

# CENTURION

UNIVERSITY OF TECHNOLOGY AND MANAGEMENT  
Andhra Pradesh



Centurion  
UNIVERSITY

Submitted By

**Names:** M.B. Sameeri  
**Reg.No:** 181801120004  
**Branch:** Computer Science and Engineering  
**Semester:** 7<sup>th</sup>

2021 – 2022

# **CENTURION UNIVERSITY OF TECHNOLOGY AND MANAGEMENT**

**SCHOOL OF ENGINEERING AND TECHNOLOGY, ANDHRA PRADESH**

**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**

## **EVALUATION SHEET**

**Name of the Candidate:** M.B. Sameeri

**Registration Number:** 181801120004

**Name of the Laboratory:** Natural Language Processing using Scikit Learn

<b>S. No</b>	<b>Date</b>	<b>Attendance (1M)</b>	<b>Labwork (4M)</b>	<b>Record (3M)</b>	<b>Viva (2M)</b>	<b>Total Mark (10M)</b>
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						
<b>Total Marks</b>						

**Signature of the Faculty**

## ACKNOWLEDGMENT

It is my pleasure to be indebted to various people who directly or indirectly contributed to the development of this work and who influenced my thinking, behaviour and acts during the course of study.

I am very much thankful to **Amit Kumar** for his outstanding support, cooperation, and motivation during the implementation of this project.

I express my sincere gratitude to **Ramana Rao**, Dean Academics, for providing academic support & opportunities.

I extend my sincere appreciation to **A. Avinash**, Supervisor, who provided her valuable suggestions and precious time in accomplishing my project report.

I also extend my sincere appreciation to **Dr Uttam Mande**, HoD, who provided his valuable suggestions and precious time in accomplishing my project report.

Lastly, I would like to thank the almighty and my parents for their moral support and friends with whom I shared my day-to-day experiences and received lots of suggestions that improved the quality of work.

M.B.Sameeri  
Reg No: 18180112004

# **Chat With CUBAP**

- M.B. Sameeri  
181801120004

**INDEX:**

<b>S.NO</b>	<b>TITLE</b>	<b>PAGE NUMBER</b>
1	<b>INTRODUCTION</b>	8
1.1	<b>USES</b>	8
1.2	<b>PREREQUISISTES</b>	8
2	<b>RASA</b>	9
3	<b>RASA CORE</b>	10
3.1	<b>DOMAIN</b>	10
3.2	<b>ACTIONS</b>	11
3.3	<b>STORIES</b>	12
3.4	<b>NLU DATA</b>	12
4	<b>IMPLEMENTATION</b>	13
4.1	<b>LSTM</b>	15
5	<b>TELEGRAM INTEGRATION</b>	16
6	<b>RESULTS</b>	17
7	<b>CONCLUSION</b>	20
8	<b>REFERENCES</b>	21

**List of figures:**

<b>FIG NUMBER</b>	<b>DESCRIPTION</b>
2.1	<b>Architecture of RASA Framework</b>
2.2	Application Flow of RASA Chatbot
3.1	Slots of the file domain for intents action
3.2	Stories from which action performed based on intent
3.3	Language is extracted based on intents examples
4.1	Server UI where the model runs
4.2	UI from RASA Framework
5.1	ngrok shell
6.1	Admission Intent on Telegram Bot
6.2	Transport Intent and EAMCET Intent action on Telegram Bot
6.3	Placement Intent and Facilities Intent action on Telegram Bot

## **ABSTRACT**

In the era of chatbots, besides imitating humans they can also perform complex tasks like booking tickets for movie etc. Out of various implementations, RASA is open source implementation for NLU and DIET model. It can interact with database, api, conversational flow, interactive learning with reinforcement Neural network. In this study, various features of rasa core are studied and upto much extent it can perform complex tasks. Implementation details are studied like interaction with database, API. Tracker Store has been examined with modifying the socket.io core file adding metadata to the user message data, so that user ip and port can be captured. Furthermore, the action, interactive learning and implementation details are tested on windows PyCharm IDE. To understand the type of RASA framework developed CUBAP an informational Chatbot.

## **1. INTRODUCTION**

A chatbot is an application that can initiate and continue a conversation using auditory and/or textual methods as a human would do. A chatbot can be either a simple rule-based engine or an intelligent application leveraging Natural Language Understanding. Many organizations today have started using chatbots extensively. Chatbots are becoming famous as they are available 24\*7, provide a consistent customer experience, can handle several customers at a time, are cost-effective and hence, results in a better overall customer experience. The chatbot design is the process that defines the interaction between the user and the chatbot. The chatbot designer will define the chatbot personality, the questions that will be asked to the users, and the overall interaction. It can be viewed as a subset of the conversational design. In order to speed up this process, designers can use dedicated chatbot design tools, that allow for immediate preview, team collaboration and video export. An important part of the chatbot design is also centered around user testing. User testing can be performed following the same principles that guide the user testing of graphical interfaces.

### **1.1 USES**

- Customer support
- Frequently Asked Questions
- Addressing Grievances
- Appointment Booking
- Automation of routine tasks
- Address a query

### **1.2 PREREQUISITES**

The prerequisites for developing and understanding a chatbot using Microsoft Azure are:

- Python installed
- Microsoft Build tools with visual c++ 14.0 installed. Link:  
<https://visualstudio.microsoft.com/downloads/>



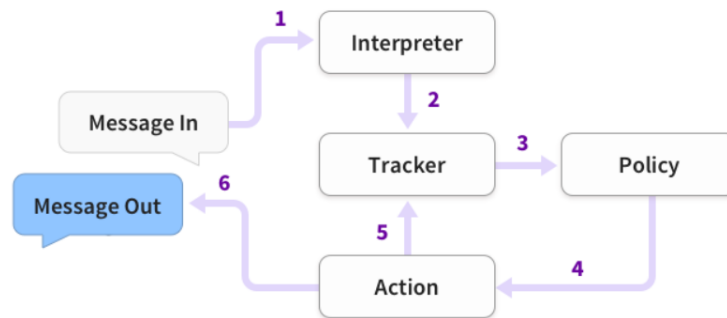
## 2. RASA

Rasa is an open-source machine learning framework for building contextual AI assistants and chatbots.

### Rasa has two main modules:

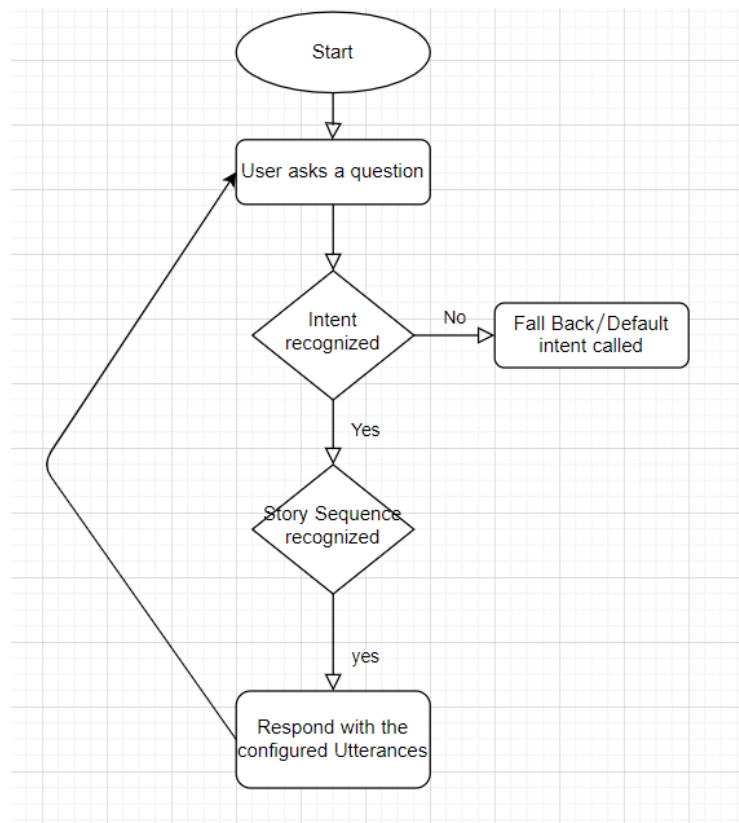
- NLU for understanding user messages
- Core for holding conversations and deciding what to do next

### 2.1 RASA ARCHITECTURE:



**Fig 2.1** Architecture of RASA Framework

### 2.2 APPLICATION FLOW



**Fig 2.2 :** Application Flow of RASA Chatbot

### 3. RASA CORE

Rasa Core takes in structured input: intents and entities, button clicks, etc., and decides which action your bot should run next. If you want your system to handle free text, you need to also use Rasa NLU or another NLU tool. The main idea behind Rasa Core is that thinking of conversations as a flowchart doesn't scale. It's very hard to reason about all possible conversations explicitly, but it's very easy to tell, mid-conversation, if a response is right or wrong. Hence, rather than writing a bunch of if/else statements, a Rasa bot learns from real conversations. A probabilistic model chooses which action to take, and this can be trained very easily.

The advantages of this approach are that:

- debugging is easier
- your bot can be more flexible
- your bot can improve from experience without writing more code
- you can add new functionality to your bot without debugging hundreds of rules.

#### 3.1 DOMAIN

The Domain defines the universe in which your bot operates. It specifies exactly:

- which intents you are expecting to respond to
- which slots you wish to track
- which actions your bot can take
- What are Slots?

Slots are the things you want to keep track of during a conversation. For example, in the messages above you would want to store “Mexican” as a cuisine type. The tracker has an attribute like `tracker.get_slot(“cuisine”)` which will return “Mexican”

```

utter_greet:
- buttons:
  - payload: /centurion_intro
    title: Centurion-AP
  - payload: '[Centurion University - Odisha](https://cutm.ac.in/)'
    title: Centurion-Odisha
  - payload: /cuee
    title: Centurion Entrance Examination
  text: Hey! I am a bot, powered by Centurion How can I help you?
utter_iamabot:
- text: Thank You CUTM-AP bot.
utter_did_that_help:
- text: Did that help you?
utter_goodbye:
- text: Thank you very much , Hope you got all the details.
utter_affirm:
- text: Is there anything I can help you further?
utter_deny:
- text: Please type in your query will try to reach out to the info you are looking for.
utter_cuee:
- text: Apply for CUEE https://cuee.cutm.ac.in/
utter_centurion_intro:
- buttons:
  - payload: /admission
    title: Admission
  - payload: /courses
    title: Courses
  - payload: /program-fee

```

**Fig 3.1 :** Slots of the file domain for intents action

## 3.2 ACTIONS

Actions are the things your bot can actually do. They are invoked by calling the `action.run()` method. For example, an action can:

- respond to a user
- make an external API call
- query a database

### 3.3 STORIES

A training data sample for the dialogue system is called a story. This shows the bot how to act and react to the inputs given by the user

```
story: centurion
steps:
- intent: centurion_intro
- action: utter_centurion_intro
- intent: centurion_spread
- action: utter_centurion_campuses
- action: utter_did_that_help

story: cuee
steps:
- intent: cuee
- action: utter_cuee
- action: utter_button
- action: utter_affirm

story: admission
steps:
- intent: admission
```

**Fig 3.2 :** Stories from which action performed based on intent

### 3.4 NLU DATA

The data to train the NLU can be given either by JSON or by MARKDOWN. Chose markdown, so this is what that looks like

```
- intent: faculty
  examples: |
    - teachers in each department
    - Faculty in each department their qualifications
    - How many teachers are there in each department
    - Qualification of Faculty in the department
    - Name of the Head in each department

- intent: location
  examples: |
    - Address where it is located
    - How much distance from the university to complex
    - Nearest Airport and Railway Station
    - How much distance from Vizianagaram
    - Exact location of the campus
    - Place of the campus
```

**Fig 3.3 :** Language is extracted based on intents examples

## 4. IMPLEMENTATION

- Create a new folder for your chatbot project.
- Open that folder using PyCharm
- Create a new environment for your chatbot project from pycharm or from anaconda prompt.
- Run the command `pip install rasa x` for installing all the rasa dependencies
- Run the command `pip install spacy` for installing spacy library.

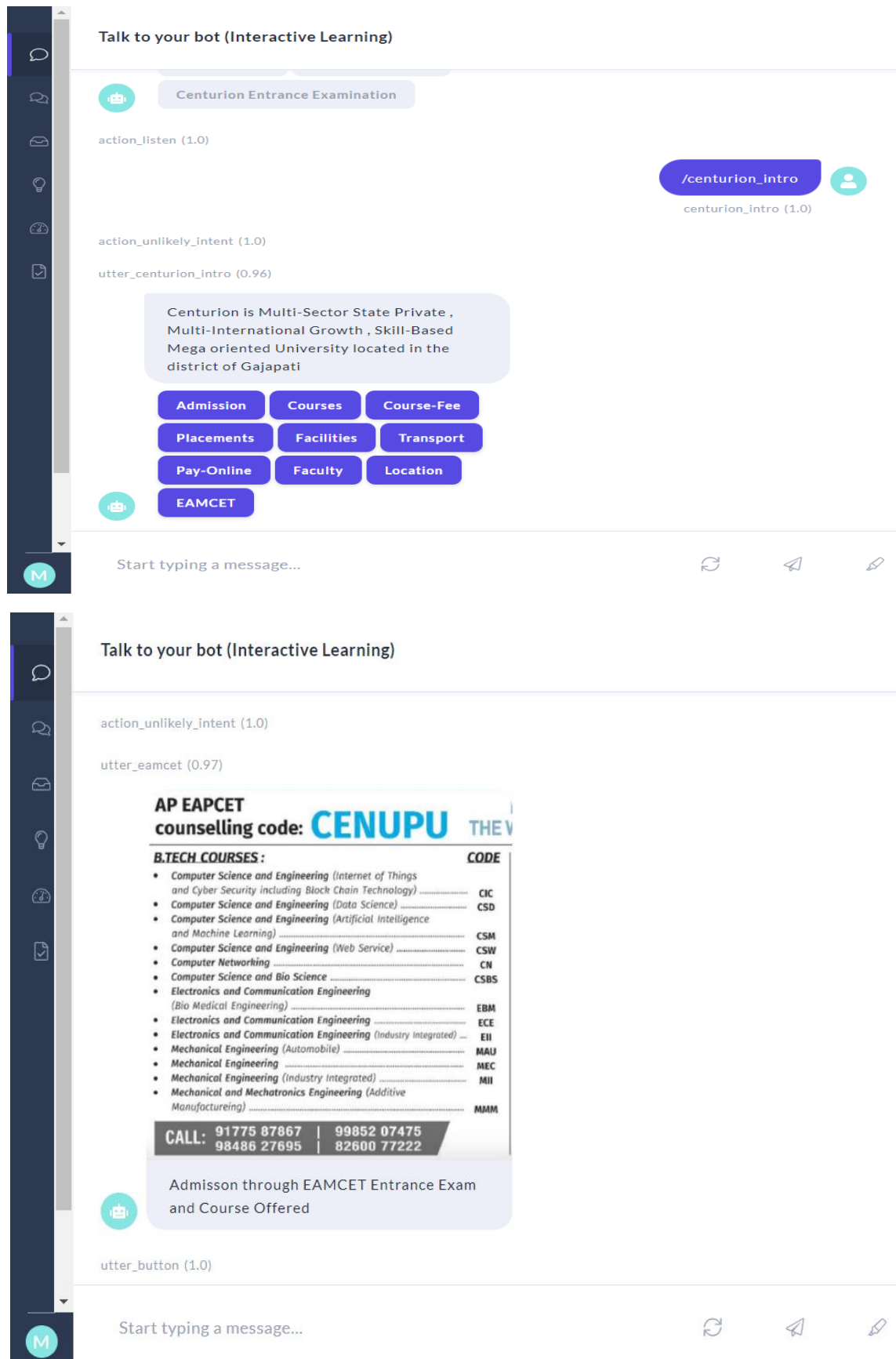
Then enter the following commands:

- `python -m spacy download en`
  - `python -m spacy download en_core_web_md`
  - `python -m spacy link en_core_web_md en`
- 
- After all this command run successfully, enter the command `rasa init` and for all the subsequent actions choose Y (for training the model etc).
  - You'll then end up with all the predefined structures with RASA
  - This file is used to create the conversation flows.
  - After all this, you can just enter the command '**rasa train**' to train the model with new conversation elements.
  - After the training is completed, enter the command '**rasa x**' to test your chatbot in the web UI.
  - Copy this URL in your web browser and you'll see the web UI for your chatbot

A terminal window with a dark background. The text 'The server is running at' is in green, followed by a blue URL: <http://localhost:5002/login?username=me&password=n8DDzwUAo9LL>.

```
The server is running at http://localhost:5002/login?username=me&password=n8DDzwUAo9LL
```

**Fig 4.1 :** Server UI where the model runs



**Fig 4.2:** UI from RASA Framework

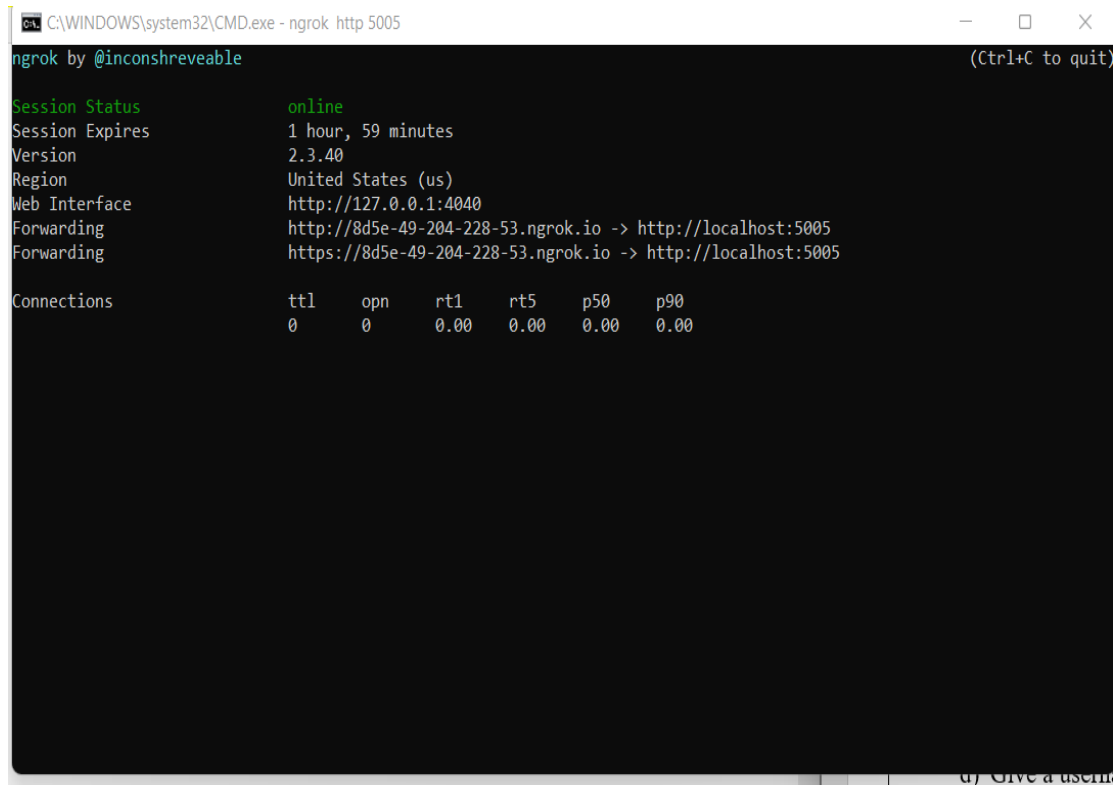
## 4.1 LSTM

Long Short-Term Memory networks – usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997), and were refined and popularized by many people in following work. They work tremendously well on a large variety of problems, and are now widely used.

- LSTMs are explicitly designed to avoid the long-term dependency problem.
- Remembering information for long periods of time is practically their default behavior, not something they struggle to learn!
- All recurrent neural networks have the form of a chain of repeating modules of neural network.
- In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.
- LSTMs also have this chain like structure, but the repeating module has a different structure.

## 5. Telegram Integration

- Download ngrok from <https://ngrok.com/download>
- After extracting the zip file, open the ngrok file and run it.
- In ngrok, enter the command '**ngrok http 5005**':



```
C:\WINDOWS\system32\CMD.exe - ngrok http 5005
ngrok by @inconshreveable (Ctrl+C to quit)

Session Status      online
Session Expires     1 hour, 59 minutes
Version             2.3.40
Region              United States (us)
Web Interface        http://127.0.0.1:4040
Forwarding            http://8d5e-49-204-228-53.ngrok.io -> http://localhost:5005
                    https://8d5e-49-204-228-53.ngrok.io -> http://localhost:5005

Connections          ttl   opn  rt1  rt5  p50  p90
                   0     0    0.00 0.00 0.00 0.00
```

**Fig 5.1 : ngrok shell**

Then go to telegram and create your own bot using Botfather:

- a) Open the telegram app and search for botfather(**it is an inbuilt bot used to create other bots**)
- b) Start a conversation with botfather and enter **/newbot** to create a newbot.
- c) Give a name to your bot
- d) Give a username to your bot, which must end in **\_bot**. This generates an access token.

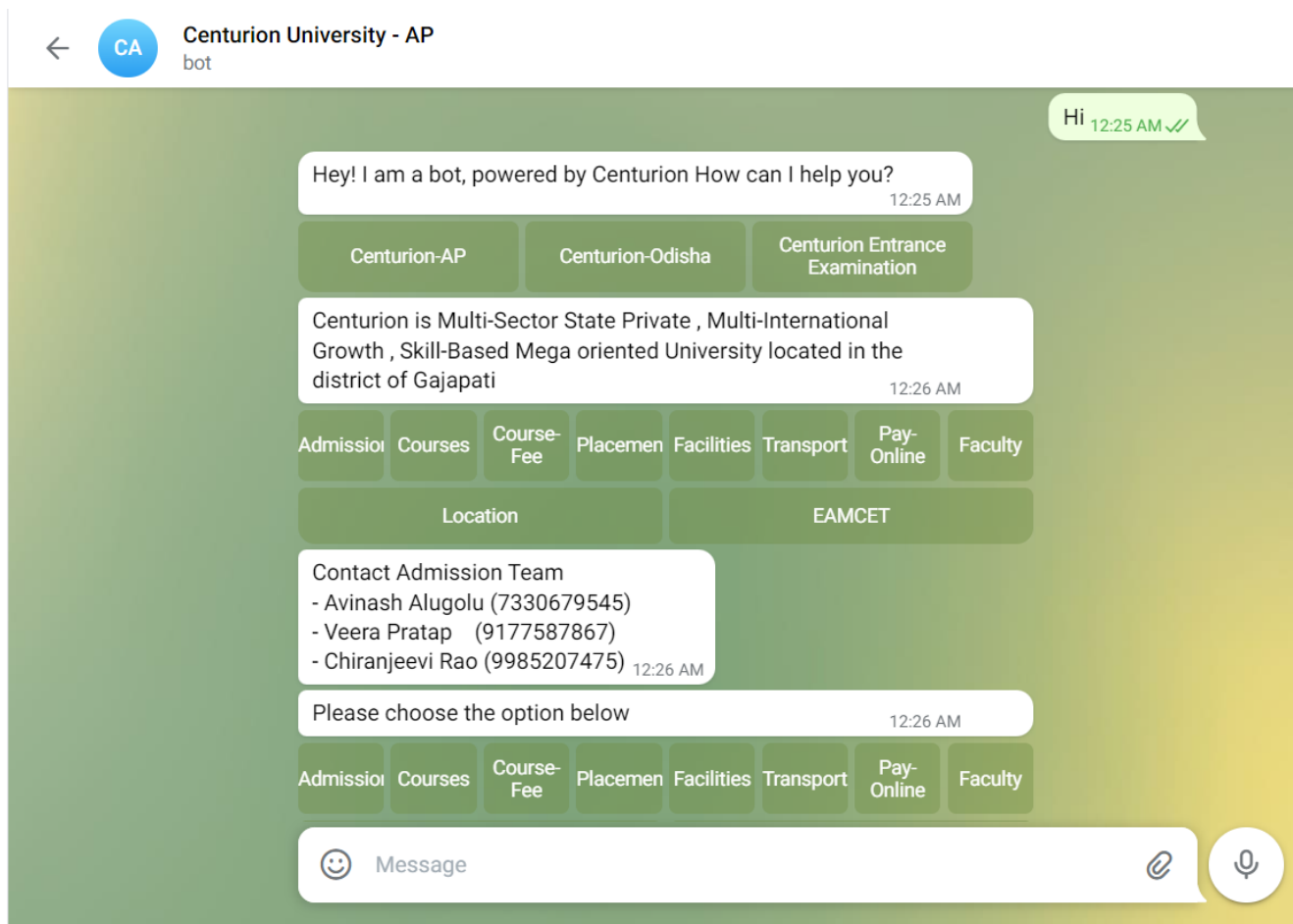
Open '**credentials.yml**' and enter:

```
1. telegram:
2. access_token: "obtained from telegram"
3. verify: "your bot username"
4. webhook_url: "https://<ngrokurl>/webhooks/telegram/webhook"
```

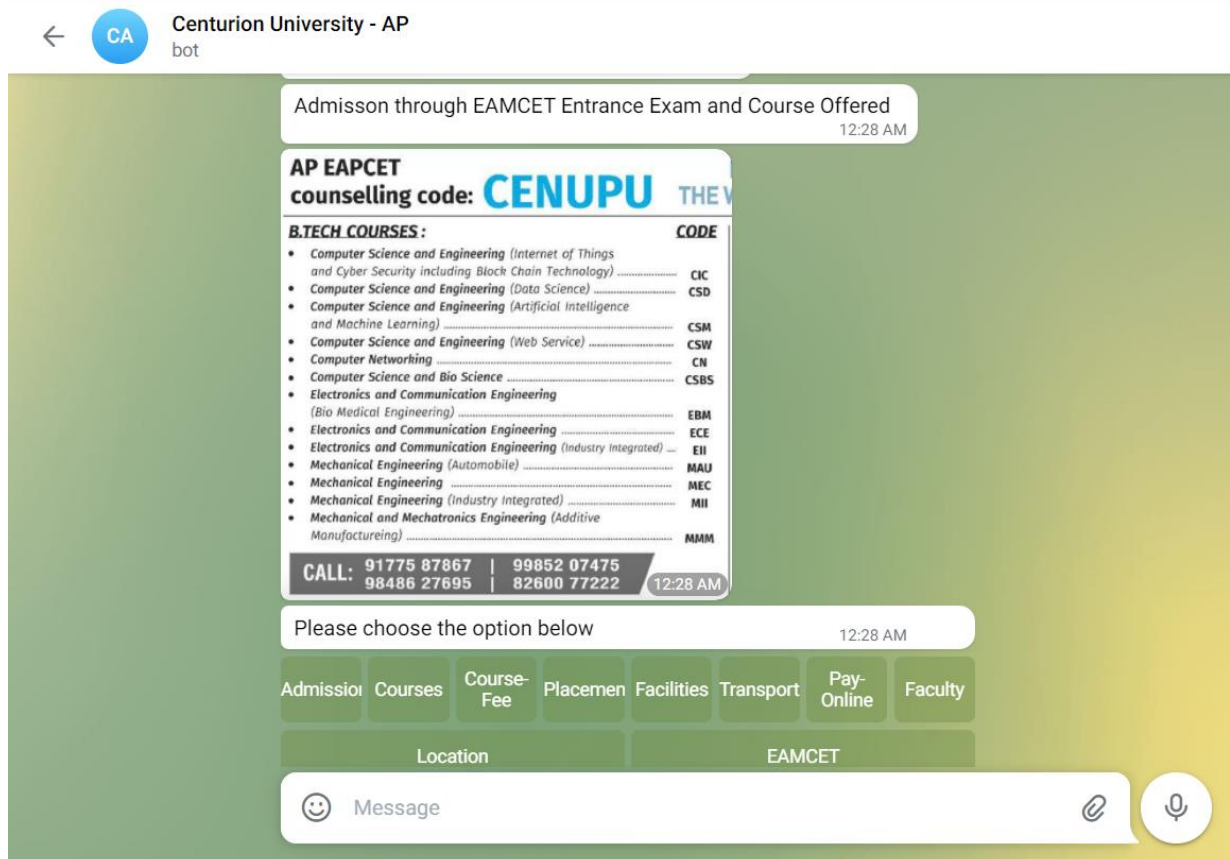
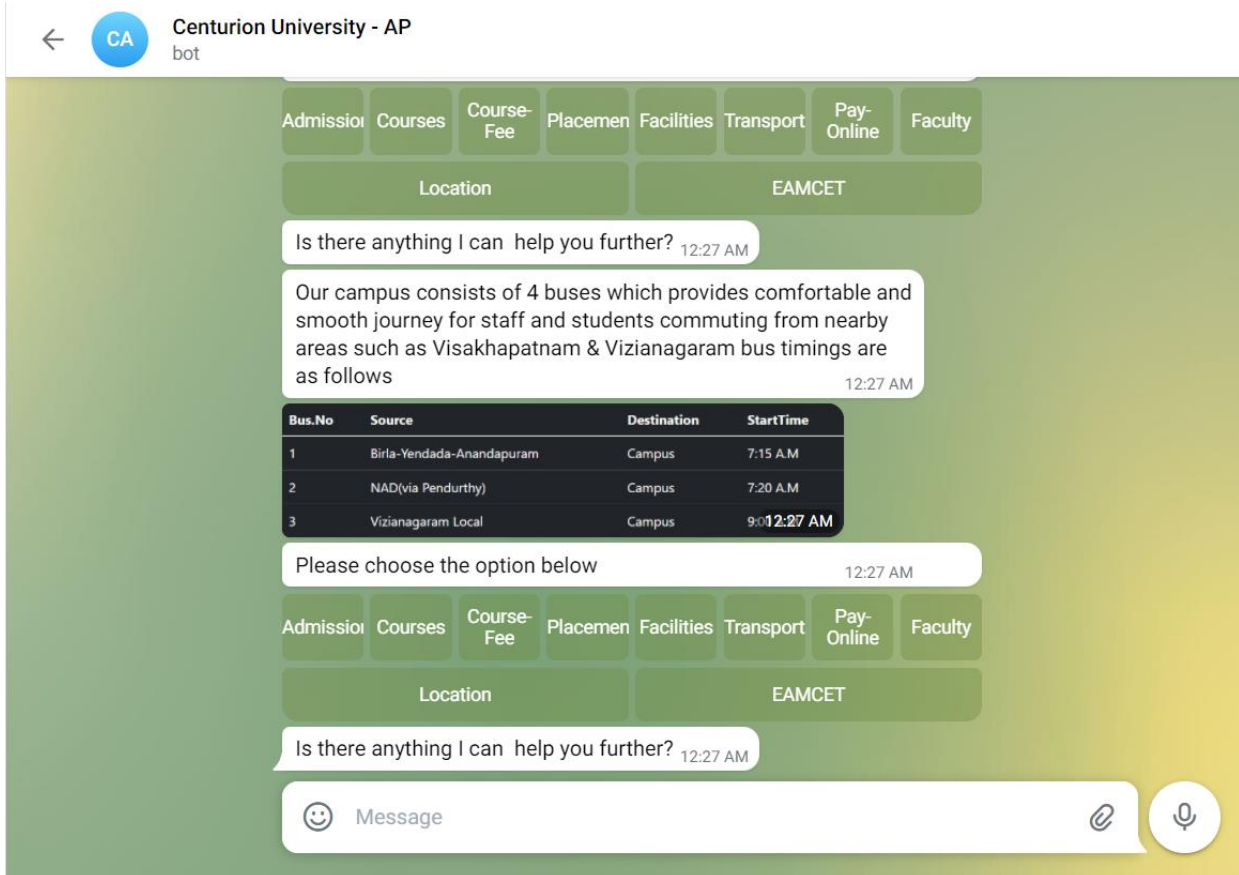


- Go to terminal and enter the command '**rasa run**'
- Open one more terminal and run the command '**rasa run actions**'
- Now, you can chat with your bot from Telegram.

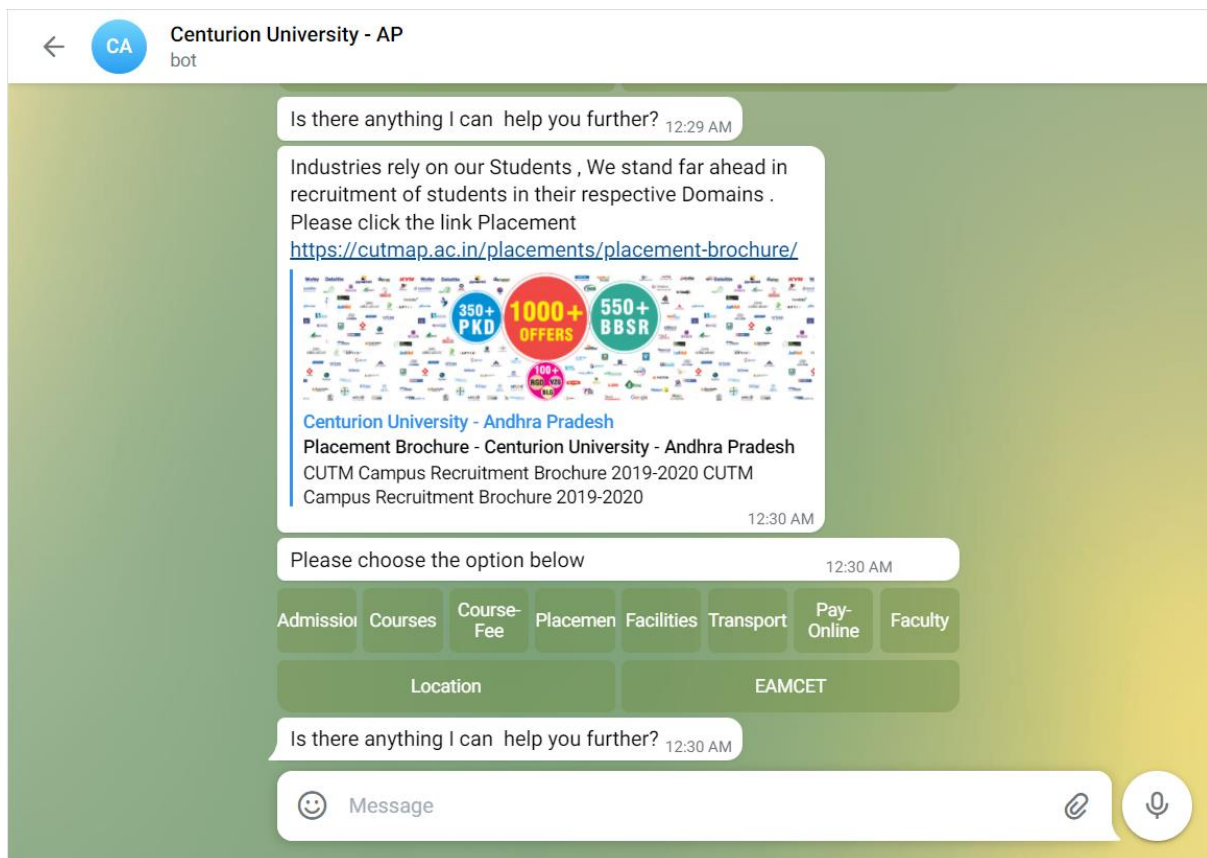
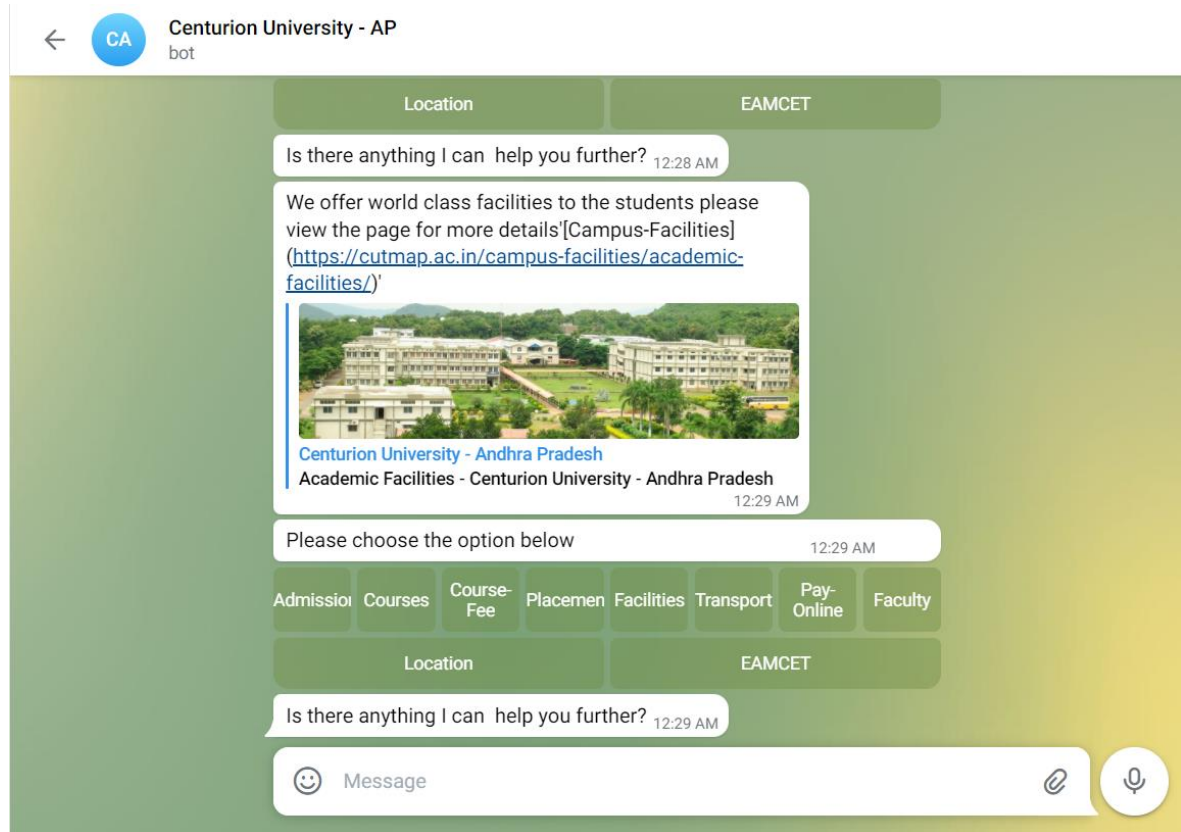
## 6. RESULTS



**Fig 6.1 : Admission Intent on Telegram Bot**



**Fig 6.2 : Transport Intent and EAMCET Intent action on Telegram Bot**



**Fig 6.3 : Placement Intent and Facilities Intent action on Telegram Bot**

## 7. CONCLUSION

Rasa is fully customizable and easily interchangeable. It is possible, and sometimes recommended to use Rasa Core or Rasa NLU separately. When using Rasa NLU, we can choose among several backend NLP (Natural Language Processing) libraries. The LSTM neural network which Rasa Core uses for action prediction can be easily exchanged for any other type of Neural Network, if you know how to implement them in Keras. The other competent frameworks have a lot of data sharing issues, as in we need to share data with them. It is an open-source bot building framework. It doesn't have any pre-built, pre-existing and downloadable models on a server that we can call using an API. This gives complete control of all the components in the chatbot. Finally, **CUBAP** is completely based on intent as well as NLU classification with LSTM as its backend based on the training and testing it can work both text as well payload based but more number of intents are to be added so as to reduce the ambiguity of the model and better performance over new text. Since pipeline is very similar where all the preprocessing happens it is up to the NLU trained by the user . Telegram integration is successful but UI on mobile is not suggested due to react issues and can be solved in further bugs.

## 8. REFERENCES

- [1] Jiao, Anran. "An intelligent chatbot system based on entity extraction using RASA NLU and neural network." *Journal of Physics: Conference Series*. Vol. 1487. No. 1. IOP Publishing, 2020.
  
- [2] Sharma, Rakesh Kumar, and Manoj Joshi. "An Analytical Study and Review of open source Chatbot framework, Rasa." *International Journal of Engineering Research and* **9.06** (2020).
  
- [3] Windiatmoko, Yurio, Ahmad Fathan Hidayatullah, and Ridho Rahmadi. "Developing FB chatbot based on deep learning using RASA framework for university enquiries." *arXiv preprint arXiv:2009.12341* (2020).
  
- [4] Pérez-Soler, Sara, et al. "Choosing a chatbot development tool." *IEEE Software* (2021).
  
- [5] Windiatmoko, Yurio, Ridho Rahmadi, and Ahmad Fathan Hidayatullah. "Developing Facebook Chatbot Based on Deep Learning Using RASA Framework for University Enquiries." *IOP Conference Series: Materials Science and Engineering*. Vol. 1077. No. 1. IOP Publishing, 2021.
  
- [6] Rasa Official documentation <https://rasa.com/docs/rasa/user-guide/installation/>