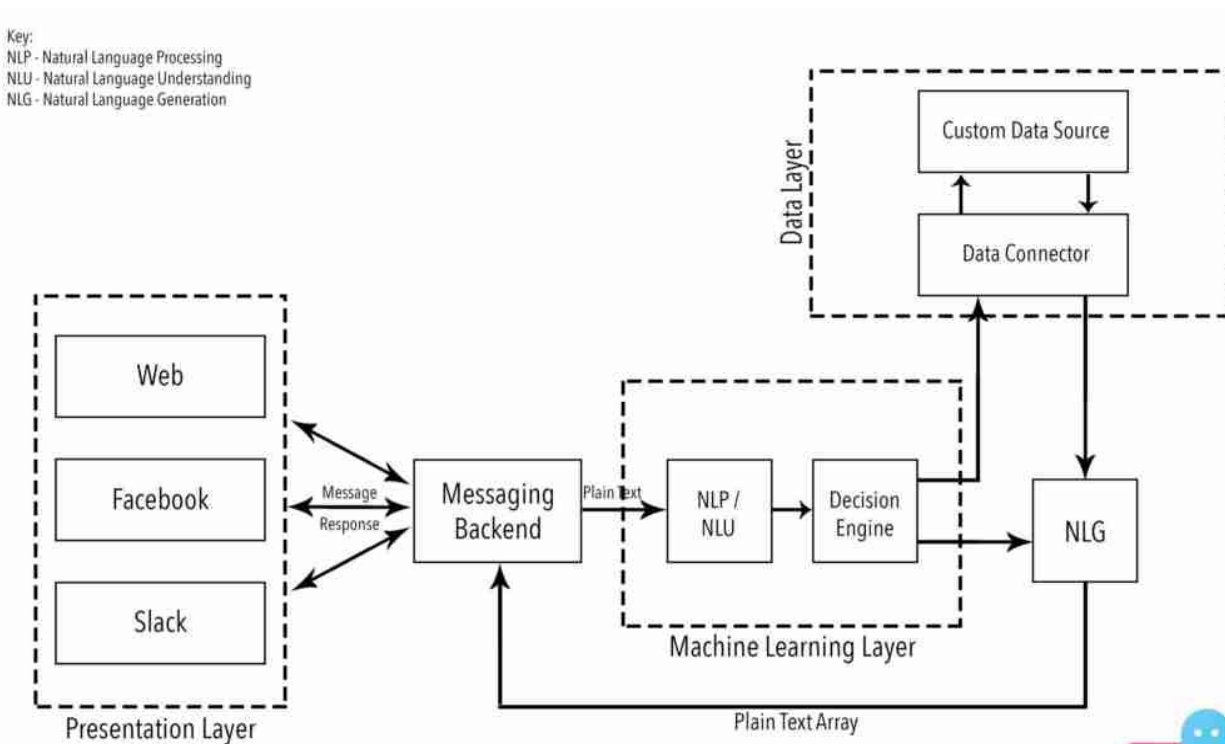


Fig 3.1 Architectural Design

Here NLU means Natural Language understanding

NLP means Natural Language Processing

NLG means Natural Language Generation



Graphical User interface is present at the presentation layer which may be connected to Web , Slack, Facebook etc. Messaging backend receives the message from presentation layer and the plain text is then transferred to the NLP unit or NLU unit which transforms the text and analyzes it and the decision engine gives decision by checking with database( custom data source, data connector) which is then transferred to NLG. The generated response is sent as plain text array to the the messaging backend. Which provides the response to user on GUI

A better architecture of conversational AI chatbot is as follows:

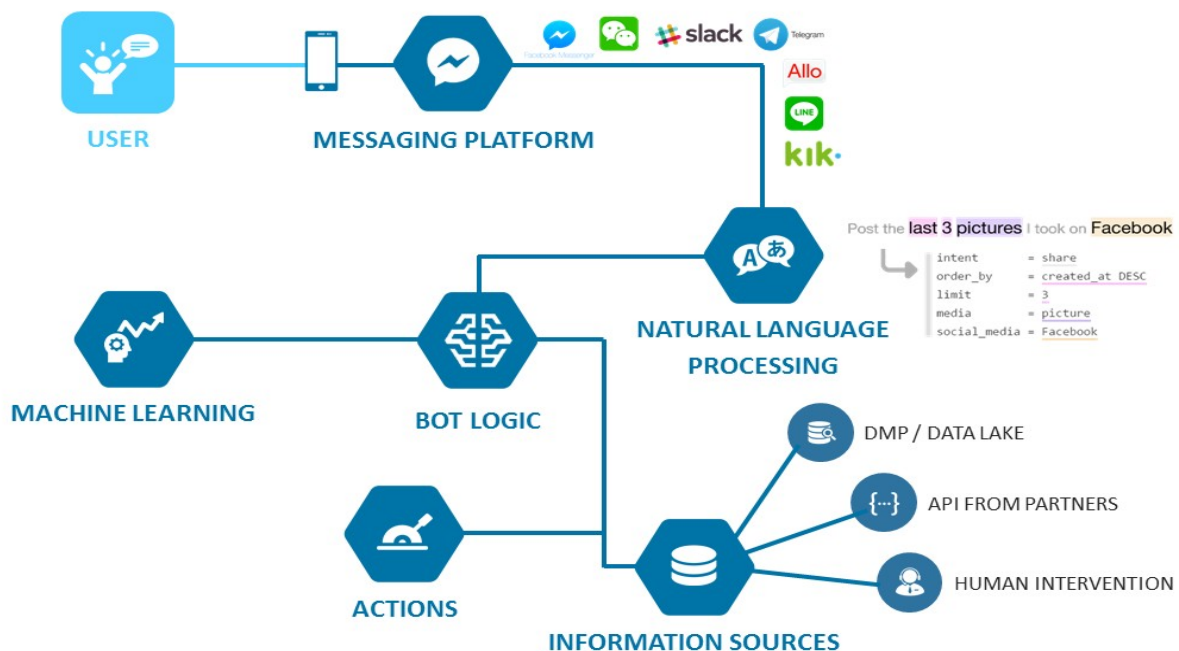


Fig 3.2 Project architecture

The bot logic here consists of data with its grouped classes/ clusters which are used to give response to the user. Bot logic infers with machine learning approach and uses natural language processing, natural language understanding and natural language generation. Actions

## 3.2 UML Diagrams

UML Diagrams for our application are as follows:

### 3.2.1 Data Flow Diagram

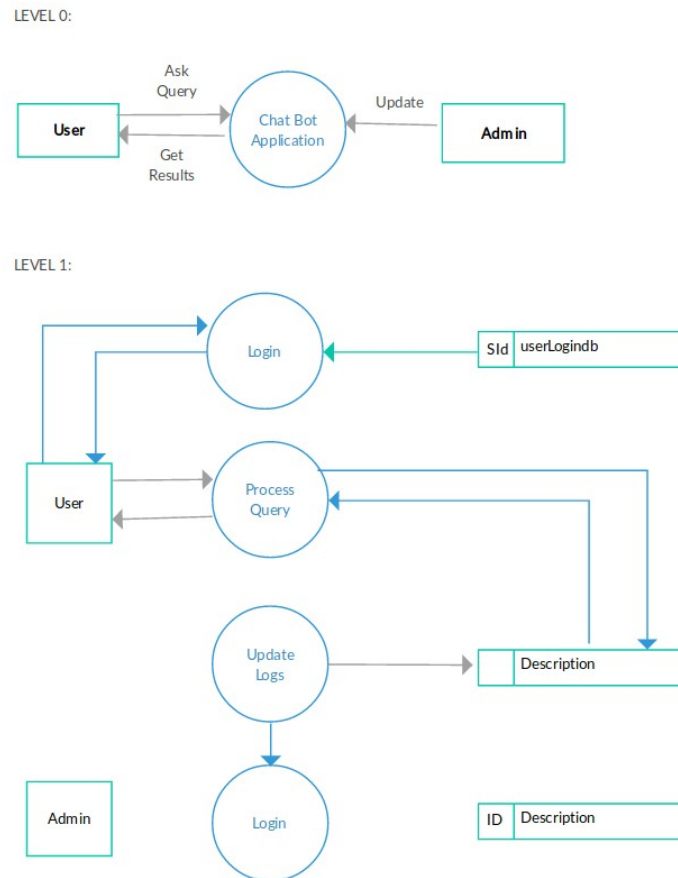


Fig 3.3 Data flow diagram

A **data-flow diagram** (DFD) is a way of representing a **flow** of a **data** of a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself. The above diagram is used to represent the different levels of data flow in our project. User has a login id and password and user enters a query and query is processed and description is provided. The logs are updated and again the admin can analyze all the activities and logs.

### 3.2.2 Use Case Diagram

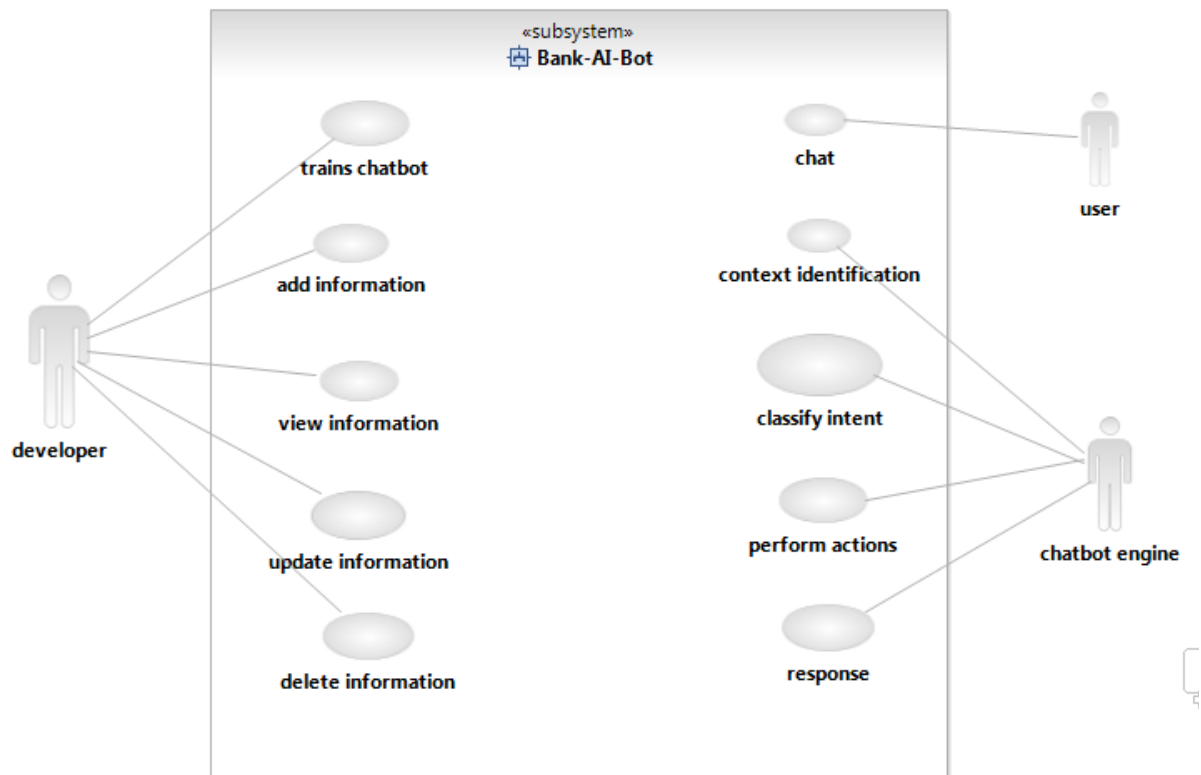


Fig 3.4 Use Case Diagram

In this context diagram, the information provided to and received from the 'Number Plate Recognition System' is identified. The arrows represent the information received or generated by the application. The closed boxes represent the set of sources and sinks of information.

In the system, we can observe that the user interacts with the application through a graphical user interface. The inputs to the system are the queries or messages that are entered by the user. The chatbot engine identifies the context of the text, classifies the intent, performs actions and gives response. Developer develops the chatbot engine by training chatbot with adding training data, removing data , updating data etc.

### USER:

Users just have to query through the bot which is used for chatting. Users can chat using any format there is no specific format the user has to follow.

### DEVELOPER:

Here the developer is the one who creates the chatbot engine, trains it and maintains it. Developer can analyze the data, correct the error responses, add train data, delete train data and also view data. Developer has every access to the application.

### 3.2.3 Class Diagram

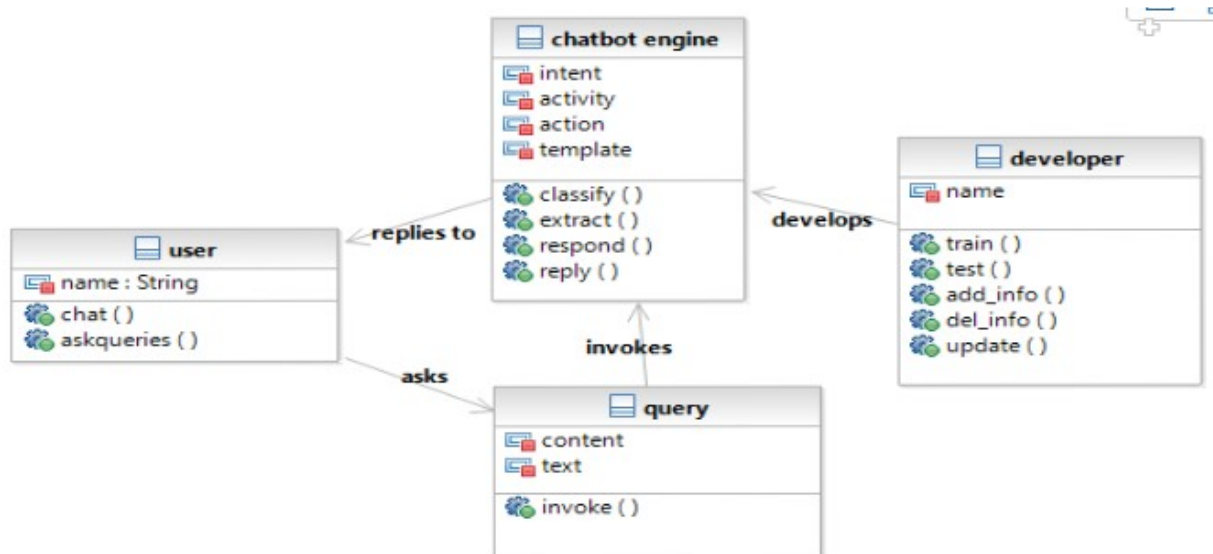


Fig 3.5 Class Diagram

**class diagram** in the Unified Modeling Language (UML) is a type of static structure **diagram** that describes the structure of a system by showing the system's **classes**, their attributes, operations (or methods), and the relationships among objects. We have identified the basic classes of our project as above.

### 3.2.4 Activity Diagram

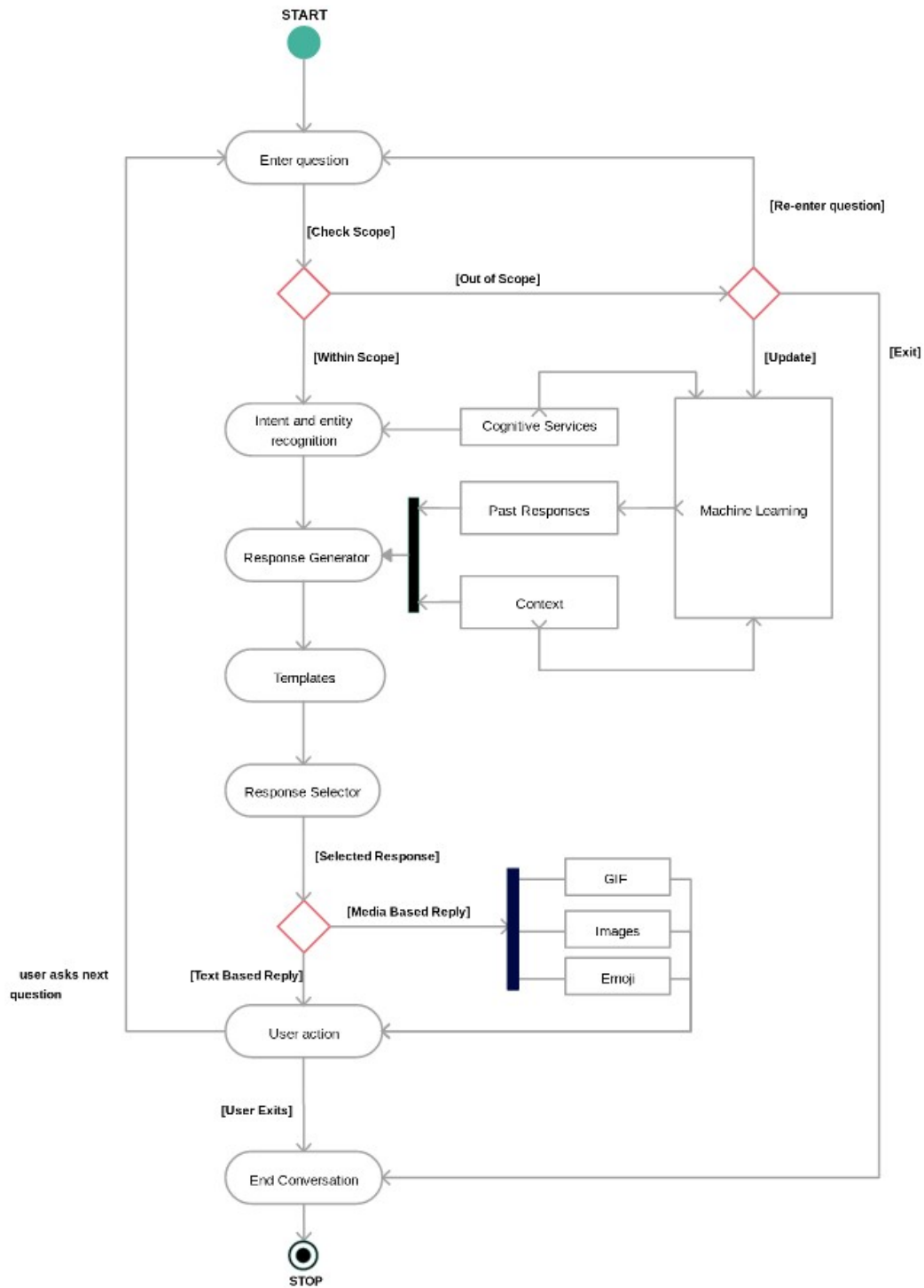


Fig 3.6 Activity Diagram

### **3.3 Modules:**

#### **1. Development module**

This module deals with development of chatbot using RASA framework. This module includes development using two sub-modules: RASA CORE and RASA NLU.

##### **1.1. RASA Core module:**

core module deals with the collections of different pathways of the chatbot to create a response template action pair. It also deals with the domains, responses, actions and policies that the chatbot has to follow.

##### **1.2 RASA NLU module:**

This module deals with the brain of chatbot. Brain in the sense all the computing informational responses and replies are taken care by this module. It has components such as Tokenization, Featurizer, Entity extraction, classifier.

#### **2. Training module**

This module includes creation of training data for chatbot, creation of sample responses for every intent and reply of each action. This module deals with analyzing the activities of the chatbot and creating the training data according to the functional requirements. This module even uses the help of development module (NLU) to create training data

#### **3. Testing module**

This module deals with the testing of the bot and even correction of the error responses. This module includes a special browser application which is used to see, test, talk evaluate the responses. The responses can even be classified for each intent and action manually and trained.

#### **4. Deployment module**

This deals with deployment of the chatbot engine to any application. This module uses rasa X an inbuilt deployment engine which is graphically designed web browser to test, train and even analyze the chatbot data. Rasa X is a tool that helps you build, improve, and deploy AI Assistants that are powered by the Rasa framework. Rasa X runs on your own computer and you can deploy it to your own server. None of your conversations or training data are ever sent anywhere.

## 4. SYSTEM IMPLEMENTATION

### 4.1 CODE:

The **rasa init** creates a RASA project which is valid. A valid RASA project will have:

__init__.py	an empty file that helps python find your actions
actions.py	code for your custom actions
config.yml *	configuration of your NLU and Core models
credentials.yml	details for connecting to other services
data/nlu.md *	your NLU training data
data/stories.md *	your stories
domain.yml *	your assistant's domain
endpoints.yml	details for connecting to channels like fb messenger
models/<timestamp>.tar.gz	your initial model

the most important files are marked with (\*) .

#### 4.1.1 config.yml :

# Configuration for Rasa NLU.

# <https://rasa.com/docs/rasa/nlu/components/>

language: en

pipeline: supervised\_embeddings

# Configuration for Rasa Core.

policies:

- name: MemoizationPolicy
- name: KerasPolicy
- name: MappingPolicy

### 4.1.2 credentials.yml :

# This file contains the credentials for the voice & chat platforms which your bot is using.

# <https://rasa.com/docs/rasa/user-guide/messaging-and-voice-channels/>

#### **rest:**

# you don't need to provide anything here - this channel doesn't

# require any credentials

#### **facebook:**

verify: "bank-ai-bot"

secret: "<< access key >>"

page-access-token: "<<page access token>>"

#### **slack:**

slack\_token: "<your slack token>"

slack\_channel: "<the slack channel>"

#### **socketio:**

user\_message\_evt: <event name for user message>

bot\_message\_evt: <event name for but messages>

session\_persistence: <true/false>

#### **rasa:**

url: "<http://localhost:5002/api>"

**NOTE : enable only one platform from the above ( facebook, slack, socketio)**



### 4.1.3 domain.yml :

This file contains all the intents , templates and actions that are required by the chatbot engine. This file in our project consists of more than thousand number of lines and hence only a sample code of this file is displayed.

#### **intents:**

- DD\_offline
- DD\_online
- DD\_charge
- DD\_cancel
- DD\_expiry
- loan\_types
- acct\_types
- current\_acct
- savings\_acct

#### **entities:**

- number
- feedback\_value

#### **templates:**

##### **utter\_technical\_question:**

- text: I can't answer that.

##### **utter\_default:**

- text: Sorry, I didn't get that . Could you please rephrase?

- text: I didn't understand, could you rephrase that?
- text: I'm sorry, but I didn't understand you. Could you please rephrase what you just said?
- text: I'm afraid I didn't get what you just said. Could you rephrase that?
- text: I didn't quite get that, could you rephrase your message?
- text: Could you rephrase your message? I didn't get it, I'm sorry.

**utter\_diff\_DD:**

- text: offline at bank , and online

**utter\_DD\_offline:**

- text: visit any bank and take the application form and fill up the details and credentials

**utter\_DD\_online:**

- text: this feature is only applicable through internet banking. It takes 2-5 days to courier the D

**utter\_DD\_charge:**

- text: yes, it depends on the bank and is mostly 0.5% of money

**utter\_DD\_cancel:**

- text: yes, If you got the DD by cash deposit, you need to submit original DD as well as receipt of cash payment. If paid from account then amount will be credited directly. Amount will be refunded with some deduction of amount (around Rs 100-150).

**utter\_DD\_expiry:**

- text: any demand draft is valid only upto 3 months from date issued

**utter\_loan\_types:**

- text: Unsecured personal loans Secured personal loans Payday loans Title loans Pawn shop loans Payday alternative loans Home equity loans Credit card cash

advances

**utter\_acct\_types:**

- text: 1) Current Account. 2) Savings Account. 3) Recurring Deposit Account. 4)

Fixed Deposit Account.

**utter\_current\_acct:**

- text: Current bank account is opened by businessmen who have a higher number of regular transactions with the bank.

**utter\_savings\_acct:**

- text: A savings account is a basic type of bank account that allows you to deposit money, keep it safe, and withdraw funds, all while earning interest

**utter\_fixed\_deposit\_acct:**

- text: its a type of savings/investment account that promises the investor a fixed rate of interest and it is paid only at end of investment period

**utter\_bank\_types:**

- text: 1. commercial banks 2. exchange banks 3. industrial banks 4. agricultural or co-operative banks 5. savings banks 6. central banks 7. union banks

**utter\_open\_acct:**

- text: your personal details along with any two identification proofs and residential proof and a photograph

**utter\_debit\_card:**

- text: you need to have a bank account and fill debit card application with account details.

**utter\_debit\_card\_types:**

- text: 1. Mastercard 2. Visa 3. VisaElectron 4. Rupay 5. Maestro

**utter\_rupay:**

- text: it works only in india and hence performs faster transactions comparatively

**utter\_master\_card:**

- text: mastercard transactions are accepted every where around the world.

**utter\_card\_lost:**

- text: immediately approach the bank and proceed further to block it

**utter\_max\_ATM:**

- text: You can withdraw a minimum of Rs 500 and maximum of Rs 10,000 in a single transaction and Rs 20,000 in a day from an SBI ATM.

**utter\_credit\_apply:**

- text: you must be at least 18 to apply for a credit card. If you are under 21, to get approved for credit card, you must provide proof of independent income and assets

**utter\_change\_PIN:**

- text: yes, provided the OTP to registered mobile number

**actions:**

- utter\_diff\_DD
- utter\_DD\_offline
- utter\_DD\_online
- utter\_DD\_charge

- utter\_DD\_cancel
- utter\_DD\_expiry
- utter\_loan\_types
- utter\_acct\_types
- utter\_current\_acct
- utter\_savings\_acct
- utter\_fixed\_deposit\_acct
- utter\_bank\_types
- utter\_open\_acct
- utter\_age\_banking
- utter\_pancard
- utter\_debit\_card
- utter\_debit\_card\_types
- utter\_rupay
- utter\_master\_card
- utter\_card\_lost
- utter\_max\_ATM
- utter\_credit\_apply
- utter\_change\_PIN
- utter\_min\_balance

#### **4.1.4 data/nlu.md**

This file consists of training data which is used to build model. This file in our project consists of more than thousand number of lines and hence only a sample code of this file is displayed.

##### **## intent:DD\_cancel**

- can i cancel the DD
- can i cancel my dd

##### **## intent:DD\_charge**

- is there any charge for making DD?
- is there any charge for making dd

##### **## intent:DD\_expiry**

- is there any expiry date for demand draft?
- is there any expiry date for dd

##### **## intent:DD\_offline**

- how to take demand draft offline?
- how to take dd offline

##### **## intent:DD\_online**

- how to take demand draft online?
- how to take dd online

##### **## intent:acct\_types**

- what are the diff types of bank accounts
- what are the different types of bank accounts

##### **## intent:age\_banking**

- age eligible for banking?
- what is the eligibility age for banking?

### **## intent:ask\_builder**

- can you share your boss with me?
- i want to get to know your owner

### **## intent:ask\_howbuilt**

- how were you built?
- how were you built.

### **## intent:ask\_howdoing**

- Ahoy matey how are you?
- are you alright

### **## intent:ask\_isbot**

- are you a bot?
- are you a real bot?

### **## intent:ask\_languagesbot**

- Which languages do you speak?
- What are the languages you can speak?

### **## intent:ask\_whatspossible**

- how can you help me

### **## intent:bank\_types**

- what are the different types of banks
- what are different types of banks

### **## intent:card\_lost**

- what to do if my card is lost?
- what should i do if my card is lost?

### **## intent:change\_PIN**

- can i change my card pin at ATM
- can i change pin at atm

### **## intent:credit\_apply**

- when can i apply for credit card
- when can i apply for a credit card

### **## intent:current\_acct**

- what is current account
- what is a current account

### **## intent:debit\_card**

- how to get a debit card
- how to take debit card

### **## intent:debit\_card\_types**

- types of debit card?
- what are the types of debit card

### **## intent:diff\_DD**

- what are the different ways to make a demand draft
- what are the different ways to make a dd?

### **## intent:fixed\_deposit\_acct**



#### 4.1.5 data / stories.md :

##### ## New Story

- \* greet
  - utter\_greet
- \* diff\_DD
  - utter\_diff\_DD
- \* DD\_online
  - utter\_DD\_online
- \* DD\_charge
  - utter\_DD\_charge
- \* DD\_cancel
  - utter\_DD\_cancel
- \* DD\_expiry
  - utter\_DD\_expiry

##### ## bank story

- \* bank\_types
  - utter\_bank\_types
- \* open\_acct
  - utter\_open\_acct
- \* age\_banking
  - utter\_age\_banking
- \* pancard
  - utter\_pancard
- \* debit\_card
  - utter\_debit\_card
- \* debit\_card\_types
  - utter\_debit\_card\_types

#### 4.1.6 actions.py :

This file contains all the custom actions and even the API calls related code. This code which is listed below just gives a basic idea of how the code would be there. To get the total code visit <https://github.com/bikumalla-saikiran>.

```
# This files contains your custom actions which can be used to run custom Python code.  
# See this guide on how to implement these action:  
# https://rasa.com/docs/rasa/core/actions/#custom-actions/  
# This is a simple example for a custom action which utters "Hello World!"
```

```
from typing import Any, Text, Dict, List  
from rasa_sdk import Action, Tracker  
from rasa_sdk.executor import CollectingDispatcher  
class ActionHelloWorld(Action):  
    def name(self) -> Text:  
        return "action_hello_world"  
    def run(self, dispatcher: CollectingDispatcher,  
            tracker: Tracker,  
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:  
        dispatcher.utter_message("Hello World!")  
        return []
```

## 4.2 TOOLS AND TECHNOLOGY:

Rasa is an open-source machine learning framework for building [contextual AI assistants and chatbots](#). To make complete AI-based chatbot it has to handle two things: Understanding the user's message which is done through Rasa [NLU](#). The bot should be able to carry dialogue with context which is done by Rasa [Core](#).

Core module deals with the collections of different pathways of the chatbot to create a response template action pair. It also deals with the domains, responses, actions and policies that the chatbot has to follow. NLU module deals with the brain of chatbot. Brain in the sense all the computing informational responses and replies are taken care by this module. It has components such as Tokenization, Featurizer, Entity extraction, classifier.

A chatbot is an artificial intelligence (AI) software that can simulate a conversation (or a chat) with a user in natural language through messaging applications, websites, mobile apps or through the telephone. Why are chatbots important? A chatbot is often described as one of the most advanced and promising expressions of interaction between humans and machines. However, from a technological point of view, a chatbot only represents the natural evolution of a Question Answering system leveraging Natural Language Processing (NLP). Formulating responses to questions in natural language is one of the most typical [Examples of Natural Language Processing](#) applied in various enterprises' end-use applications.

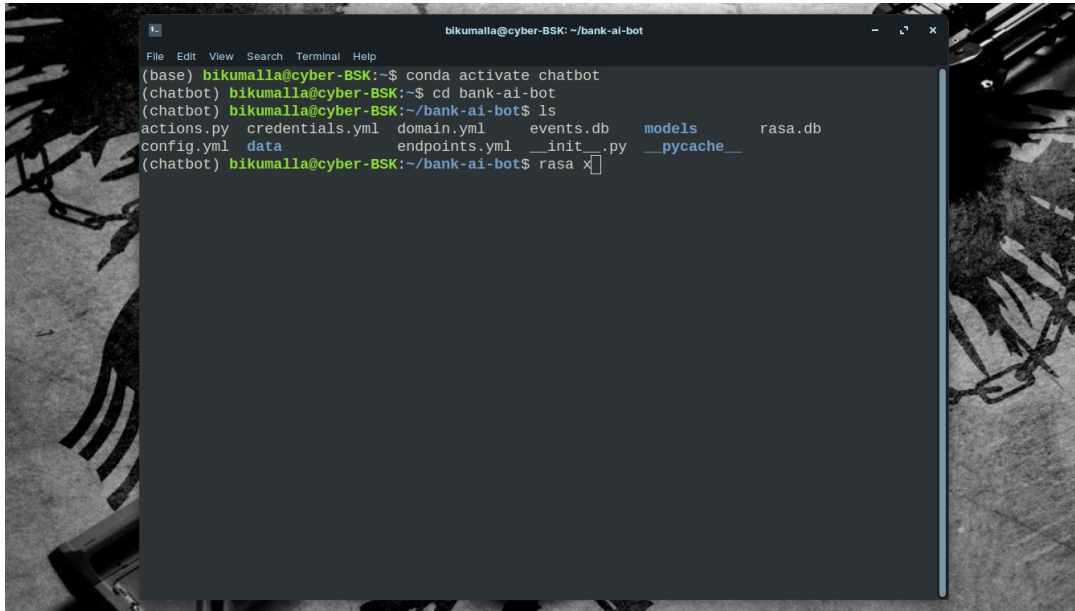
**User request analysis:** this is the first task that a chatbot performs. It analyzes the user's request to identify the user intent and to extract relevant entities.

**Returning the response:** once the user's intent has been identified, the chatbot must provide the most appropriate response for the user's request. The answer may be:

- a generic and predefined text
- a text retrieved from a knowledge base that contains different answers
- a contextualized piece of information based on data the user has provided

## 5. RESULTS AND OUTPUT SCREENS

### RUN THE CHATBOT ENGINE:



```
bikumalla@cyber-BSK: ~/bank-ai-bot
File Edit View Search Terminal Help
(base) bikumalla@cyber-BSK:~$ conda activate chatbot
(chatbot) bikumalla@cyber-BSK:~$ cd bank-ai-bot
(chatbot) bikumalla@cyber-BSK:~/bank-ai-bot$ ls
actions.py  credentials.yml  domain.yml  events.db  models  rasa.db
config.yml  data  endpoints.yml  __init__.py  __pycache__
(chatbot) bikumalla@cyber-BSK:~/bank-ai-bot$ rasa x
```

Fig 5.1 run the chatbot

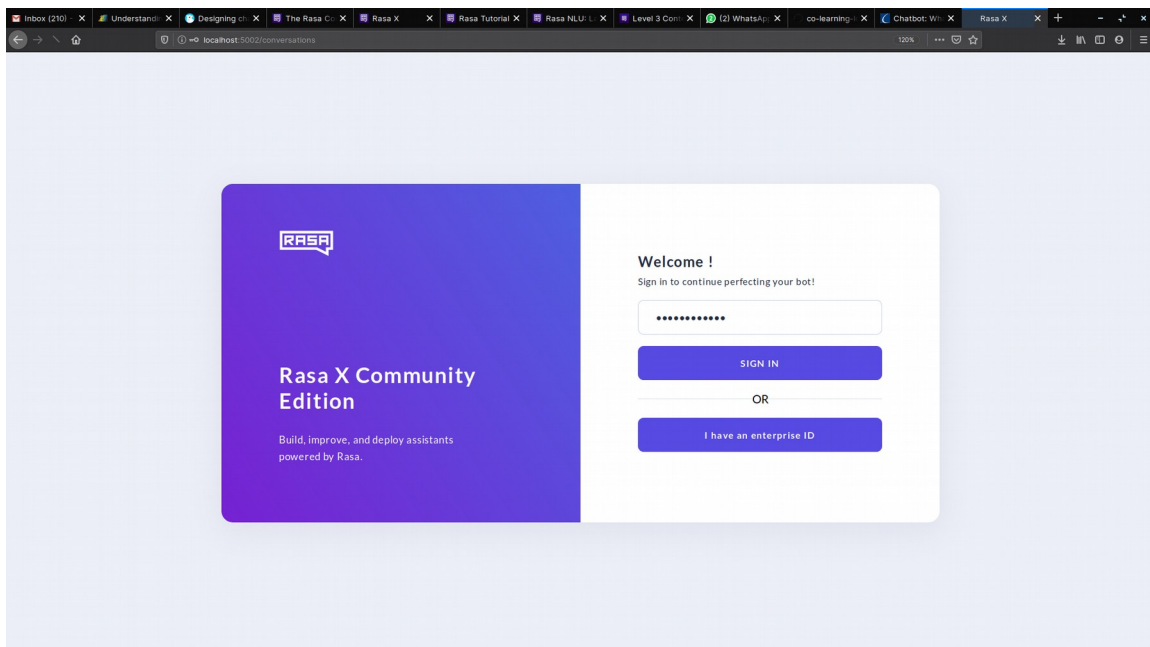


Fig 5.2 RASA X login

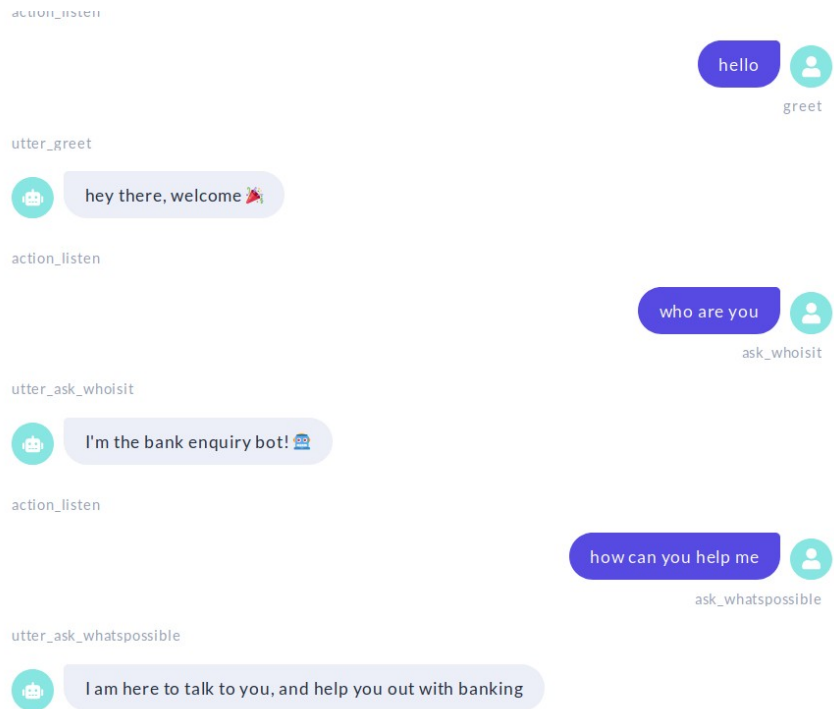


Fig 5.3 basic chat - greet

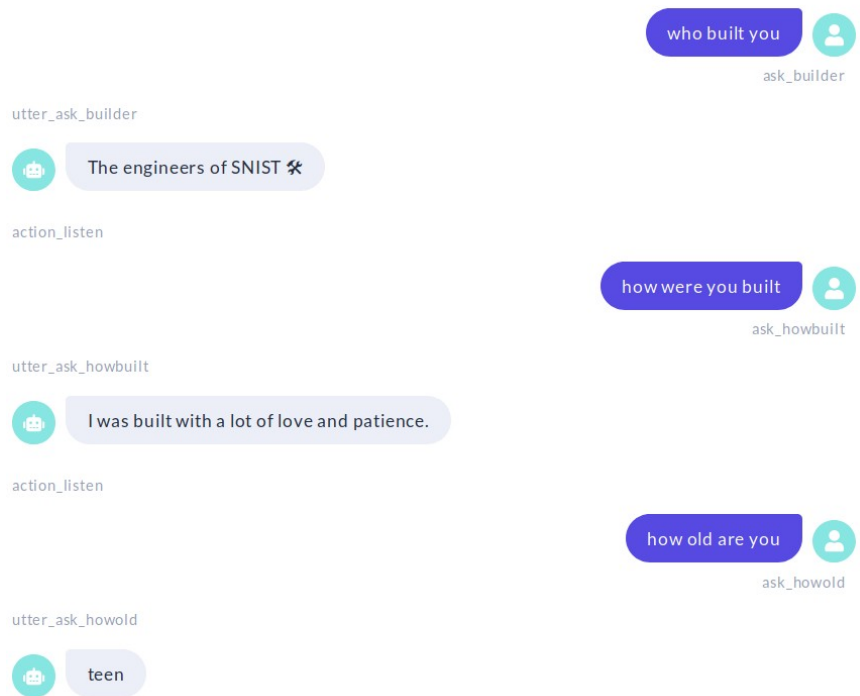


Fig 5.4 basic chat - talk

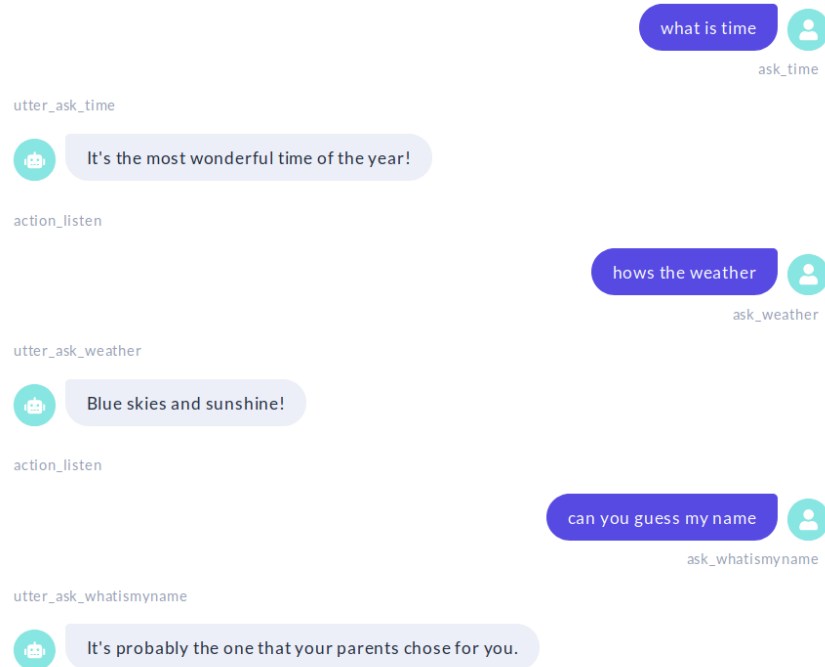


Fig 5.5 Basic Chat 3

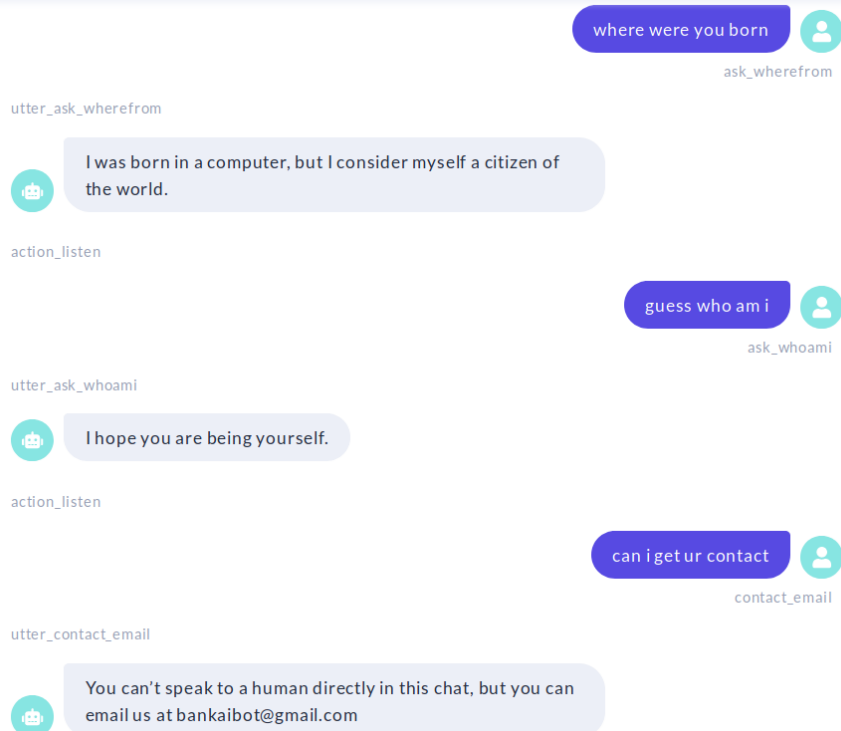


Fig 5.6 Basic Chat 4

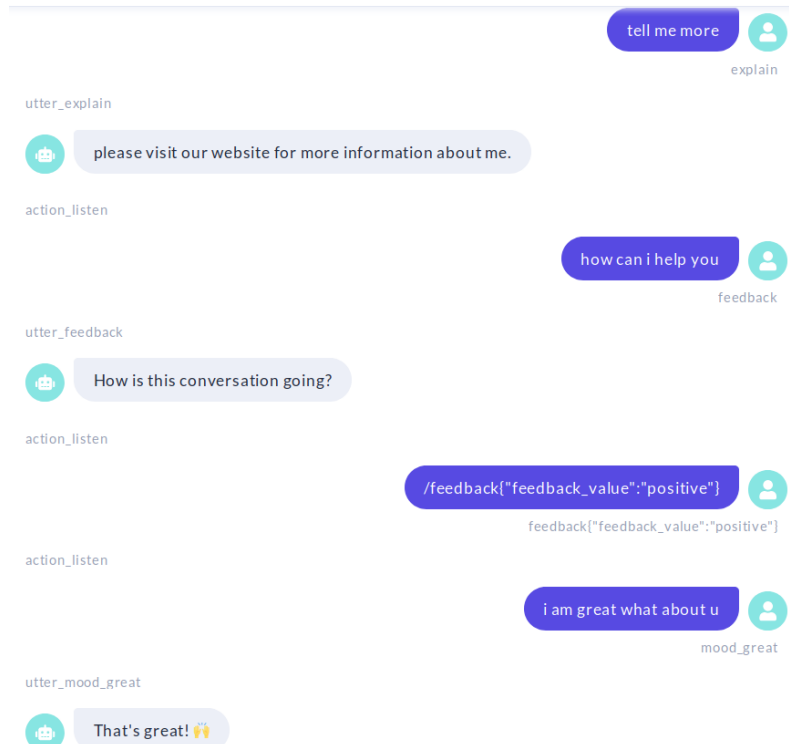


Fig 5.7 Basic chat 5

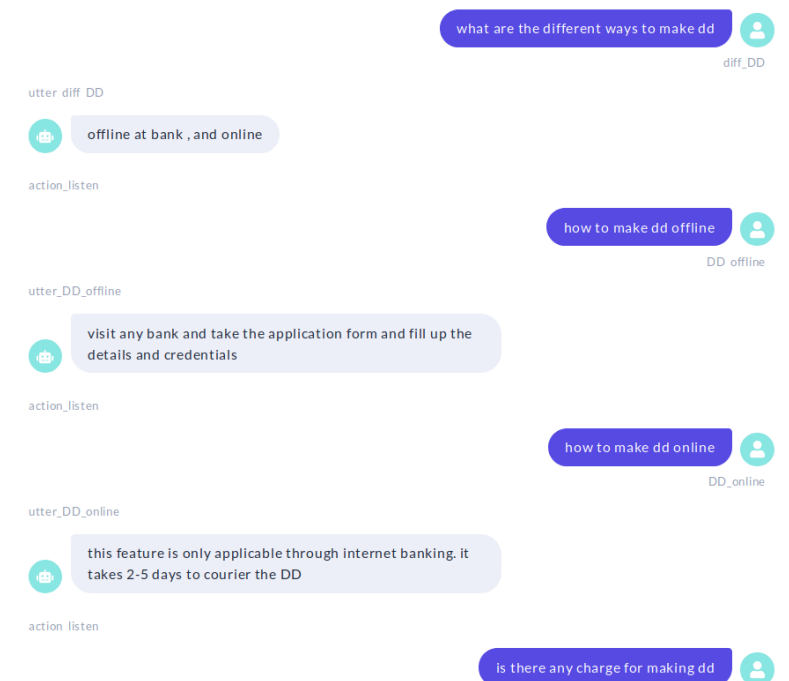


Fig 5.8 Banking chat - DD

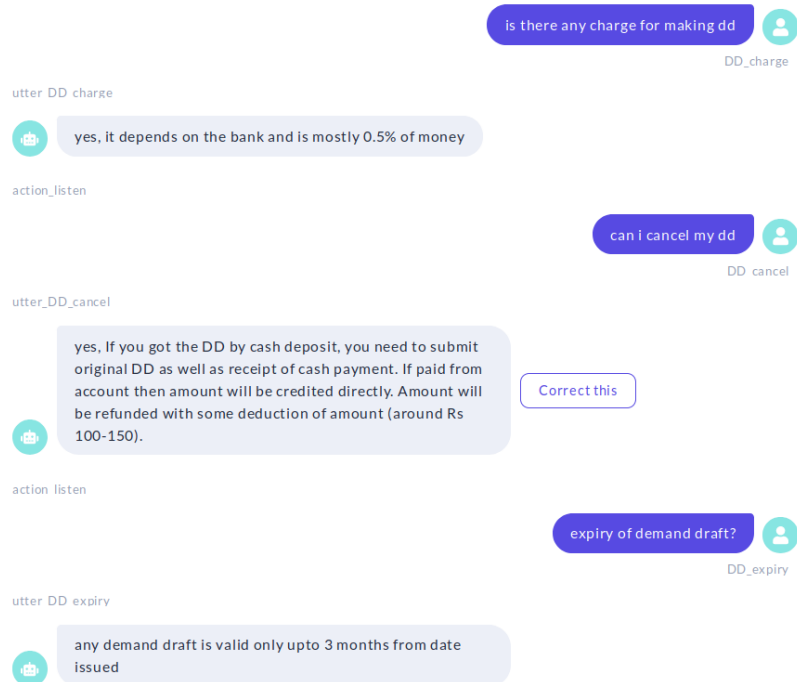


Fig 5.9 Banking chat – DD cancel

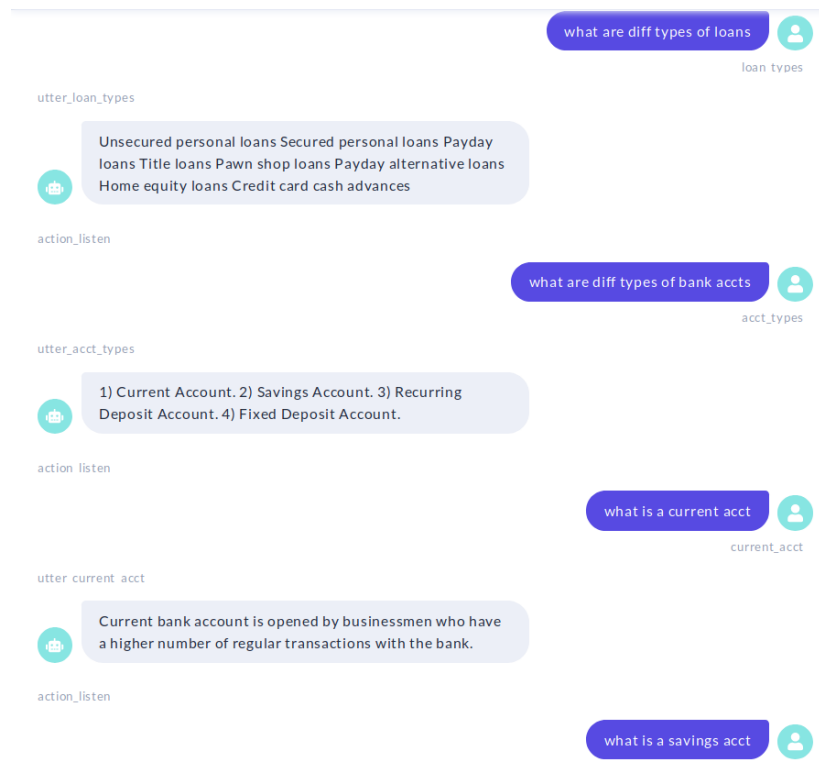


Fig 5.10 Banking Chat - loans



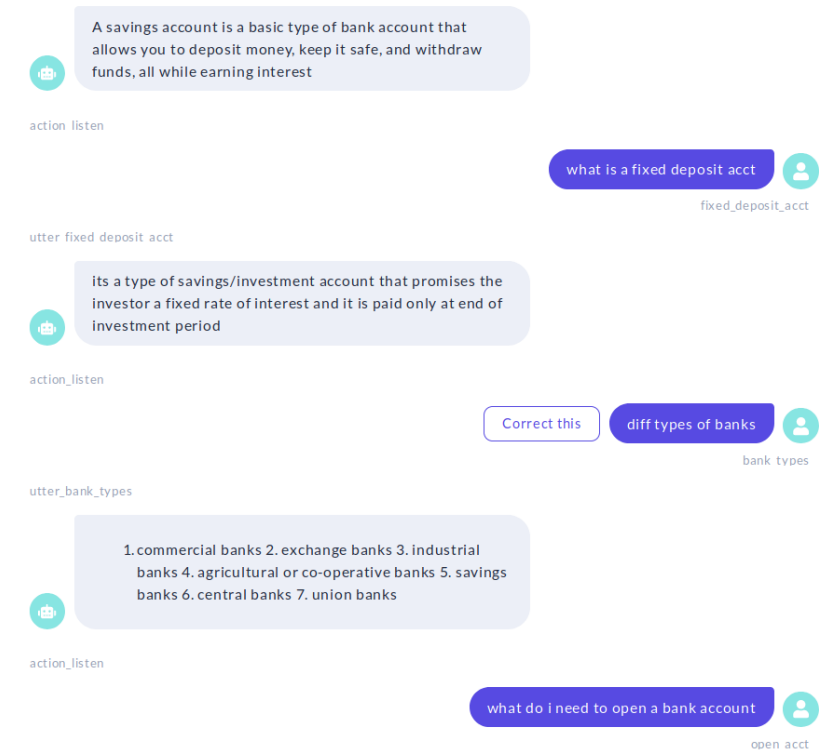


Fig 5.11 Banking Chat - account

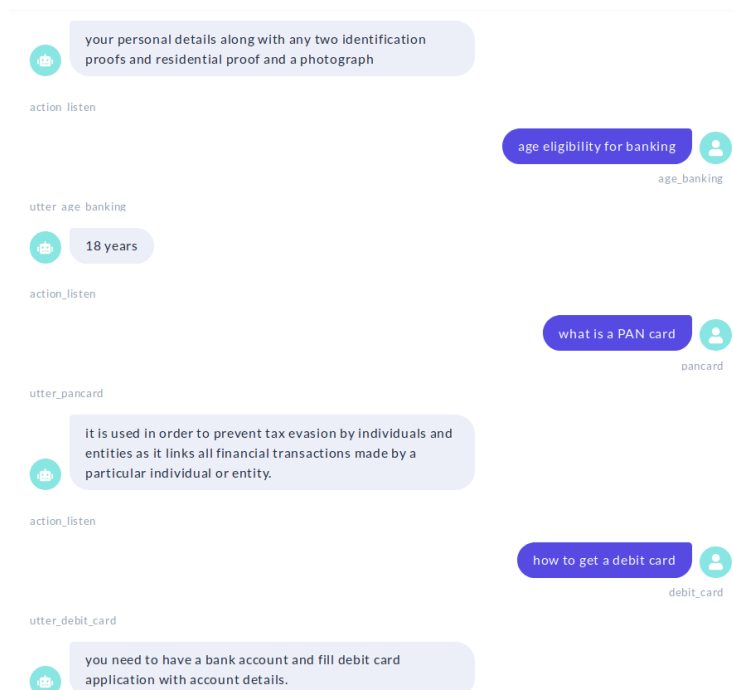


Fig 5.12 Banking chat – card

## 6. CONCLUSION AND FUTURE SCOPE

### CONCLUSION:

With the increasing need for individualized, context-rich Customer Care Banks are increasingly investing into service automation. That's why Chatbot is a revolution in Banking. Bots help serve clients faster, at lower costs, wherever they are and whenever they wish. Natural Interaction overcomes the limitation of traditional pre-recorded audio/visual guides. Chatbots benefit the functionality improvement, Time saving, Service flexibility, efficiency and service improvement and also costs reduction and sales increase.

The successful adoption of **chatbots** by end users has led to the use of more and more bots in advanced artificial intelligence technologies and their usage by a custom software development company. Even there are reports that 80–85% of businesses will be deploying advanced **chatbots** by 2020.

### FUTURE SCOPE:

Increasing the efficiency and making it more featuristic. Integrating with API's and connecting it across various platform. The successful adoption of chatbots by end users has led to the use of more and more bots in advanced artificial intelligence technologies and their usage by a custom software development company. Even there are reports that 80–85% of businesses will be deploying advanced chatbots by 2020. Enhance the chatbot reasoning capabilities and natural language interaction. Addition of slots, implementing on different pipelines and configurations and improving the efficiency of entity extractors. Making the chatbot to respond for any banking related issues and also adding speech recognition is a challenge to be faced.

## BIBLIOGRAPHY

- [1] <https://rasa.com/docs>
- [2] <https://rasa.com/docs/rasa/core>
- [3] <https://rasa.com/docs/rasa/nlu>
- [4] <https://rasa.com/docs/rasa-x/>
- [5] <https://blog.rasa.com/>
- [6] [www.wikipedia.org](http://www.wikipedia.org)
- [7] <https://chatbotslife.com>
- [8] <https://towardsdatascience.com>
- [9] <https://www.analyticsvidhya.com>
- [10] <https://hackernoon.com/build-simple-chatbot-with-rasa>
- [11] <https://developers.facebook.com>

## APPENDIX-A : SKELETON OF RASA

Since hype was to match chatbot with humans. We will take the human analogy to understand the components of the chatbot.

**Bot configuration :** Firstly we will understand the body parts of the human (mostly brain) which we call “Bot configuration” in the bot world.

Primary thing we humans do is communicate. And language is the primary means of communication. So for the bot as well we need to set language. We will use the English language for the bot. But you can build a multi-lingual bot with RASA. For more information about languages supported by rasa refer: <https://rasa.com/docs/rasa/nlu/language-support/>

The point is whenever we hear something we process the information through millions of neurons to understand the meaning of the sentence with its context, etc. And our brain is smart enough to generate a proper response based on a question. So are we going to build that intelligent bot?. Hold on! We can but not right now. The best way to think and start building a chatbot is like a newborn baby. It learns with experience :) Now let's understand how the brain of chatbot works. It's called NLU(Natural language understanding) unit where it's components do the job. Component includes as follows.

**1. Tokenization:** We read and understand the sentence word by word, right? Similarly, tokenizer will break the sentence into words(called word tokenizer). For more information on RASA supported tokenizer refer: <https://rasa.com/docs/rasa/nlu/components/#tokenizers>

**2. Featurizer:** We infer meaning by words and when all words are combined in a sentence then we infer the meaning of the sentence with context, right? Similarly, tokenized words are used as features to the post components of the pipeline. These features are has meaning of the word(mathematically) which is called Embeddings. Get to know more about word embedding here. Embedding comes in below two flavors.

## **Embedding:**

### **2.1. Pre-trained:**

Here word embeddings are already trained on huge text datasets with various state-of-the-art architecture. Popular word embeddings are XLNet, BERT, Glove, etc. We can use word embedding as it is in our NLP pipeline when we don't have much training data. This technique is called as transfer learning.

### **2.2. From scratch:**

When pre-trained does not work well because it might have trained on your domain-specific then we can train our own word embedding from scratch. It is recommended when you have sufficient training samples. RASA supports both types of word embedding. Refer this for more: <https://rasa.com/docs/rasa/nlu/choosing-a-pipeline/#a-longer-answer>

For more information on RASA supported featurizers refer: <https://rasa.com/docs/rasa/nlu/components/#featurizers>

### **2.3. Count vectorizer:**

You can convert a sentence into features using a bag of words. Where you can have unigram, bi-gram, tri-gram. Check this for more information: <https://rasa.com/docs/rasa/nlu/components/#countvectorsfeaturizer>

**Another interesting tweak is to increase the number of n-grams, which is 1 by default. By using a max\_ngram of 2, you will create additional features for each couple of consecutive words. For example, if you want to recognize “I'm happy” and “I'm not happy” as different intents, it helps to have not happy as a feature.**

### 3. Entity extraction :

These are a chunk of information we extract from sentences to complete the action. For example when we say I want to travel from Hyderabad to Mumbai by flight. Here the intent is "travel\_flight" and to fetch information we need to know source i.e Hyderabad and destination i.e Mumbai and couple more entities like date of travel etc.

Once the intent is identified and all entity is extracted then we can complete the action by calling the required API. Read more about entity extraction here: <https://rasa.com/docs/rasa/nlu/entity-extraction/>

### 4. Classifier :

Now you know the meaning(features) of the sentence(words through tokenization). It's time to classify to its appropriate category. For e.g I want to travel by cab should classify to travel\_cab and I want to travel by flight travel\_flight. All this is done by using Machine learning or Deep learning classifier.

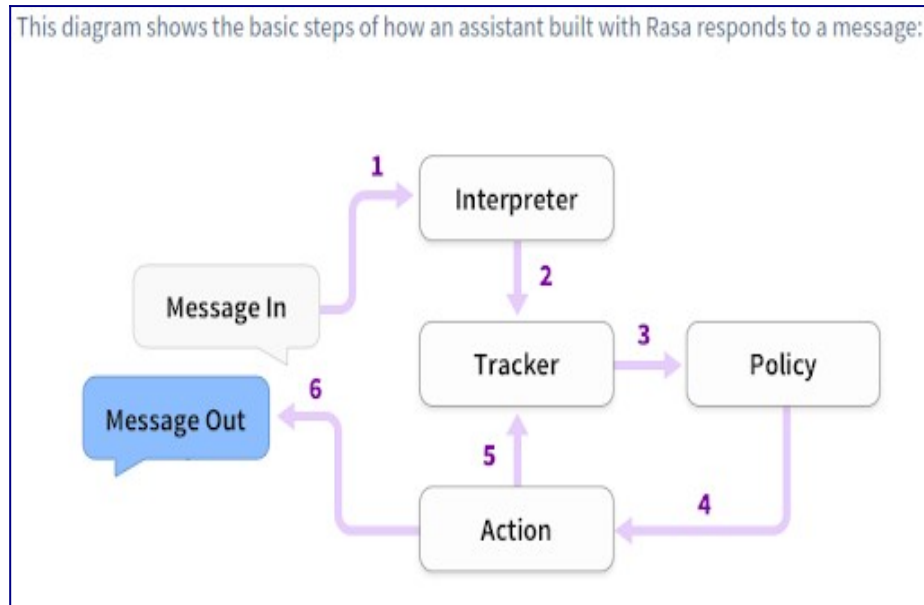
Read more about supported RASA classifier here: <https://rasa.com/docs/rasa/nlu/components/#intent-classifiers>

For more information about NLU pipeline and it's component refer: <https://rasa.com/docs/rasa/nlu/choosing-a-pipeline/> & <https://rasa.com/docs/rasa/nlu/components/>

Also read this in-depth information about NLU here: <https://blog.rasa.com/rasa-nlu-in-depth-part-1-intent-classification/>

Core policy Till now we saw how chatbot understands the user sentence and classifies to proper intent and extract entities. But we humans follow natural conversation where we remember context and reply accordingly.

So how does rasa handles all this? It is done through various elements of the RASA. Let's look at the architecture of the RASA.



Here Interpreter is part of NLU and Tracker, policy and action are part of Core.

- The message is passed to an Interpreter, which converts it into a dictionary including the original text, the intent, and any entities that were found.
- The Tracker is the object which keeps track of the conversation state. It receives the info that a new message has come in. Know more about it here: <https://rasa.com/docs/rasa/api/tracker-stores/>
- The policy receives the current state of the tracker, chooses the next action which is logged by the tracker and response is sent to the user. There are different policies to choose from. You will get more information here: <https://rasa.com/docs/rasa/core/policies/#policies> Along with policy “slots” which act as bot’s memory(context). It also influences how the dialogue progresses. Read more about slots here: <https://rasa.com/docs/rasa/core/slots/>

These settings are part of config.yml (Think this file as the brain of chatbot :P)

We have gone through the psychology of the bot. Now it’s time to look at the environment of the chatbot which will help it to learn. Just like a growing baby, he/she learn from whatever is experienced. Similarly will need to train a chatbot with right training data. which comes in the

form of text utterances part of defined intent with the tagged entity for training NLU and as a story (like a conversation) to train RASA core. Read more about training data for NLU here: <https://rasa.com/docs/rasa/nlu/training-data-format/> and for stories here: <https://rasa.com/docs/rasa/core/stories/>

As planned before we need to be very thorough with the scope of the bot. Hence we need to define its own universe in which our bot operates. It has the intents which it should classify to, entities which it should extract, slots which it should remember to maintain context, and actions which it should perform to complete the task. And response templates which bot should utter back. Read more about domain file here: <https://rasa.com/docs/rasa/core/domains/>

Actions are the things your bot runs in response to user input.

Read more about actions here: <https://rasa.com/docs/rasa/core/actions/>



## APPENDIX-B : RASA X

At Rasa, we are constantly looking for ways to make it easier for developers to build great conversational AI. Developing a conversational AI assistant is not an easy task, and improving it over time using real conversations that don't always go right is a whole new challenge.

**Rasa X is a tool designed to make it easier to deploy and improve Rasa-powered assistants by learning from real conversations.**

**Rasa X is packed with new UI features which allow you to:**

- **View and annotate conversations:** Filter, flag, and fix conversations that didn't go well to continually improve your assistant.
- **Get feedback from testers:** Share your assistant with testers to get immediate feedback and more training data
- **Version and manage models:** Track and manage your models, promote to production, and easily roll back.
- **Deploy anywhere:** Ready-to-deploy Docker containers and orchestration to run Rasa on premise or using your favourite cloud provider.

Rasa X can also be used to develop an assistant from scratch, but our main goal behind developing it was to build a new UI tool which would make it as easy as possible for your assistant to improve by learning from real conversations.

### **Step 1: Upgrading to Rasa 1.0**

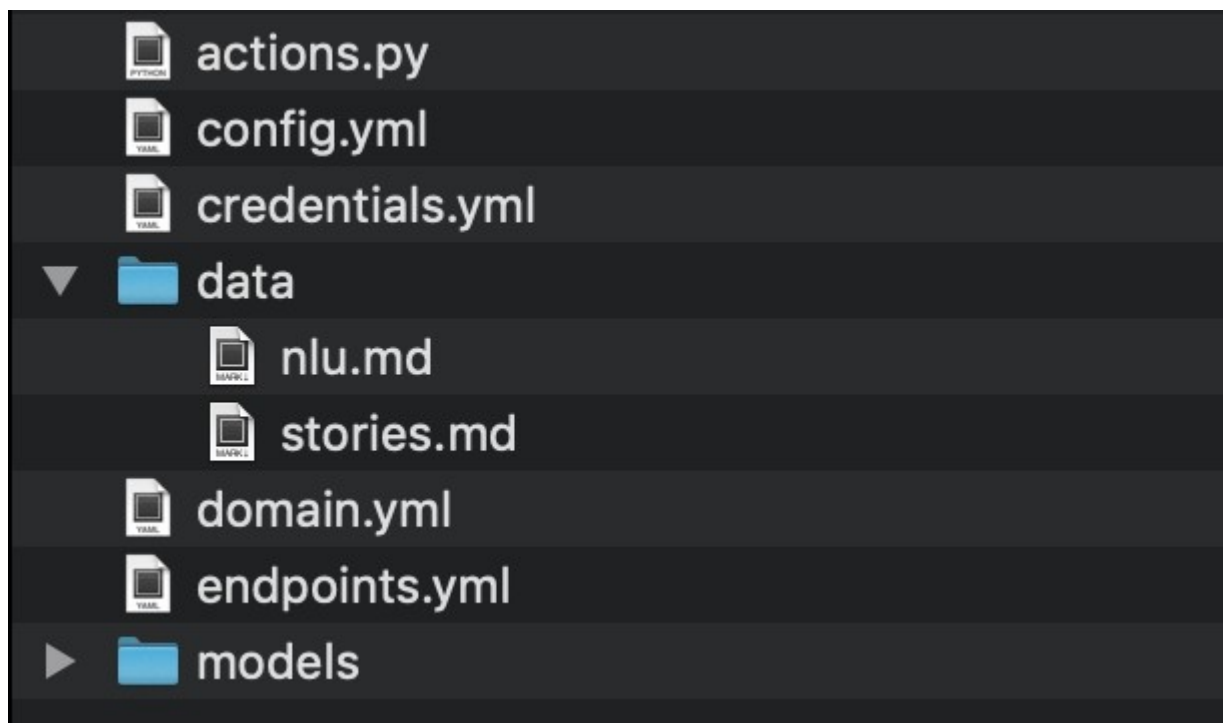
Alongside the launch of the Rasa X, the open source framework has reached an important milestone, with Rasa released as 1.0. It is the first version compatible with Rasa X and it comes with a bunch of new features like a new Mapping Policy, SQL tracker stores, new easy-to-use CLI and much more. Rasa 1.0 also is the first release of Rasa NLU and Core [merged into one package](#)

To run Rasa X, you have upgrade to Rasa 1.0.

To upgrade, run:

```
pip install rasa-x --extra-index-url https://pypi.rasa.com/simple
```

The biggest new feature in Rasa 1.0 is a command line interface (CLI) which makes it much easier to train and test your NLU and dialogue models on the command line. The only thing you have to check is that your project's files are in the right place. Your NLU and dialogue data should be stored in a `./data` directory in your project while all other files (domain, custom actions script, credentials, endpoints, nlu configuration file) should be kept in the main directory of your project folder. The image below represents the project structure the Rasa X expects to see:



After upgrading to Rasa 1.0 and updating your project structure, train a unified NLU and dialogue model by running the command below which will create a `.tar.gz` model file in a `./models` directory:

```
rasa train
```

With the files in this structure, you don't have to pass any flags to rasa commands to tell it where to find your domain, stories, etc. So the command really is just ***rasa train***. If you have a different project structure, you can still pass in arguments like ***rasa train -d /path/to/domain.yml***.

Once the model is trained, you can load it and talk to your assistant by running:

***rasa shell***

The command above will load your assistant for you to test. If, for example, you want to test only the NLU model, you can use:

***rasa shell nlu***

The new CLI comes with a bunch of other handy commands which should make it easier to build and improve your assistants using the file system and a command line.

## **Step 2: Using Rasa X with your existing assistant**

You can keep improving your assistant using the file system and command line, but a much easier way to take your assistant to a new level is to use Rasa X. Since you run Rasa X on your own machine, your training data and the conversations your users have with your assistant are kept completely private and never passed to Rasa. Ready to give Rasa X a spin? Let's go!

### **Extracting existing conversations from the tracker store**

If you have existing conversations between your bot and the users persisted in a tracker store (MongoDB, Redis, or SQL), you can import them to Rasa X using the [migration script](#). First, run the migration script on your command line:

***python migrate\_tracker\_store.py***

The script will connect to your tracker store and extract the conversations into *rasa.db* and *tracker.db* files. Make sure they are placed in your project directory and you are good to open and annotate them using the Rasa X UI

.

## Annotating conversations using Rasa X

First, launch the Rasa X UI. In your project directory run the command below:

```
rasa x
```

Once the Rasa X UI is launched, you can see the imported conversations in the Conversations tab of the Rasa X UI. Here, you can go through the conversations which were exported from your tracker store in the previous step and annotate them. The Conversations tab also lets you filter conversations by length, by specific intents and actions, and more. This comes in handy if you have a large number of conversations.

## APPENDIX-C : UNDERSTANDING THE CONVERSATIONAL CHATBOT ARCHITECTURE

Chatbot architecture is the heart of chatbot development. Based on the usability and context of business operations the architecture involved in building a chatbot changes dramatically. So, based on client requirements we need to alter different elements; but the basic communication flow remains the same. **Learn How to choose the right chatbot architecture and various aspects of the Conversational Chatbot.**

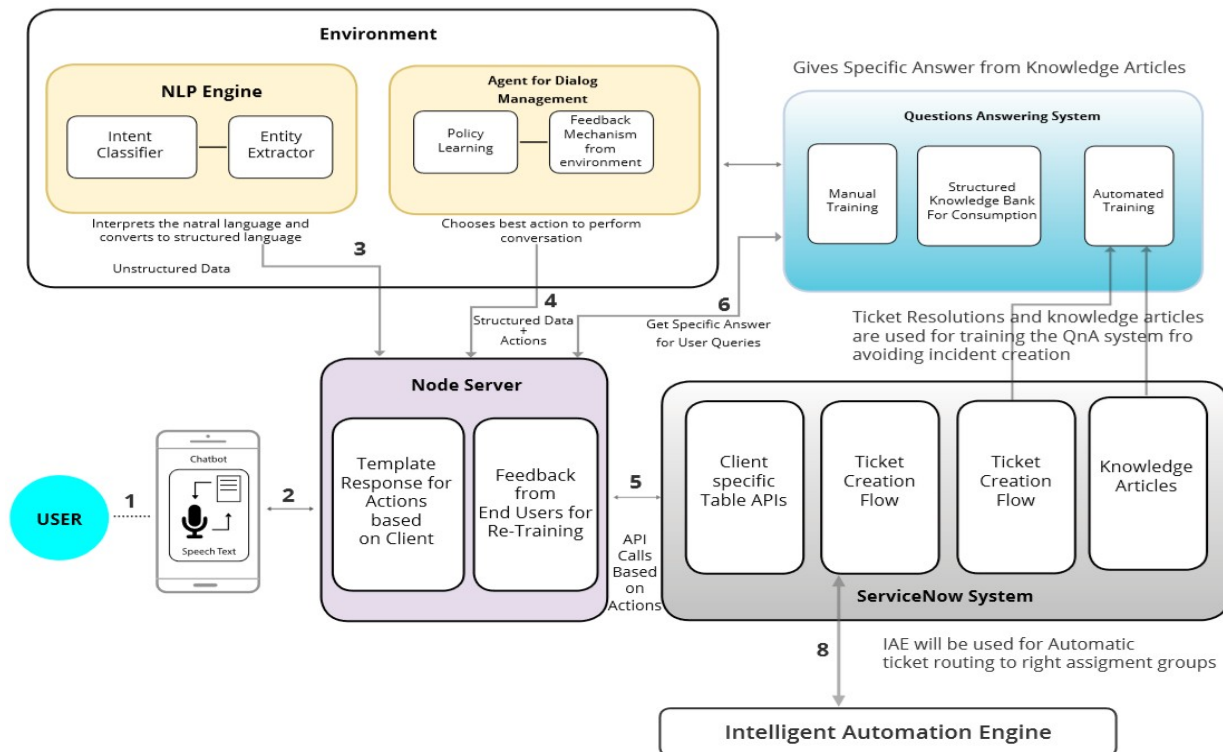
### Choosing the Right Chatbot Architecture

Choosing the correct architecture depends on what type of domain the [chatbot](#) will have. For example, you might ask a chatbot something and the chatbot replies to that. Maybe in mid-conversation, you leave the conversation, only to pick the conversation up later. Based on the type of chatbot you choose to build, the chatbot may or may not save the conversation history. For narrow domains a pattern matching architecture would be the ideal choice. However, for chatbots that deal with multiple domains or multiple services, broader domain. In these cases, sophisticated, state-of-the-art neural network architectures, such as Long Short-Term Memory (LSTMs) and [reinforcement learning](#) agents are your best bet. Due to the varying nature of chatbot usage, the architecture will change upon the unique needs of the chatbot.

### Understanding The Chatbot Architecture

Following are the key components of a conversational chatbot architecture:

1. Environment
2. Question and Answer System
3. Plugins/Components
4. Node Server / Traffic Server
5. Front-end Systems



**Figure: Architecture of a Conversational Chatbot**

## 1. Environment

This is where the core Natural Learning Process (NLP) engine and context interpretation happens

### NLP Engine

NLP Engine is the core component that interprets what users say at any given time and converts the language to structured inputs that system can further process. Since the chatbot is domain specific, it must support so many features. [NLP engine](#) contains advanced machine learning algorithms to identify the user's intent and further matches them to the list of available intents the bot supports.

NLP Engine further has two components:

- **Intent Classifier:** Intent classifier takes user's input identifies its meaning and relates back to one of the intents that the chatbot supports.

- **Entity Extractor:** Entity extractor is what extracts key information from the user's query.

### Agent for Dialogue Management

It manages the actual context of the dialogue. For example, the user might say "He needs to order ice cream" and the bot might take the order. Then the user might say "Change it to coffee", here the user refers to the order he has placed earlier, the bot must correctly interpret this and make changes to the order he has placed earlier before confirming with the user. Dialog management plugin enables us to do this.

Dialogue management further has following key plugins:

- **Feedback Mechanism:** Here the agent takes the feedback from user time to time to learn if the bot is doing fine with the conversation and the user is satisfied with the bot's response. This reinforces the bot to learn from mistakes and corrects itself in future conversations.
- **Policy Learning:** Policy learning is a higher-level framework that teaches the bot to take more of happy paths during the conversation to improve overall end-user satisfaction.
  - Broadly it creates the network of happy paths and routes the conversation to end-user satisfaction.
  - The bot then tries to learn from the interactions and follows the interaction flow about the conversation it had with similar users in the past.

## 2. Question and Answer System

This is the key component in answering users' frequently asked questions. Q & A system interprets the question and responds with relevant answers from the knowledge base. It has the following components

- **Manual Training:** Manual training involves the domain expert creating the list of frequently asked users queries and map its answers. This helps the bot quickly identify the answers to the most important questions.
- **Automated Training:** Automated training involves submitting the company's documents like policy documents and other Q&A type of documents to the bot and ask it to train it-

self. The engine comes up with a list of question and answers from these documents. The bot then can answer with confidence.

### **3. Plugins/Components**

Plugins offer chatbots solution APIs and other [intelligent automation](#) components for chatbots used for internal company use like HR management and [field-worker](#) chatbots.

### **4. Node Server / Traffic Server**

The server that handles the traffic requests from users and routes them to appropriate components. The traffic server also routes the response from internal components back to the front-end systems.

### **5. Front-End Systems**

Front-end systems can be any client-facing platforms. They can be the actual chatbot interfaces that reside in various platforms like:

- Facebook
- [Slack](#)
- Google Hangouts
- Skype for Business
- Microsoft Teams