![Studocu](studocu logo)

Chatbot final documentation

Python (Bangalore University)

Palestine Polytechnic University

College of IT and Computer Engineering

Graduation Projects

Project name:

Admission Chatbot

Project team: Ali Jboor & Maher Salamin

Supervisor: Mr.Mohammad Jabari

May-2021

Abstract

As a result of the rapid technological development and the development of the chatbot concept and the time and effort it can save. Many specialized frameworks have emerged to undertake chatbot creation and development. By relying on artificial intelligence, the chatbot has integrated machine learning within it, and it has become more comprehensive and wider for various technological fields. Therefore, we will create a chatbot for the University's Admission and Registration, the project aims to build a chatbot to facilitate the process of accessing information related to students' inquiries towards admissions, Registration and the university itself. The motivation for the work of this project is that there is no university-level equivalent from previous graduation projects, as this project mainly targets all palestinian tawjihi students and other palestinians, non palestinian students . As a conclusion, it lies in answering frequent and common questions by people and providing the answer to these questions at any time the person wants.

What is natural language processing (NLP)?

Natural language processing, which evolved from computational linguistics, uses methods from various disciplines, such as computer science, artificial intelligence, linguistics, and data science, to enable computers to understand human language in both written and verbal forms.

What is natural language understanding (NLU)?

Natural language understanding is a subset of natural language processing, which uses syntactic and semantic analysis of text and speech to determine the meaning of a sentence.

Content list

Figures and Tables list

Figures:

4

Tables:

Chapter 1: Introduction

1.1 Introduction

Chatbot is a computer program that humans will interact with in natural spoken language and including artificial intelligence techniques such as NLP (Natural language processing) that makes the chatbot more interactive and more reliable.

Based on the recent epidemiological situation, the increasing demand and reliance on electronic education has become very difficult to access to the university due to the curfew imposed, and this has led to limited access to information for academics at the university.

This project aims to build a chatbot for Admission and Registration to answer every person who asks about the university, colleges, majors and admission policy.

1.2 Problem Statement

At the start of each academic semester, registration opens for those wishing to join the university in various disciplines, and telephone calls for admission and registration abound. This leads to an increase in the loads and work for the employees of the Deanship of Admission and Registration as a result of the constant pressure of those wishing to register and their families by flocking to the Deanship, so the employees are not able to answer the phone calls and social media. This often leads to many students who wish to register to be ignored.

1.3 Objectives
- Save effort and time for both the Admission and registration staff and students who wish to enroll.
- Provide detailed information about colleges and majors.
- Easy access to information.

1.4 Scope
- People who wish to enroll Palestine Polytechnic University
- Admission and registration staff

6

1.5 Timeline

Number Of Task	Task name	Required time in weeks
1	Proposal	2
2	Information planning and collecting	3
3	Background and Alternatives	2
4	Determine system requirements	2
5	System requirements analysis and design	2
6	System programming and configure	9
7	System Test	3
8	System documentation	All project period

Table 1.1: Timeline .

task number	first semester work	WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 5	WEEK 6	WEEK 7	WEEK 8	WEEK 9	WEEK 10	WEEK 11
1	Proposal	■	■									
2	Information planning and collecting			■	■	■						
3	Background and Alternatives						■	■				
4	Determine system requirements								■	■		
5	System requirements analysis and design										■	■

Table 1.2: First Semester Timeline.

Chapter 2: Background and Alternatives

2.1   Chatbot :

A chatbot is a computer program that can simulate a conversation or chat with a user in natural language through messaging applications, website, or mobile applications and interact with users according to their input and should be available 24/7.

Chatbots are developed and became so popular due to the increased use of smart devices and IoT technology.

2.2   Types of chatbots

a.   Base-line chatbot:

It is a chatbot that is based on a database and uses if / then logic to create a conversation flow and that takes a lot of time to ensure the understanding of the question and the answer needed.[1]

b.   AI chatbot:

This type of chatbot is more complex than base-line but it is more interactive and personalized and needs big data training to be impressive if the problem is matched to their capabilities.[2]

c.   Hybrid Model:

A hybrid approach mixes the Base-line & AI chatbot to make it smart and his behavior more expected by depending on database and Ai algorithm to work together.[3]

2.3   How do chatbots work?

Briefly and as mentioned in the definition, humans interact with chatbots.

There are two ways to interact with a chatbot:

a.   Text

chatbot analyzes the inputted text and matches the text with predefined data called intents which are categorized to manage the conversation. The user utterance is tagged with one of these intents, even if what the user says stretches over two or more intents. Most chatbots will take the intent with the highest score and take the conversation down that avenue.

b. Voice

Some chatbots can interact and understand the voice of the user using a set of application programming interfaces (api's) that converts the recorded voice to the language and then convert the voice to words of that language and then deal with the transformed text as mentioned above.

2.4   Options to build a chatbot.

a. From Scratch

At first we have to identify the opportunities for our chatbot and decide its field and scope to achieve efficiency and accuracy. and a precise understanding of the customer needs is required to solve the operational challenges.

Then the design of the bot comes to be a significant stage to decide the user engagement with your app or website.

and we can categorize chatbot interactions as structured and unstructured interactions.

❏ Structured interaction. You already know about this kind of interaction. You know what your customers will ask and can design it easily — it's just like an FAQ section of your app or website[4] . This information will link to your contact information, services, products, etc.

❏ Unstructured interaction. The unstructured conversation flow includes freestyle plain text. It's hard to predict what queries will emerge and it looks like an extempore speech competition for your chatbot. the role of AI comes to lights here, it decodes the context of the text based on NLP analysis. while the same NLP will provide a voice to the chatbot.[5]

The later choice will need specialized chatbot developers with an understanding of programming languages, machine learning, and AI. We can use some of the code-based frameworks to build and handle the chatbot like wit.ai and api.ai.

b. Using platforms

It is similar to scratch chatbots but the only difference is that you do not have to hire a specialized developer and use the chatbot builder platforms like Chatfuel, Botsify and Rasa, it's not hard or impossible to achieve it. but it's not possible to create a NLP-enabled chatbot that can deal with unstructured data.

9

2.5 Chatbot Platforms alternatives:

a. **IBM Watson:** is touted as a question-and-answer system that can be used to build applications and chatbots. The IBM Watson platform allows us to create an application that shares a dialog interaction between our chatbot and users on Quick n' Easy Projector Rentals. The IBM interface is simple to use, and no back-end coding is shown at first glance. The chatbot can be easily integrated into other applications such as Slack, Facebook, and Twilio.

b. **Google Dialogflow:** is an easy to understand conversational agent. Theoretically, we can have a bot up and running by understanding 3 core concepts: intents, entities, and dialog control. As stated earlier, these general concepts are followed across a majority of the chatbot platforms we played with.

c. **Rasa:** is an open source chatbot that is equipped with a natural language processing tool. The open source tool is called Rasa NLU. You can tweak and customize the machine learning algorithm that Rasa uses so that you can create a model that provides the results you desire. Rasa NLU can be run wherever you want it to, and none of your training data has to be passed over to Google, Microsoft, Amazon, or Facebook to train your chatbot.

2.6 Selected Platform:

We will use the rasa framework to build the required chatbot, but why did we choose rasa ?

Will, rasa has a lot of advantages such as:

- Highly customisable with various pipelines can be employed to process user dialogues.
- The rasa framework can be run as a simple http server or can be used from python, using APIs.
- It has the Rasa-nlu, when run on a server, can mimic other commercial NLP platforms such as LUIS or wit.ai. This makes it easy to migrate an existing application to rasa-nlu.[3]

But as we know, nothing is perfect and rasa has its disadvantages like :

1. server requirements although spacy is a very fast NLP platform, it seems to be very memory hungry.
2. Learning curve - Installation, configuration and training phases require machine learning expertise (at least basic level)
3. Context based conversation not available out of the box - Rasa-nlu does not maintain the context automatically. This has to be programmed separately into the chat service.

2.7   How To Setup Rasa and start a demo ChatBot[8]

● First, we have to know that Rasa needs python to work, so we'll install python and the version should be from 3.6 to 3.8 at maximum.
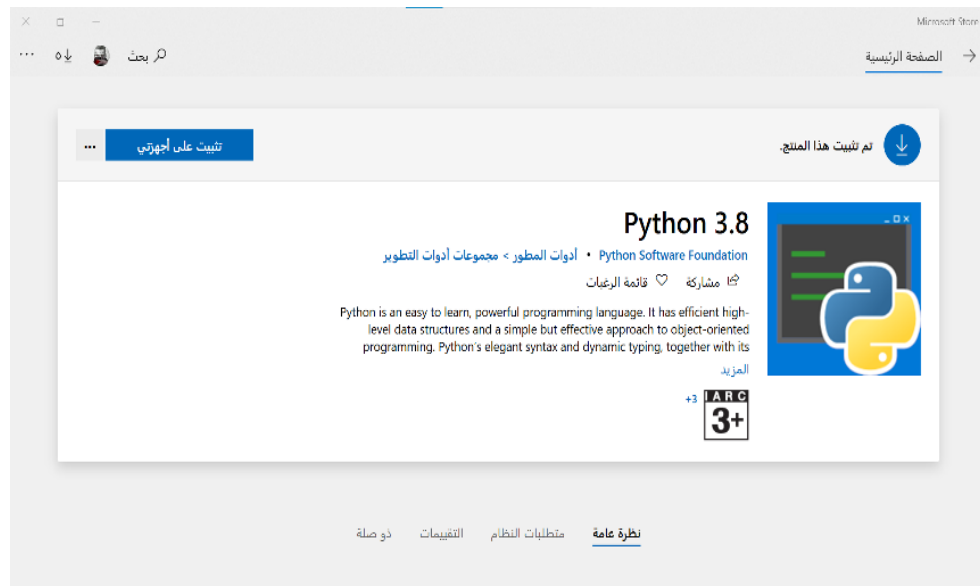

Figure 2.1:Python main screen in microsoft store.

Go to Microsoft Store and search for python 3.8. Then,Check if your Python environment is already configured by opening the cmd : type **python3 --version**
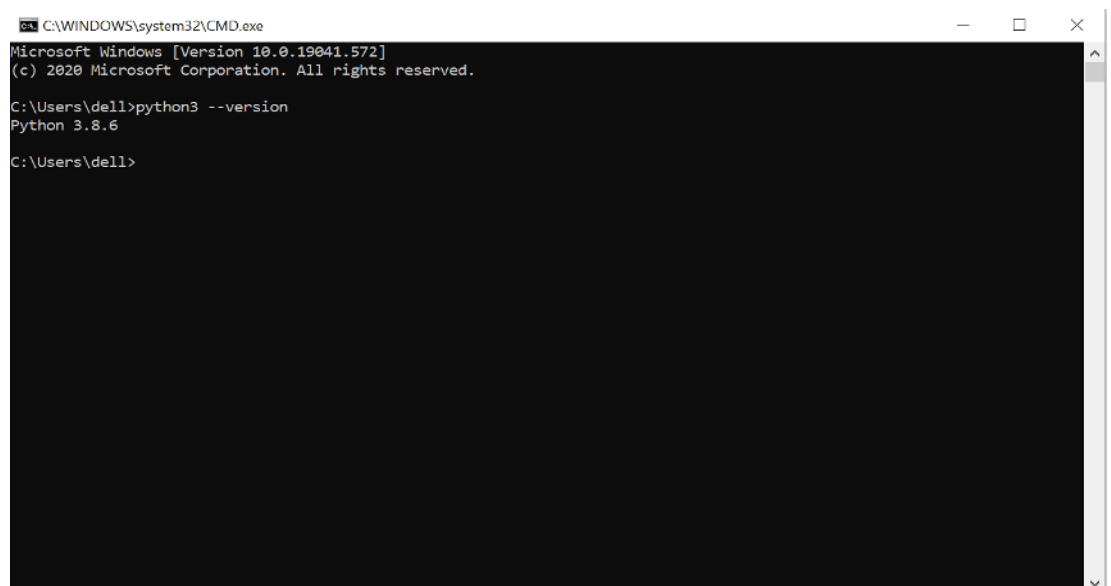
Figure 2.2: Testing python version.

11

IF you did NOT find a version, then type this command :**pip3 install -U pip**

- **Virtual Environment Setup**

  Create and activate the virtual environment using the below commands.

  python3 -m venv ./venv

  Activate the virtual environment:

  .\venv\Scripts\activate
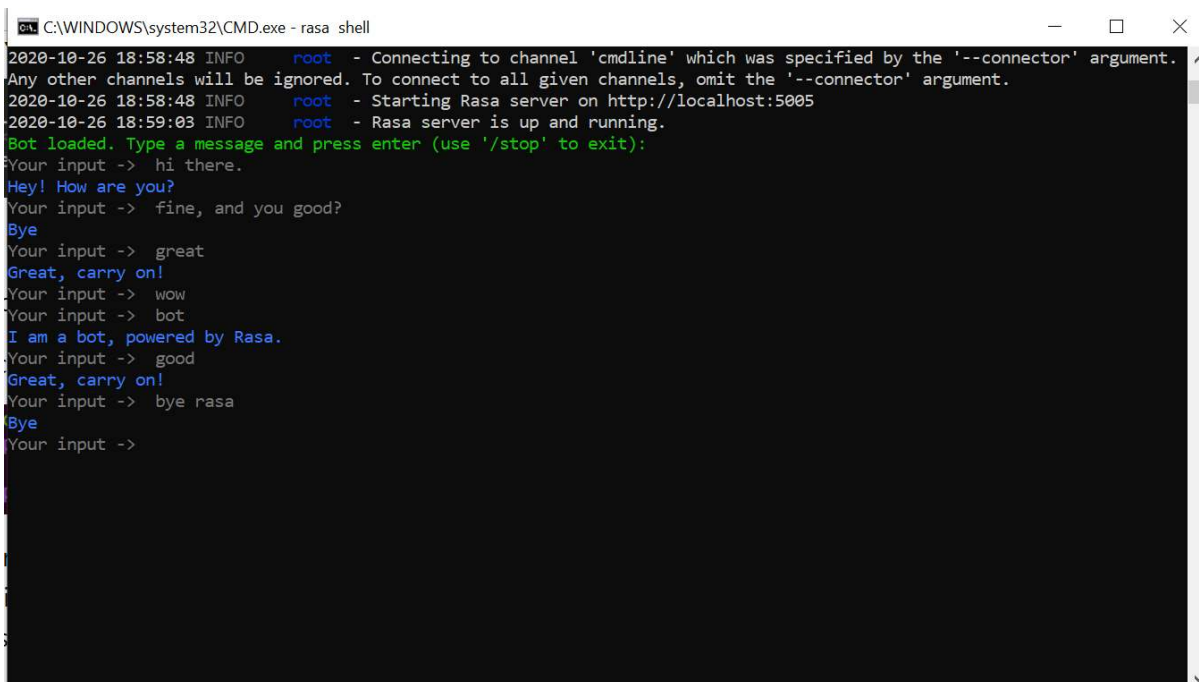
- Install Rasa open-source with the below command:

  pip install rasa

- make a directory and move to it

  mkdir test-chatbot && cd test-chatbot

Create the new project with **rasa init** command and start the conversation with the initial demo chatbot.

Figure 2.3: Rasa bot initial chatting.

2.8   supportive information:

After a great effort in searching and exploring personalities related to the topic of chatbot and natural language processing (NLU), we want to thank Mr. Jonathan Wheat, who is a certified developer for a framework Rasa for all his efforts and time to communicate with us and to provide wonderful, useful and inspiring answers about several Questions we asked him about Rasa and NLU. The following is the information that he provided us.

12

Our Question: What are the actual parts that are required and needed to be programmed for chatbot using Rasa Framework ?.

  " There are 3 major parts to the Rasa Framework:

- The Rasa Server: This is the actual software that runs and interprets the user's input and handles the responses based on a trained model (more on that below).

- The Action Sever: This handles some of the complex form logic if you need it, and/or API calls, etc. This is written in Python, and there are many examples in the Rasa repo to look at. Depending on how complex you need your bot to be, you may or may not need to write anything for this.

- The chat interface - This is not part of the normal Rasa framework, you'll probably want to grab one of the many chat interfaces, and/or connect Rasa to FaceBook Messenger, Slack, Telegram or other service. Rasa has connectors for these major services to allow you to interact with your bot using commercial chat interfaces."[4]

 Training data.

"The training data for Rasa 2.0 is all formatted as .yml files. There are 3 types of training data.

- There is the main domain.yml file that contains a list of intents, entities, slots, responses and some actions/form configuration - these equate to what the bot thinks you mean (intent), the text the bot responds with (responses), the bots "memory" (slots and entities), and some setup if you're using the Action server above.

- The next file is an nlu.yml file - this is your training data, it lists all of your intents and the text, words, phrases that the bot can learn from and interpret as that specific intent. The easiest example is a "greet" intent. That would have things like - hi, hello, how are you, what's up?, how are you doing? as the training data and Rasa will process all that when you train.

- The last part of the training data is the stories.yml file - these are actual story flows that start with an intent and you script how you want the conversation to flow. These are fluid, they're not super strict "wired up" flows, but they help teach the bot how a conversation should look. If done properly, it will handle tangents (someone talking about one topic and switching to another and the bot will keep track). You can, and should, have as many stories as possible, different permutations and flows to give Rasa the best idea of a conversation. It'll surprise you with how it works, and will (again if done properly with enough data) converse very well.

    To train the bot, it is as simple as typing - rasa train. and getting a cup of coffee or a beer depending on how big your bot is. In the beginning it'll train pretty quickly, but as you add more and more data, it'll take longer and longer. I have one that takes over an hour to train because of the data involved."[4]

Our Question: Is there any majority of advanced python code that I need to write?, or are there ready-made components and all I have to do is use them ?

"As mentioned above, the only python you'll need is for complex logic and any API calls (ie - getting the weather, or traffic or server status, etc). When I say complex logic, it is possible to set up a strict flow to questions (think form fields) for the bot to ask. You can branch depending on answers and ask follow-up questions based on previous answers. This is all written in Python. You don't need any specialized python skills, I don't think I've done anything outside of my basic python knowledge. It is mostly python Classes, functions, etc, and there is a specific pattern you need to follow to get a FormAction to work properly, so there is a lot of "boilerplate" code if that makes sense."[4]

Our Question: If I have to build a chatbot from scratch, which algorithms should I use and what skills and information should I have about NLP ?, and what components do I need to achieve my work ?.

"Wow, it all depends on how robust you want to make it. You'll definitely need to level up on your Python and/or C, C++. It also depends on what you want the bot to do, there are some simple classification algorithms that could map an intent with your phrase(s) and then when a new phrase comes in, it could, with some confidence, predict what intent it maps to, and then have the bot respond. You could implement a SVM (Support Vector machine) to do something like this, but for me that's WAY too much effort, and I don't have a doctorate in Machine Learning, or Data Science. Rasa has it all there for you. What's great is that if there is some cool new algorithm, policy or pipeline you want to use, you can add it into Rasa and use it. I don't want to discourage you from rolling your own, but there's a lot to it. There are tutorials out there if that's what you want to do though."[4]

Our Question: the tutorials and outer resources that can help me get the work done and overcome this challenge?.

"Rasa has a bunch of examples of bots in their gitHub repo as well as a few video series (Rasa Master Class, Algorithm Whiteboard and Breakout Diaries) on YouTube - all free. They also host webinars and (pre-covid) had a chatbot conference, this year it was online - I think most of those talks are on youtube as well."[4]

Chapter 3: System Requirements

3.1 Requirements:

- **Functional Requirements:**

  1. The system must provide clear information about Admission policy.
  2. The system must provide clear and fully detailed information about university colleges.
  3. The system must provide clear and fully detailed information about colleges' programs.
  4. The system must provide clear and fully detailed information about colleges' majors.
  5. The system should clarify information about the permitted secondary school branch for each major.
  6. The system should clarify the minimum GPA in high school for each major.
  7. The system should clarify the duration of study for each major.
  8. The system should clarify the parallel study policy for each major.
  9. The system should provide the graduation plans for each major.
  10. The system should provide information about placement tests.
  11. The system should provide information about first installment costs for each major and the credit hour price.

- **Non-Functional :**

  1. The system shall handle multiple users inputs, If two or more students are chatting with the bot, none of the students has to wait too long to be answered by the system.
  2. The bot should have a delay in response, to let the student feel like he/she is talking to a human instead of a bot. A little late response from the chatbot makes the student feel as though he is talking to a human.
  3. The system should have the appropriate data set. The correct data set is the basis for the chatbot, when the data set is correct and tuned, the chatbot will be trained on it to give the best possible result.
  4. The system should have Data Training. Data training mainly depends on the content targeted at Admission and Registration deanship.
  5. The system should prevent abusive language.
  6. The system should be used on http://reg.ppu.edu/ website.
  7. Ability to extend the project to include all colleges.
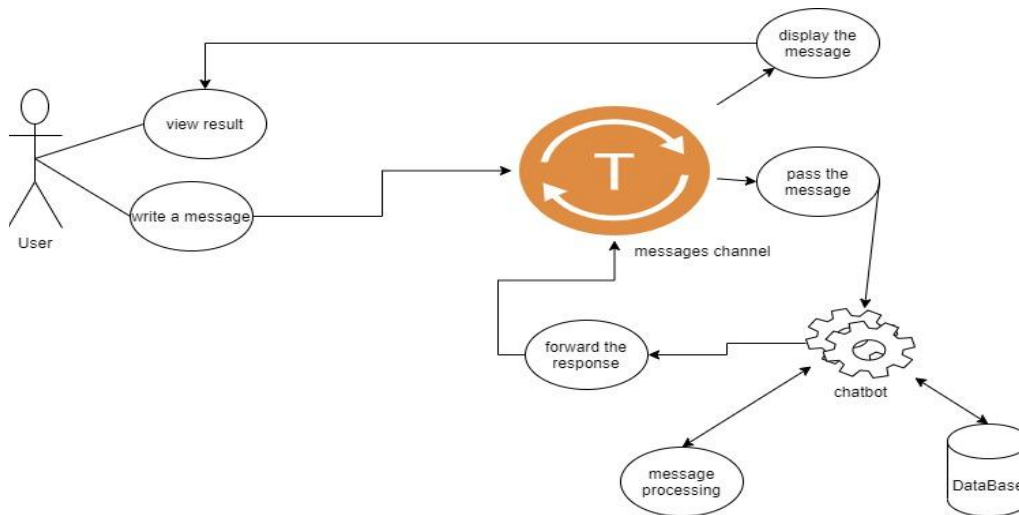
Figure 3.1: Use case diagram.

3.2 Class Diagram and use case diagram

In this part will explain the class diagram that is used to describe the system classes and relation between them.
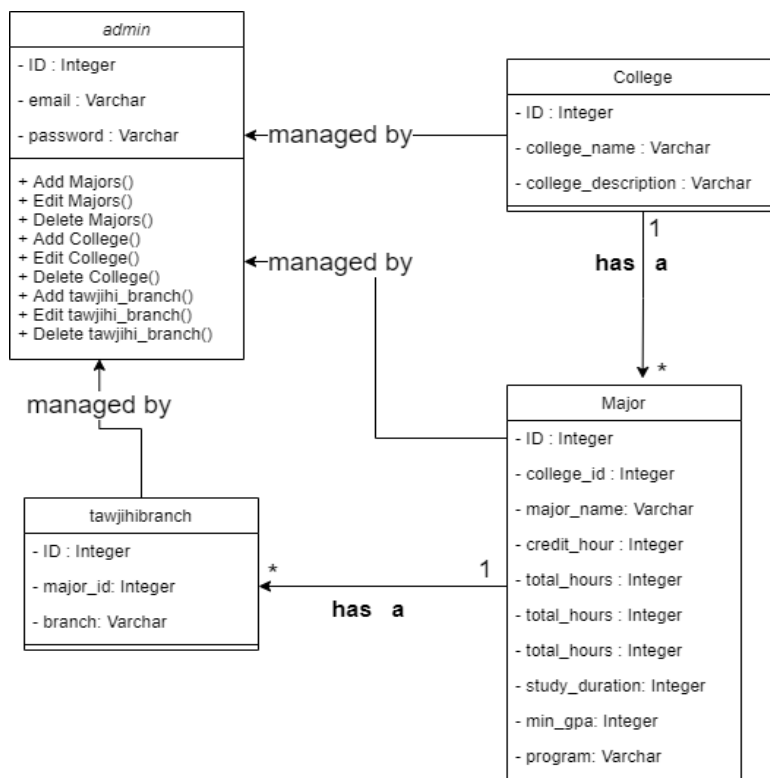

Figure 3.2: Class diagram

3.3 Description and analysis system requirements:

Function name:	The system must provide clear information about Admission policy.
Main Actor:	System.
Aim:	Clarify the university's policies regarding student admission
Condition:	- Have an internet connection. - opening university website for registration.
Measures:	- open registration website and ask about acceptance of Palestinian high school students. - open registration website and ask about acceptance of non-palestinian high school students.
Exceptions:	- Provide information about students how are not in high school.
Solution Exceptions:	- Ask the user to give information about if the student is in another university and switch to " Students of other universities wishing to transfer " selection

Table 3.1: Handling Admission policy table.

Function name:	The system answers shall be sent as text messages.
Main Actor:	People.
Aim:	control the structure of the conversation.
Condition:	- Have a internet connection. - Opening Registration website. - Chat with the system.
Measures:	- open Registration website and start chatting.
Exceptions:	- voice messages.
Solution Exceptions:	- show text message to tell the user that the system does not support this feature.

Table 3.2: Answers form table.

Function name:	The bot should ask for more simplification when the questions are not clear.
Main Actor:	Student.
Aim:	reach specific information.
Condition:	- Have a internet connection. - Login to his Facebook Account. - start chatting. - unclear question or syntax error.
Measures:	- unclear question or syntax error. - ask more question depends the closest intents.
Exceptions:	- Ambiguity.
Solution Exceptions:	- ask more question for the closest intents.

Table 3.3: Unclear question handling table.

Function name:	The admin add new major to college
Main Actor:	admin.
Condition:	- Have a internet connection. - Login to the administrator..
Measures:	- after the condition can add new major by press to add button in major page.
Exceptions:	- there are similar major already existed.
Solution Exceptions:	- warn the admin the same data was found.

Table 3.4: Adding new major

Function name:	The admin edit existed major's data
Main Actor:	admin.
Condition:	- Have a internet connection. - Login to the administrator..
Measures:	- after the condition can edit the major's data by press the edit button in major page.
Exceptions:	- there are similar major already existed.
Solution Exceptions:	- warn the admin that data types are not compatible - warn the admin about empty boxes to be filled.

Table 3.5: Editing major data

Function name:	The admin delete major from college
Main Actor:	admin.
Condition:	- Have a internet connection. - Login to the administrator..
Measures:	- after the condition can delete majors by press delete button in major page.
Exceptions:	
Solution Exceptions:	

Table 3.6: Delete major

18

Function name:	The admin add college to colleges
Main Actor:	admin.
Condition:	- Have a internet connection. - Login to the administrator..
Measures:	- after the condition can add new college by press add button in college page.
Exceptions:	- Similar college name
Solution Exceptions:	- warn the admin the same data found

Table 3.7: Add new college

Function name:	The admin edit existed college's data
Main Actor:	admin.
Condition:	- Have a internet connection. - Login to the administrator..
Measures:	- after the condition can edit the college's data by press the edit button in college page.
Exceptions:	- there are similar college already existed.
Solution Exceptions:	- warn the admin that the name is already used.

Table 3.8: Editing existing college

Function name:	The admin delete college
Main Actor:	admin.
Condition:	- Have a internet connection. - Login to the administrator..
Measures:	- after the condition can delete college by press delete button in college page.
Exceptions:	- college can't be deleted because of its related majors.
Solution Exceptions:	- warn the admin to delete the majors then delete the college

Table 3.9: Delete college

19

Function name:	The admin add new tawjihi branch to tawjihi branches
Main Actor:	admin.
Condition:	- Have a internet connection. - Login to the administrator..
Measures:	- after the condition can add new branch by press to add button in tawjihi branches page.
Exceptions:	- there are similar branches already existed.
Solution Exceptions:	- warn the admin the same data was found.

Table 3.10: Add new tawjihi branch

Function name:	The admin delete tawjihi branch
Main Actor:	admin.
Condition:	- Have a internet connection. - Login to the administrator..
Measures:	- after the condition can delete branch by press delete button in tawjihi branches page.
Exceptions:	
Solution Exceptions:	

Table 3.11: Add new tawjihi branch

20

Chapter 4: Models and interfaces

4.1   Basics Of Rasa Open Source Conversational AI:

The diagram below shows the general rasa open source conversational associated with machine learning. [5]
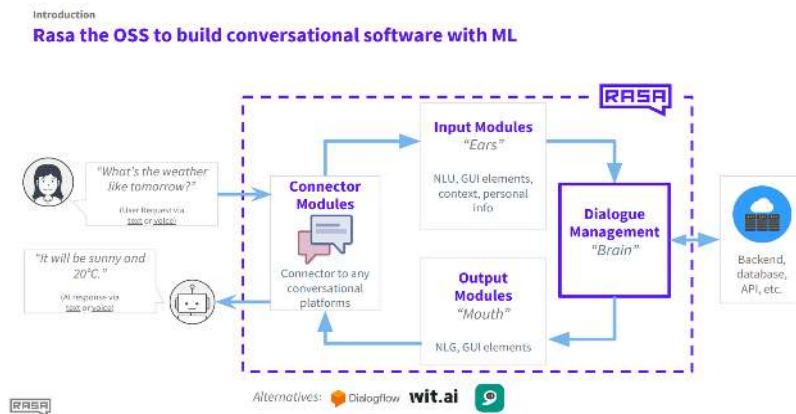
Figure 4.1. Basics Of Rasa Open Source Conversational with ML .

4.2   Architectural model :

"The diagram below provides an overview of the Rasa Open Source architecture. The two primary components are Natural Language Understanding (NLU) and dialogue management.

NLU is the part that handles intent classification, entity extraction, and response retrieval. It's shown below as the NLU Pipeline because it processes user utterances using an NLU model that is generated by the trained pipeline. The dialogue management component decides the next action in a conversation based on the context. This is displayed as the Dialogue Policies in the diagram."[6]

Figure 4.2: Architectural model.

4.3   Rasa framework files:

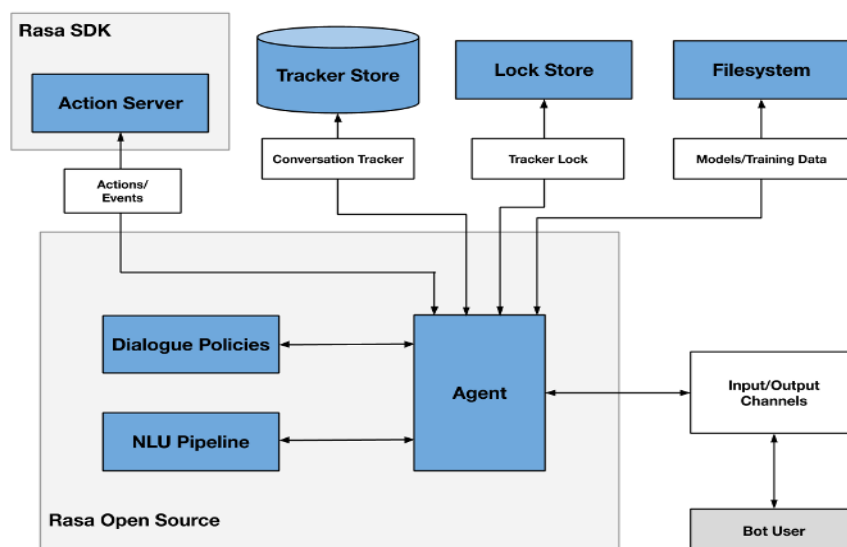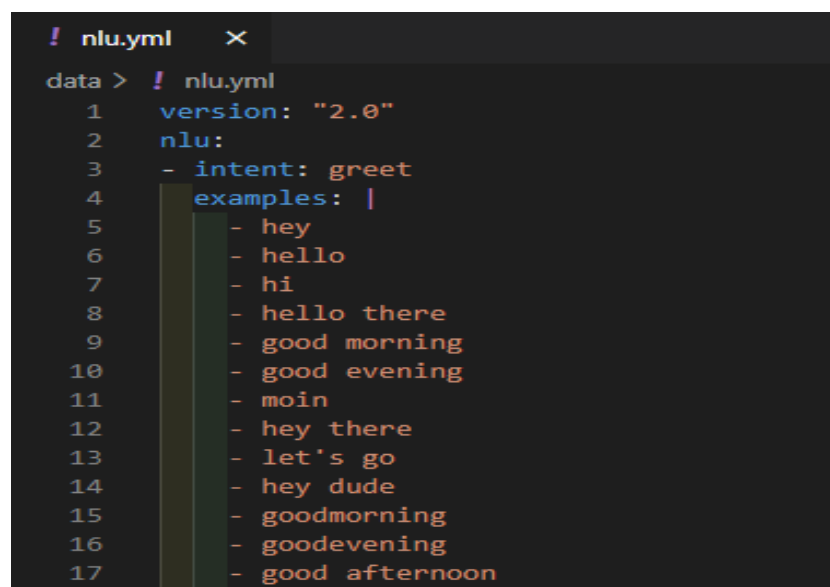At the moment of writing this report, we have been using the latest version of the Rasa framework v2.x and in this release the existing file structure and file extensions have changed from previous versions. Dealing with the framework changed and became better and smoother by using files with .yml extensions.

- nlu.yml:
  - **Intents**: The file contains the Natural language understanding (NLU) model training examples. This includes intents, which are user goals, and example utterances that represent those intents. The NLU training data also labels the entities, or important keywords, the assistant should extract from the example utterance.
  - **Entities**: The important keywords that an assistant should take note of. For example, the message 'My name is Maher' has the name 'Maher' in it. An assistant should extract the name and remember it throughout the conversation to keep the interaction natural.
  - **Utterances**: Anything a user says. A single utterance is an entire sentence passed as input to the Chatbot to intent. By definition, an utterance holds an intent and could potentially include one or more entities. During each transaction with the Chatbot, an utterance is sent to the Chatbot's understanding unit to parse and interpret.


```
! nlu.yml    ×

data > ! nlu.yml
   1   version: "2.0"
   2   nlu:
   3   - intent: greet
   4     examples: |
   5         - hey
   6         - hello
   7         - hi
   8         - hello there
   9         - good morning
  10         - good evening
  11         - moin
  12         - hey there
  13         - let's go
  14         - hey dude
  15         - goodmorning
  16         - goodevening
  17         - good afternoon
```

Figure 4.3. Nlu.yml file content.

22

- Domain.yml: Defines the environment in which the assistant operates, including:
  - What the user means: specifically, what intents and entities the model can understand.
  - What responses the model can provide: such as utterances or custom actions.
  - What to say next: what the model should be ready to respond with.
  - What info to remember: what information an assistant should remember and use throughout the conversation.

Figure 4.4. Domain and nlu files.

- data/Stories.yml: example end-to-end conversations. A transcription of a conversation between a user and a Chatbot, where user inputs are expressed as corresponding intents (and entities where necessary), and the Chatbot's responses are expressed as corresponding action names.

Figure 4.5: Domain and stories files.

- actions/actions.py: all programming is here. The action class describes how the Chatbot fulfills various intents. Each action has access to the tracker, which contains data about the conversation, and implements a run function which holds the logic for carrying out said action.
- Tracker: A tracker object maintains the current state of the conversations. It keeps track of the events that have happened so far, such as utterances and actions, as well as other data such as the slots and entities.
- Slots: Slots are variables a Chatbot requires to perform a specific task. Slots are essential to interpret a user's input and adequately execute the action. Slots are commonly filled using Entities. Slots serve as the building block for a Chatbot's context manager
- processing pipeline/Config.yml: The configuration file defines the components and policies that the model will use to make predictions based on user input.
- Endpoints.yml: A web server that reacts to the call, runs the code, and optionally, returns information to modify the dialogue state. The full configuration of the custom action server is provided in this file.
- Credentials.yml: This file holds the configuration details for connecting to external applications, like messaging channels.

4.4   State and Sequence diagrams:

In the following figure, we will show the state diagram and what is the state of the data entry process by the user and where it passes.

In the beginning, when the user enters a question through the Facebook Messenger platform, this question moves via Facebook Messenger API's to the section of classifying intens and starting the process of extracting the necessary entities. In the event that not enough words expressing intents are revealed; The bot sends a message to the user asking for more information. When the bot gets enough information to reveal intents and entities, it sends the available data to a special section called Actions.

In the Actions section, the bot specifies the appropriate response to be sent to the user after performing matching operations for extracted entities based on the highest corresponding percentage of these entities. After that, the bot sends the resulting information and shows it as a message via the Facebook Messenger platform to the user.

In the figure below, we'll show the sequence diagram. In this diagram, we will carry out a complete tracking process for all stages of the system, how it handles and processes user input.

In the beginning, the user enters the data through the Facebook Messenger platform and is sent via Facebook Messenger API's to the Rasa framework, which will perform several operations to process the available data. Where the framework sends the data to the Rasa NLU file, where both the classification of intentions and the extraction of entities from the sent data are performed in this file, where the data is divided into words and from these divided words the intent is extracted as these intentions are already predefined words Inside the Rasa NLU file. For example, if we ask what's the weather like tomorrow? When sending this question, the framework identifies the intentions through some of the key words in the question, and here the key word is weather. Then they are recognized and likewise the entities are extracted which are tomorrow in this case.

After that, the framework sends the classified and extracted intentions and entities to Rasa Core, to determine the appropriate actions and send the appropriate response to the user via the Facebook Messenger platform.
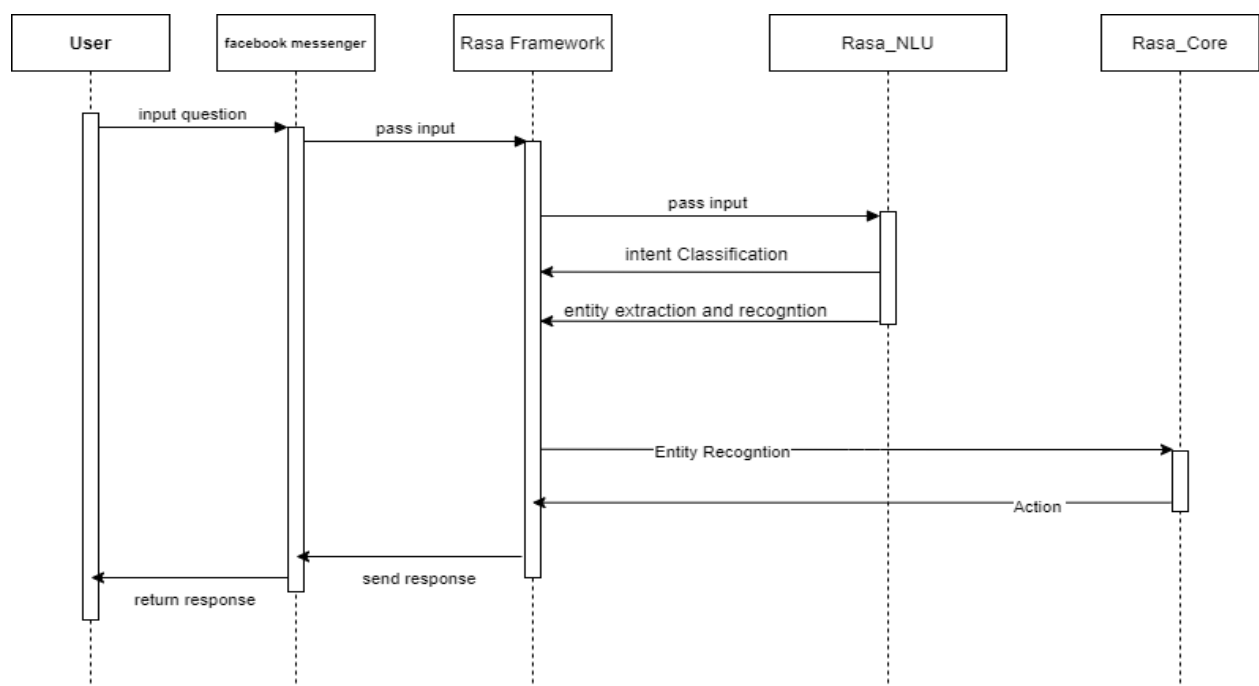
Figure 4.6: Sequence diagram.
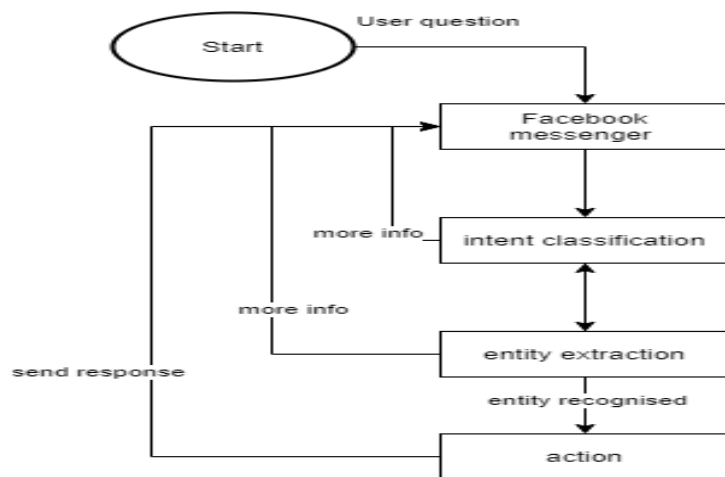
Figure 4.7: State diagram.

4.5   Rasa NLU:

The NLU module is answerable for concluding all language parsing tasks. The foremost important of those tasks are intent detection and named entity recognition. The implementation of the NLU module is modeled as a pipeline that processes the input text in consecutive steps called components. So, it's a crucial thing to grasp the components' lifecycle and the way they interact with one another. The figure below shows the Rasa NLU components' lifecycle. Before starting the pipeline, a context object is passed to among components in order that they can dissipate information. This object allows
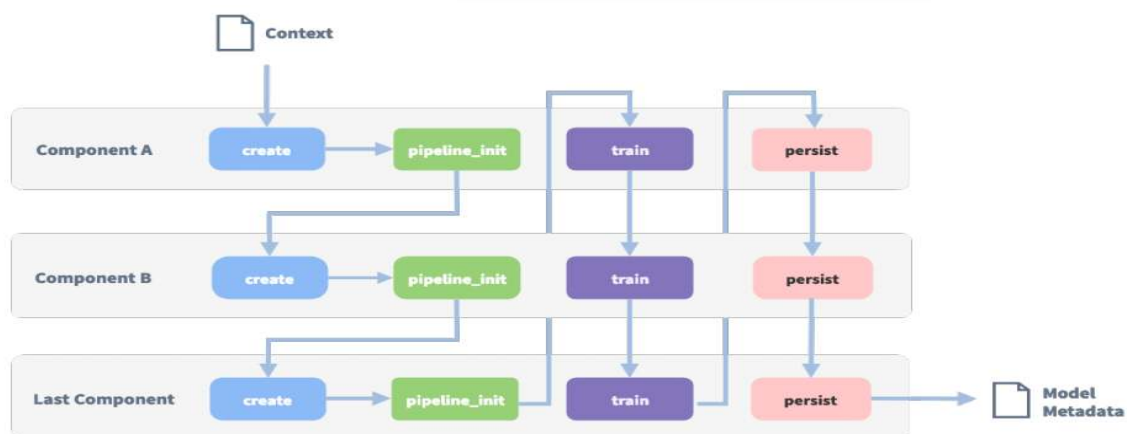
Figure 4.8: Component lifecycle.

the output of 1 component to be used because of the input of the text.

In order to do intent classification, the next has to be pre-processed first into a format that is more machine-learning friendly. "The most widely used method in literature is the word vector embedding. This method represents words as vectors in a high dimensional space with the proposition that semantically similar words would fall closer to each other in terms of distance in this high dimensional space. Thus, the first step of the NLU pipeline is to tokenize the input.

26

Following that, the input is passed to a regex featurizer which extracts features that match predefined regular expressions, such as dates and numbers, as part of a simple entity detection algorithm. The next component in the pipeline runs a Conditional Random Field model for entity extraction based on Scikit Learn. The next component maps the extracted entities to their synonyms provided by a training file. The following component converts the data to a bag of words, suitable for intent classification. The final component of the pipeline does intent classification using a model based on StarSpace."[7]

4.6   Rasa Core :Dialogue Handling/Policies:

Rasa core is used to implement the dialogue handling component to predict which action to be executed and update the state for this action and give the output as well as used for next input.

Figure 4.9: Rasa core dialog handling.

4.7   User Interface:

Asking about enrollment and payments.

Figure 4.10: chatting with bot Q1.

- Asking about majors and view specific major information

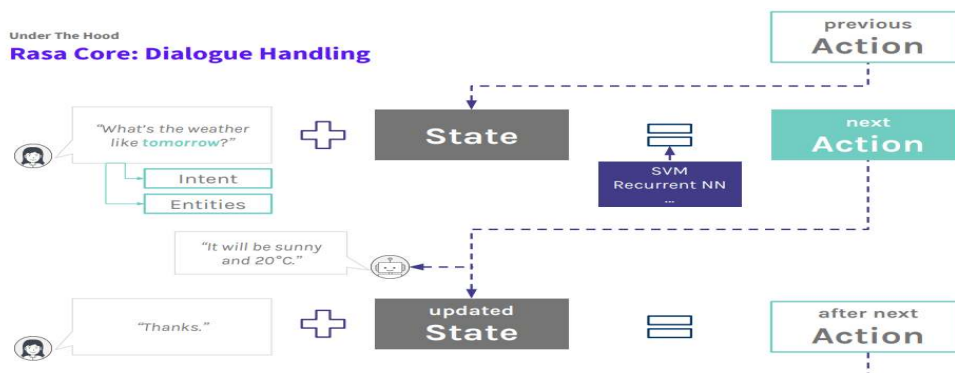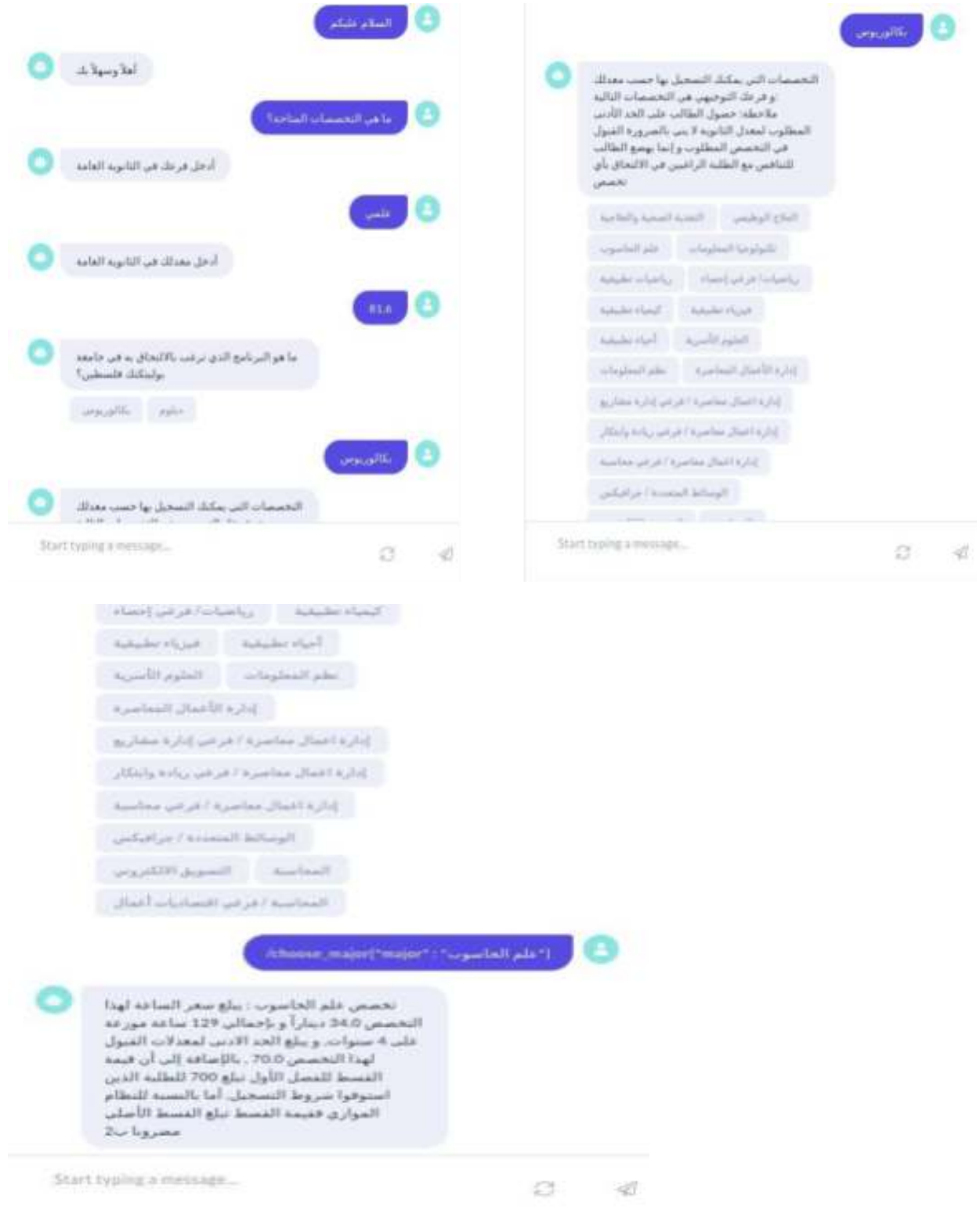

Figure 4.11: chatting with bot Q2.

28

Chapter five: System demonstration

5.1.   Admin panel:

5.1.1.   Implementation details: This interface has been developed so that those involved in the Deanship of Admission and Registration at Palestine Polytechnic University add, amend or delete what is necessary regarding each of the colleges in the university in all cases. The same applies to the majors offered by each college as soon as the decision is made to establish a new specialization, change the characteristics of one of the pre-existing specializations, or cancel a specialization in itself. And also concerning the Tawjihi branches approved by the Ministry of Education, it is possible to add or delete what is necessary from these branches as necessary.

5.1.2.   Implementation issues: We did not face many problems while developing this panel due to our prior knowledge and ability to develop websites using this combination of PHP with HTML, CSS, Bootstrap, and a little Javascript. However, some of the problems that we encountered made us go through a difficult time sometimes, which is also difficult to track, because we did not use a framework to implement this panel and no error log is referred to when encountering some problems.

5.1.3.   Snapshots and and UI description:

- **Login:** This page is the first page to be shown when accessing the site
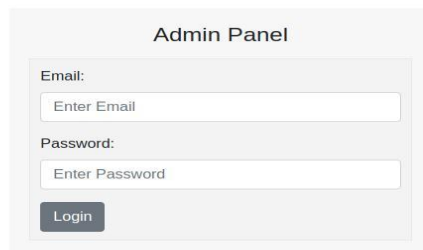
Figure 5.1:Login screen.

- **Dashboard:** This page will be the first thing that appears to the user when he logs into the site correctly, whether he is an administrator or a viewer.
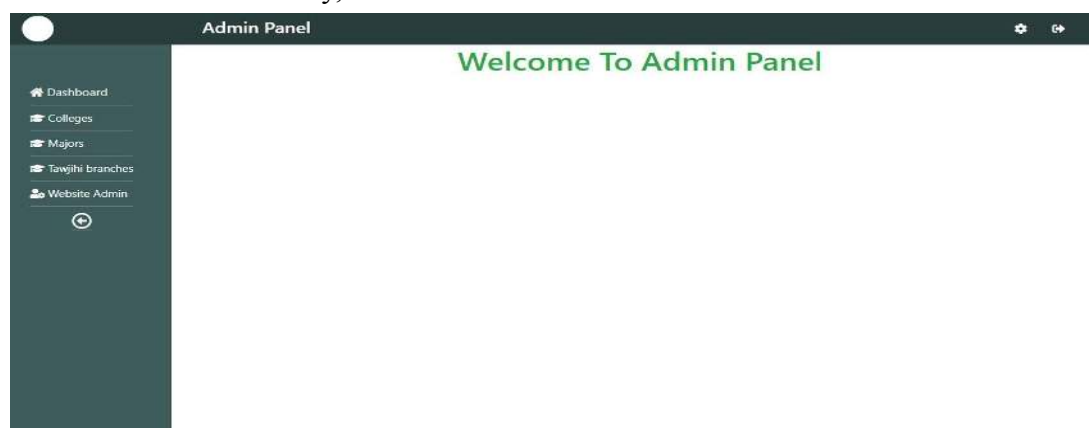
Figure 5.2: Dashboard screen.

- **Colleges menu:** This page displays all the colleges in the university, and it appears for both the viewer and the administrator. The administrator can add a new college, remove a pre-existing college, or modify pre-existing college data.

Figure 5.3: college menu screen.

- **College data:** This page displays the fields assigned to each college, name, and description. Only an administrator can access this page.

Figure 5.4: college add/edit screen.

- **Majors menu:** On this page, all the majors that the student can study on the campus of Palestine Polytechnic University are displayed, and both the viewer and the administrator can access this page.
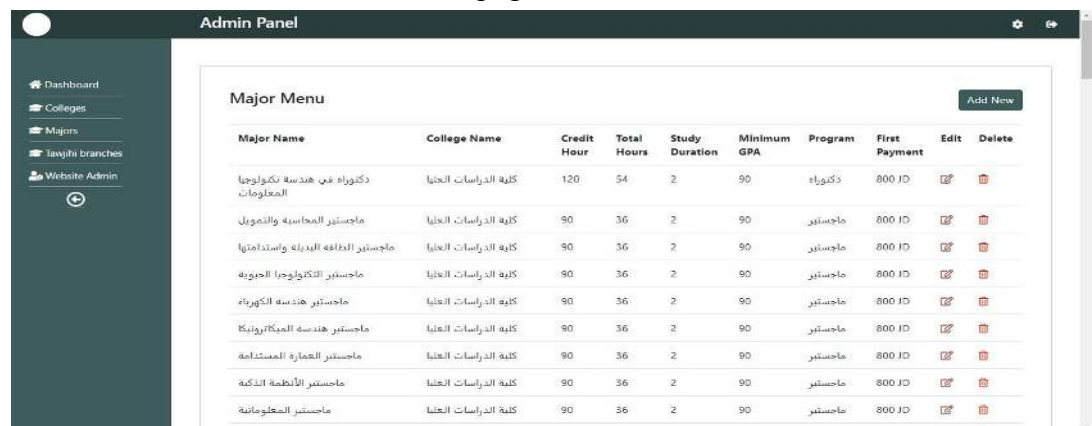
Figure 5.5: major menu screen.

- **Major data:** Only administrators can view this page, this page contains all necessary fields for majors at Palestine Polytechnic University. Starting with the title of the major to the program attached to this major through each of the college supervising this specialization and each of the total credit hours, the hourly rate specified in JOD, the value of the first installment, minimum GPA, duration of the assigned study and the approved guideline branches for each major.

Figure 5.6: major add/edit screen.

- **Branch menu:** Here all the branches of the Tawjihi that its students can enroll in the colleges' majors are displayed. Can be viewed by the viewer and administrator.

Figure 5.7: branch menu screen.

- **Admin page:** All users of the site are shown on this page. Both the viewer and the administrator can access this page.

Figure 5.8: admin page screen.

31

- **Admin data:** On this page, special data is added to create a new user for the site. Only an administrator can access this page.

Figure 5.9: admin add screen.

5.1.4. **Important information:** As a summary, this panel gives the administrator full access to all the available data regarding each of the colleges, majors, and Tawjihi branches, with the possibility of adding, deleting or modifying them. With the possibility of adding new users or deleting previous users.

5.2. Chatbot
5.2.1. Implementation details:
The project consists of several parts, the most important of which are:

1- Actions section, which is the folder that contains 3 files in the Python programming language.The most important file is the Actions file, which contains custom Actions that are built according to the need and purpose of the chatbot.

2- The data section and contains 3 important files that cannot be dispensed with: the natural language understanding file, which contains the training data necessary for the bot, which it is expected to receive during its operation from the user, and the rules file, which contains a certain structure that makes the bot act obligatory according to what exists, regardless of the circumstances in terms of the received data, and the story file, which contains scenarios of conversations with users, and all conversations are recorded within this file in .yml extension.

3- There is also a very important file, which is the domain that defines the universe in which your assistant operates. It specifies the intents, entities, slots, responses, forms, and actions your bot should know about. It also defines a configuration for conversation sessions.

4- There are also configuration files, endpoints, and credentials that are responsible for the overall properties and some permissions allowed for the bot and linking with chat channels such as Slack and Facebook Messenger.

5- There is also the Models section, in which all models are stored after each bot training. Every model we can say is like the nucleus or brain of the bot. The bot cannot work and listen to the user's messages and respond to them without the model.

6- After each bot training process, and to get the latest results, you must choose the newest model. Older models can be selected so that they can be compared with the new model in terms of additions.

5.2.2. Implementation issues: We faced many problems. In each stage we passed, we encountered many problems, some of which took days to solve. We will mention the most important problems that we encountered and stood in our way:

1- The problem of installing the framework, during the installation on Windows we did not encounter any serious problems except for the problem of installing Rasa X, which is the GUI. The last version was not very compatible with Windows at the time, so we had to use one of the old versions so that we could work on Rasa X.

2- Installing the framework on Linux was one of the biggest problems we faced, and the problem was divided into smaller problems: the issue of the Python programming language version, the problem of pip packages, and the problem of environment variables and dependencies.

3- We also encountered problems with the actions, especially that the file was sensitive to the issue of functions' names and how to deal with it and not respond except by relying on the correct and acceptable naming rules.

4- The biggest problem we faced was the problem of integrity error while working with Rasa X and its connection with the Action Server on Windows. The problem revolved around the inability to record or respond to commands and conversations between the user and the bot in the dedicated database. We repeatedly searched, asked, and investigated this problem that has not been resolved at the time of writing this report, as it was a problem related to the Windows operating system. We did not encounter the problem while running the bot on Linux.

5- One of the problems that we faced was about slots and not filling them out according to the rules in the Domain file. While talking to the bot, some answers are extracted and processed, and responded to according to what was extracted and we encountered the problem on Windows.

6- The bot training process is one of the most frequent problems in terms of time and space, so every training process for the bot requires at least 4 minutes at least. After completing the training process, a model with a size of more than 20 MB will be produced. And you must activate the latest model after each training process.

7- Not all buttons appear when the bot is connected to chat channels such as Slack and Facebook Messenger.

8- Uploading the bot to Docker was one of the most difficult problems for us due to our lack of knowledge of Docker and the mechanism of uploading projects to it and the necessity of using Linux.

5.2.3. Snapshots and UI description:
- Rasa training log screen:

Figure 5.10: rasa training log screen.

- Rasa actions server terminal screen:

Rasa action server runs custom actions for a Rasa Open Source conversational assistant. When your assistant predicts a custom action, the Rasa server sends a POST request to the action server with a json payload including the name of the predicted action, the conversation ID, the contents of the tracker and the contents of the domain. When the action server finishes running a custom action, it returns a json payload of responses and events. See the API spec for details about the request and response payloads. The Rasa server then returns the responses to the user and adds the events to the conversation tracker.

Figure 5.11: rasa action server screen.

- Rasa X terminal screen:

Figure 5.12: rasa x screen.

34

5.2.4. Important information:

- When training the bot, it is important to make sure that the structure in the domain file is in the right format or it will fail.
- Training time is machine dependent.
- It's ok to delete old models, but it's not recommended.
- Intents in the NLU file have to contain at least two training examples. One training example may lead to training failures.
- Intents have to be part of the domain. If not, it may lead to issues in the conversations.
- Any change in the actions file requires to stop the action server and restart it again by typing *rasa run actions* from the terminal ONLY to acquire the changes.
- Any change in nlu, rules, stories, config, credentials, domain, and endpoints files requires to stop the bot and restart it again in case of using terminal. In case of using Rasa X GUI, the bot won't stop or need to be restarted.
- In case Rasa X was stopped, the ONLY way to start it again is using the terminal by typing *rasa x*.
- User needs a password to have access to the internal Rasa X GUI, it is automatically generated in every attempt to start Rasa X from the terminal.
- Rasa X password is not changed when attempting to train the bot using the GUI.
- It's ok to have a User Warning about the model confidence being set to softmax instead of linear norm. Rasa framework developers recommend the softmax instead of the linear norm.

Chapter 6: Testing

6.1. Testing criteria:

6.1.1. Chatbots testing criterias:

      Criteria derived from the Rasa framework, which is a set of commands that are executed to obtain degrading results based on the contents and how the bot is built. It is done by executing specific commands, data validation verifies that no major errors or inconsistencies appear in your domain, NLU data, or story data. If data validation leads to errors, training the model may also fail or result in poor performance, so it is always a good idea to do this check before training the model. Including the --fail-on-warnings flag, this step will fail warnings that indicate more minor issues.

      Another thing you can do to get the bot to be tested is to write test stories. Testing your trained model on test stories is the best way to be confident in how your assistant will behave in certain situations. Test stories, written in a modified story format, allow you to present complete conversations and test that, given certain user input, your model will behave expectedly. This is especially important when you start to present more complex stories than user conversations. Test stories are similar to the stories in your training data but include the user's message as well.

Conversation flow: in this criteria, the following things should be checked while testing:
- Does a chatbot clearly understand the questions?
- Does it always give instant responses to these questions?
- Are the answers relevant to the given questions?
- Should a user ask a series of questions until he gets the answer?
- Does a chatbot engage the user to continue the conversation?[9]

Business-specific questions: "Each industry has its specific terminology, notions, nuances. So, a tester should have a list of domain-specific questions to check if the chatbot is able to answer those."[9]

Confusion handling: "Confusion may arise if a user enters some expression with double meaning or an unknown word for a chatbot. The latter should be taught to answer in such a situation. The tester's goal is to check if the chatbot can handle misunderstandings, exceptional conversational scenarios, and unusual patterns. This capability of a chatbot is showing how "emotionally intelligent" it is."[9]

6.1.2.
The classic criteria in the software test. By testing the software at and around critical points, and what it accepts and what it does not accept. The software is also presented to a group of people to be tested and examined based on interactions and data with humans.

6.2. Test plan

6.2.1. **Features to be tested:** Tested features will be those features that the bot has been trained on and given all the necessary information and data about these features.

User input/intent	Bot action
admission_policy	utter_admission_policy
choose_college	utter_choose_college
program	utter_program
expected_major	expected_major_form
parallet_study	utter_parallel_study
placement_tests	utter_placement_tests
first_installment	utter_first_installment

Table 14: Tested features

6.2.2. **Features not to be tested:**
- Feature of return and change the value of the entered mark.
- The feature of showing a welcome message to the user when the chat window is opened.
- The feature of changing the intended intent of a previous message.

6.2.3. **Testing environment and tools description:**

Facebook Messenger and Slack were used as test environments for the bot but were abandoned for reasons mentioned in the challenges section. Therefore, the testing of the bot was completed on the local host environment by hosting the bot using NG Rock software, which creates a link that enables users outside the local environment to access and interact with the bot.

Not many testing tools were used, so to speak, a set of commands dedicated to bot testing were used by the developers of the framework, namely:
- rasa test.
- rasa data validate.
- rasa test nlu.
- rasa data split nlu.

6.2.4. Test case diagram:

This test case is a simulation of a segment of a conversation between a user and the bot that satisfies the use case model in Chapter 3. This short conversation consists of messages sent by the user, texts, and questions that the bot replied.

The following flow chart shows in detail what is being processed during this short conversation, given that the diagram is very large, we have divided it into two parts to facilitate the process of follow-up and scrutiny.

Part 1
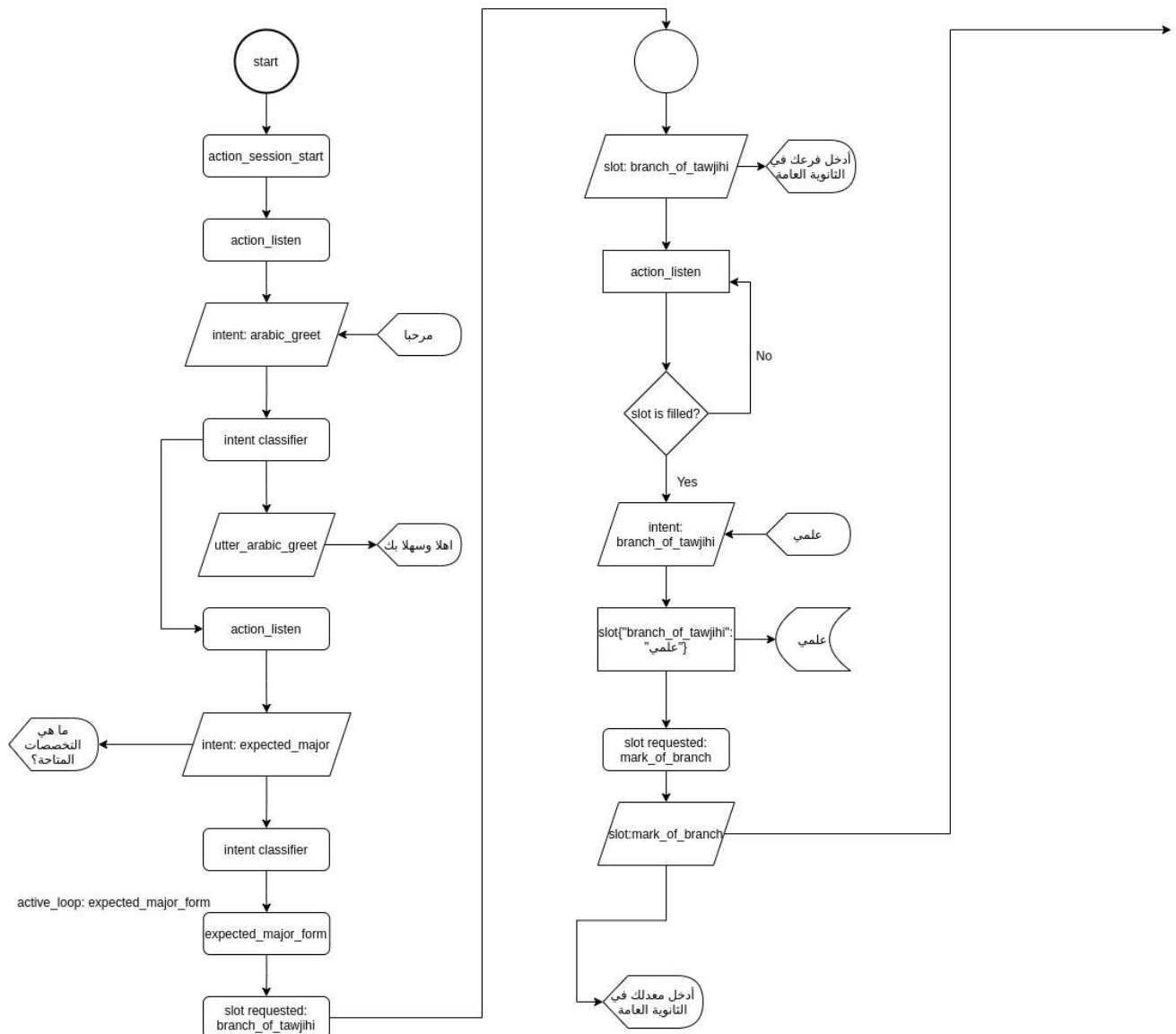
Figure 6.1: test use case part I

Part 2
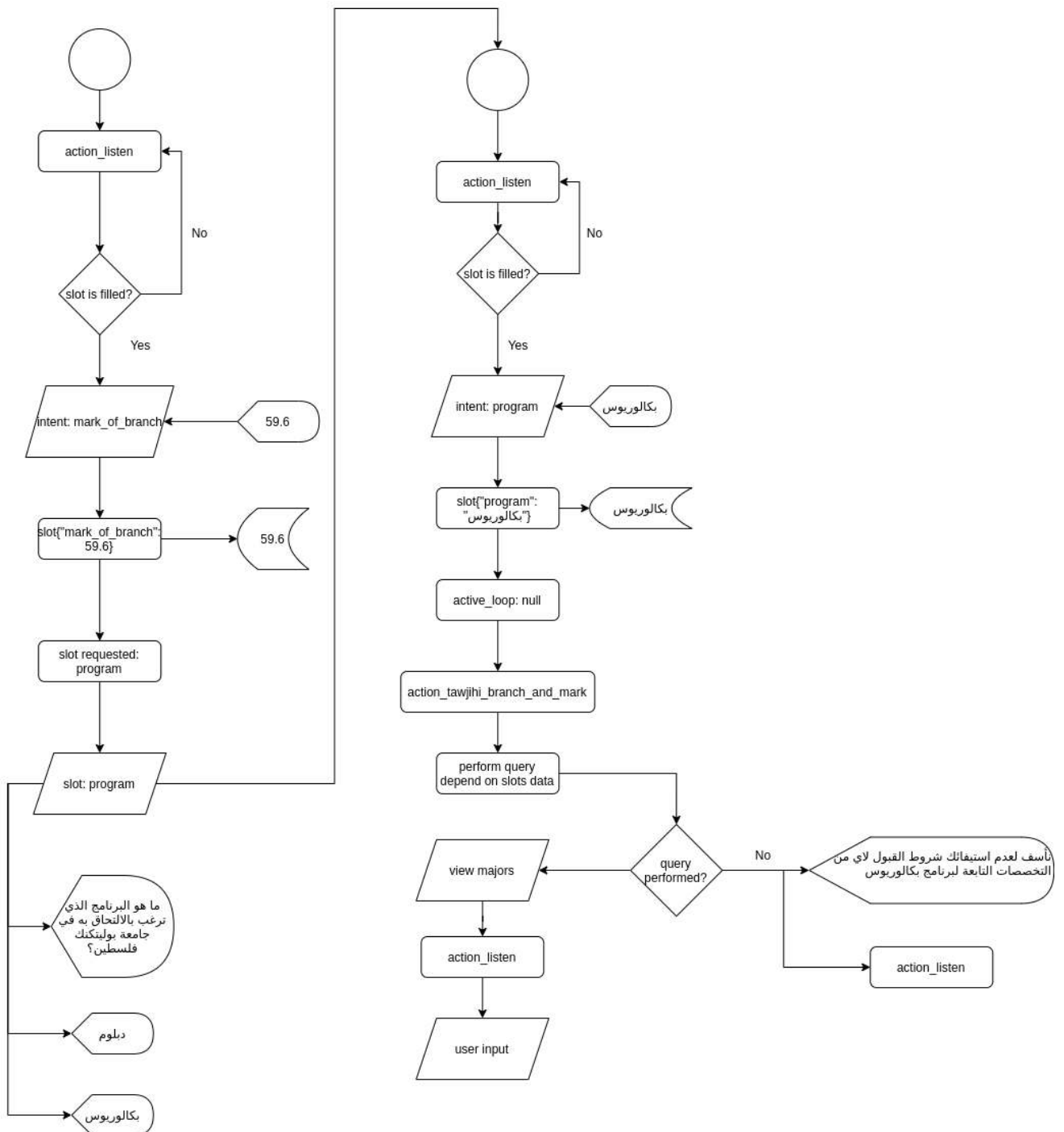
Figure 6.2: Test use case II

Conclusion

This bot was built to respond to the inquiries of the Tawjihi students regarding each of the university's faculties and their specializations, with extracted information for each specialization, familiarizing students with the level exams that students submit about their enrollment in the university, introducing the educational qualification diploma program and the mechanism for joining it. Giving students notes on the electronic enrollment application package, the locations of approved banks, and how to fill out the application. Introduce Bagrut students to the conditions and notes that must be taken into account in the event of joining Palestine Polytechnic University and the mechanism for calculating grades. Introduce students to the procedures followed to reserve a seat and what documents are required after the student is accepted. Introducing students to the system of transferring to Palestine Polytechnic University from another university on the undergraduate system. Informing students of the university's teaching system and language. Introducing students to the student exchange system with other universities. Introducing students to the system of grants, exemptions, and financial aid provided to students. Informing students of cases in which the student loses his university seat. Introducing students to the installment refund system for new students and its conditions.

Challenges: We faced many challenges, and the biggest challenge during the work was the arrest of my colleague Ali Jboor, a month before the deadline for the project. Dealing with the framework is one of the biggest challenges, as it is new and not widespread. And also the challenge of learning the necessary Python language to complete the construction of this bot. We can say that the current health situation contributed to creating communication problems between team members and problems related to slow internet and power outages during work.
We also mention the challenge we faced, so we divided the work into two parts, part for the admin panel and a part to follow up on the work on the bot. Unfortunately, Ali was arrested and I had to do the project alone in his absence, especially after we agreed on a day to share the results of the work, but that day Ali was arrested.

interesting decisions:
1- The idea of the project itself was very interesting. We considered this project and set it as a challenge to our abilities and ourselves. To be based on learning to use and implement programs using a non-renowned framework for a project that is the most important in a university student's career.
2- The idea of changing the operating system used to run the bot was one of the most crucial decisions in the workflow of the project. Windows was the best and most worthy on paper, according to the sources. But there was one problem that we had encountered for such a long time that made an operating system change necessary.

Recommendations: It is possible to modify and increase the efficiency of the bot to the fullest extent if the time factor and the human factor are available. Unfortunately, we were not able to deliver the bot to the maximum extent that we drew and expected due to the circumstances that befell us. Additional matters necessary for students related to registration, student status, and forms to official documents can be added.

Future improvements: The project scope may be expanded to include all corners of the university, including faculties and deanships of registration and follow-up of all matters that the student is interested in during their academic life.   The ability to communicate using voice messages.

References

1. "Chatbots - Artificial Solutions." [Chatbots | Conversational AI Platform for Enterprise - Teneo](). Accessed 8 Nov. 2020.

2. "How Do Bots and Chatbots Work."

    [https://www.cxtoday.com/contact-centre/how-do-bots-and-chatbots-work/]()

3. "Building Chatbots - A comparison of Rasa-NLU and Dialogflow." 17 Apr. 2018, [Building Chatbots - A comparison of Rasa-NLU and Dialogflow](). Accessed 8 Nov. 2020.

4. "Jonathan Wheat" [Jonathan Wheat - DEV Community](). Accessed 10 Nov. 2020.

5. Understanding the Basics of Rasa - Open source conversational AI

    [https://smazee.com/blog/basics-of-rasa]()

6. Rasa Architecture Overview

    [https://rasa.com/docs/rasa/arch-overview/]()

7. An Architecture for Dynamic Conversational Agents for Citizen Participation and Ideation by: Saifeldin Ahmed

    [(PDF) An Architecture for Dynamic Conversational Agents for Citizen Participation and Ideation (researchgate.net)]()

8. Installation: [https://rasa.com/docs/rasa/installation]()

9. How to test a chatbot. Accessed May 31-2021:

    https://apphawks.com/blog/a-compelling-guide-on-chatbot-testing

Appendix

1- Conversation between user and bot used in section 6.2.4:

> User: Hello
>
> Bot: Welcome
>
> User: What specialties are available?
>
> Bot: Enter your branch in high school
>
> User: scientific
>
> Bot: Enter your high school average
>
> User: 59.6
>
> Bot: What program would you like to enroll in at Palestine Polytechnic University?
>
> User: (presses the Bachelor button) / writes the word Bachelor
>
> Bot: We regret that you did not meet the admission requirements for any of the disciplines affiliated with the Bachelor's Program

2- In the following diagram, a real-time visualization of the conversation progression appears during the bot's interaction with the user's input.
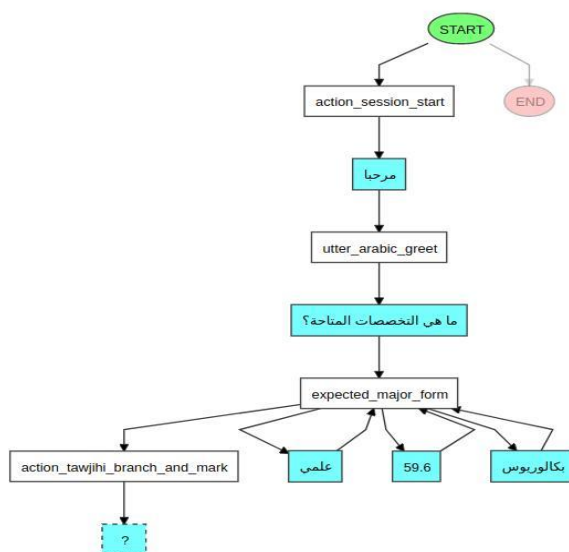
Figure I : run-time conversation visualization

3- Test case log file content:

(venv) maher@maher-PC:~/Documents/admission-bot$ rasa interactive

/home/maher/Documents/venv/lib/python3.7/site-packages/rasa/shared/utils/io.py:97: UserWarning:

Action 'utter_university_colleges' is listed as a response action in the domain file, but there is no

matching response defined. Please check your domain.

  More info at https://rasa.com/docs/rasa/responses

Nothing changed. You can use the old model stored at

'/home/maher/Documents/admission-bot/models/20210531-231738.tar.gz'.

2021-06-01 17:03:51 INFO   rasa.model - Loading model models/20210531-231738.tar.gz...

/home/maher/Documents/venv/lib/python3.7/site-packages/rasa/utils/train_utils.py:593:

UserWarning: constrain_similarities is set to `False`. It is recommended to set it to `True` when using

cross-entropy loss. It will be set to `True` by default, Rasa Open Source 3.0.0 onwards.

  category=UserWarning,

/home/maher/Documents/venv/lib/python3.7/site-packages/rasa/utils/train_utils.py:565:

UserWarning: model_confidence is set to `softmax`. It is recommended to try using

`model_confidence=linear_norm` to make it easier to tune fallback thresholds.

  category=UserWarning,

2021-06-01 17:04:07 INFO   root - Rasa server is up and running.

Processed story blocks: 100%|███████████| 26/26 [00:00<00:00, 1961.68it/s, # trackers=1]

Processed rules: 100%|███████████████| 10/10 [00:00<00:00, 3479.88it/s, # trackers=1]

Bot loaded. Visualisation at http://localhost:5006/visualization.html .

Type a message and press enter (press 'Ctrl-c' to exit).

? Your input -> مرحبا

? Your NLU model classified 'مرحبا' with intent 'arabic_greet' and there are no entit

ies, is this correct? Yes

Chat History

#	Bot	You
1	action_listen	
2		مرحبا
		intent: arabic_greet 1.00

Current slots:

colleges: None, major: None, branch_of_tawjihi: None, mark_of_branch: None, program: None,
requested_slot: None, session_started_metadata: None

? The bot wants to run 'utter_arabic_greet', correct? Yes

Chat History

#	Bot	You
1	action_listen	

#	Bot	You
2		مرحبا
		intent: arabic_greet 1.00

#	Bot	You
3	utter_arabic_greet 1.00	
	أهلا وسهلا بك	

Current slots:

colleges: None, major: None, branch_of_tawjihi: None, mark_of_branch: None, program: None,
requested_slot: None, session_started_metadata: None

? The bot wants to run 'action_listen', correct? Yes

? Your input -> ما هي التخصصات المتاحة؟

? Your NLU model classified 'ما هي التخصصات المتاحة؟' with intent 'expected_major' an
d there are no entities, is this correct? Yes

Chat History

#	Bot	You
1	action_listen	

#	Bot	You
2		مرحبا

<div dir="rtl">intent: arabic_greet 1.00</div>

---

3 utter_arabic_greet 1.00

<div dir="rtl">أهلا وسهلا بك</div>

action_listen 1.00

---

4 <div dir="rtl">ما هي التخصصات المتاحة؟</div>

<div dir="rtl">intent: expected_major 1.00</div>

Current slots:

    colleges: None, major: None, branch_of_tawjihi: None, mark_of_branch: None, program: None, requested_slot: None, session_started_metadata: None

------

? The bot wants to run 'expected_major_form', correct? Yes

Chat History

#	Bot	You

| 1 | action_listen | |

| 2 | | <div dir="rtl">مرحبا</div> <div dir="rtl">intent: arabic_greet 1.00</div> |

---

3 utter_arabic_greet 1.00

<div dir="rtl">أهلا وسهلا بك</div>

action_listen 1.00

---

4 <div dir="rtl">ما هي التخصصات المتاحة؟</div>

<div dir="rtl">intent: expected_major 1.00</div>

---

5 expected_major_form 1.00

active_loop{"name":

"expected_major_form"}

slot{"requested_slot":

"branch_of_tawjihi"}

أدخل فرعك في الثانوية العامة

Current slots:

    colleges: None, major: None, branch_of_tawjihi: None, mark_of_branch: None, program: None, requested_slot: branch_of_tawjihi, session_started_metadata: None

------

? The bot wants to run 'action_listen', correct? Yes

? Your input -> علمي

? Your NLU model classified 'علمي' with intent 'tawjihi_branch' and there are no enti

ties, is this correct? Yes

Chat History

#	Bot	You
1	action_listen	
2		مرحبا intent: arabic_greet 1.00
3	utter_arabic_greet 1.00 أهلا وسهلا بك action_listen 1.00	
4		ما هي التخصصات المتاحة؟ intent: expected_major 1.00
5	expected_major_form 1.00 active_loop{"name": "expected_major_form"} slot{"requested_slot":	

"branch_of_tawjihi"}

أدخل فرعك في الثانوية العامة

action_listen 1.00

| 6 | علمي |
| | intent: tawjihi_branch 1.00 |

Current slots:

    colleges: None, major: None, branch_of_tawjihi: None, mark_of_branch: None, program: None, requested_slot: branch_of_tawjihi, session_started_metadata: None

------

? The bot wants to run 'expected_major_form', correct? Yes

Chat History

#	Bot	You
1	action_listen	
2		مرحبا intent: arabic_greet 1.00
3	utter_arabic_greet 1.00 أهلا وسهلا بك action_listen 1.00	
4		ما هي التخصصات المتاحة؟ intent: expected_major 1.00
5	expected_major_form 1.00 active_loop{"name": "expected_major_form"} slot{"requested_slot":	

"branch_of_tawjihi"}

أدخل فرعك في الثانوية العامة

slot{"branch_of_tawjihi": "علمي"}

slot{"requested_slot": "mark_of_branch"}

أدخل معدلك في الثانوية العامة

Current slots:

colleges: None, major: None, branch_of_tawjihi: علمي, mark_of_branch: None, program: None, requested_slot: mark_of_branch, session_started_metadata: None

------

? The bot wants to run 'action_listen', correct? Yes

? Your input -> 59.6

? Your NLU model classified '59.6' with intent 'branch_mark' and there are no enti

ties, is this correct? Yes

Chat History

#	Bot	You
1	action_listen	
2		مرحبا intent: arabic_greet 1.00
3	utter_arabic_greet 1.00 أهلا وسهلا بك action_listen 1.00	
4		ما هي التخصصات المتاحة؟ intent: expected_major 1.00
5	expected_major_form 1.00 active_loop{"name":	

48

"expected_major_form"}

slot{"requested_slot":

"branch_of_tawjihi"}

أدخل فرعك في الثانوية العامة

slot{"branch_of_tawjihi": "علمي"}

slot{"requested_slot": "mark_of_branch"}

أدخل معدلك في الثانوية العامة

action_listen 1.00

6	59.6
	intent: branch_mark 0.55

Current slots:

colleges: None, major: None, branch_of_tawjihi: علمي, mark_of_branch: None, program: None, requested_slot: mark_of_branch, session_started_metadata: None

------

? The bot wants to run 'expected_major_form', correct? Yes

Chat History

#	Bot	You
1	action_listen	
2		مرحبا
		intent: arabic_greet 1.00
3	utter_arabic_greet 1.00 أهلا وسهلا بك action_listen 1.00	
4		ما هي التخصصات المتاحة؟
		intent: expected_major 1.00

5   expected_major_form 1.00

active_loop{"name":

"expected_major_form"}

slot{"requested_slot":

"branch_of_tawjihi"}

أدخل فرعك في الثانوية العامة

slot{"branch_of_tawjihi": "علمي"}

slot{"requested_slot": "mark_of_branch"}

أدخل معدلك في الثانوية العامة

slot{"mark_of_branch": 59.6}

slot{"requested_slot": "program"}

ما هو البرنامج الذي ترغب بالالتحاق به في

جامعة بوليتكنك فلسطين؟

Buttons:

1:

(بكالوريوس (بكالوريوس

2: دبلوم (دبلوم)

Type out your own message...

Current slots:

colleges: None, major: None, branch_of_tawjihi: علمي, mark_of_branch: 59.6, program: None,
requested_slot: program, session_started_metadata: None

? The bot wants to run 'action_listen', correct? Yes

ما هو البرنامج الذي ترغب بالالتحاق به في جامعة بوليتكنك فلسطين؟   1: بكالوريوس (بكال ؟
(وريوس

Chat History

#	Bot	You

1   action_listen

---

2   مرحبا

                                        intent: arabic_greet 1.00

---

3   utter_arabic_greet 1.00

    أهلا وسهلا بك

    action_listen 1.00

---

4   ما هي التخصصات المتاحة؟

                                        intent: expected_major 1.00

---

5   expected_major_form 1.00

    active_loop{"name":

    "expected_major_form"}

    slot{"requested_slot":

    "branch_of_tawjihi"}

    أدخل فرعك في الثانوية العامة

    slot{"branch_of_tawjihi": "علمي"}

    slot{"requested_slot": "mark_of_branch"}

    أدخل معدلك في الثانوية العامة

    slot{"mark_of_branch": 59.6}

    slot{"requested_slot": "program"}

    ما هو البرنامج الذي ترغب بالالتحاق به في

    جامعة بوليتكنك فلسطين؟

    Buttons:

    1:

    (بكالوريوس (بكالوريوس

    (دبلوم (دبلوم :2

    Type out your own message...

    action_listen 1.00

---

6   بكالوريوس

                                        intent: university_programs 1.00

Current slots:

  colleges: None, major: None, branch_of_tawjihi: علمي, mark_of_branch: 59.6, program: None,
requested_slot: program, session_started_metadata: None

? The bot wants to run 'expected_major_form', correct? Yes

Chat History

#	Bot	You
1	action_listen	
2		مرحبا intent: arabic_greet 1.00
3	utter_arabic_greet 1.00 أهلا وسهلا بك action_listen 1.00	
4		ما هي التخصصات المتاحة؟ intent: expected_major 1.00
5	expected_major_form 1.00 active_loop{"name": "expected_major_form"} slot{"requested_slot": "branch_of_tawjihi"} أدخل فرعك في الثانوية العامة slot{"branch_of_tawjihi": "علمي"} slot{"requested_slot": "mark_of_branch"} أدخل معدلك في الثانوية العامة	

slot{"mark_of_branch": 59.6}

slot{"requested_slot": "program"}

ما هو البرنامج الذي ترغب بالالتحاق به في

جامعة بوليتكنك فلسطين؟

Buttons:

1:

(بكالوريوس (بكالوريوس

(دبلوم (دبلوم :2

Type out your own message...

slot{"program": "بكالوريوس"}

slot{"requested_slot": null}

active_loop{"name": null}

Current slots:

   colleges: None, major: None, branch_of_tawjihi: علمي, mark_of_branch: 59.6, program: بكالوريوس, requested_slot: None, session_started_metadata: None

------

? The bot wants to run 'action_tawjihi_branch_and_mark', correct? Yes

------

Chat History

#	Bot	You
1	action_listen	
2		مرحبا
		intent: arabic_greet 1.00
3	utter_arabic_greet 1.00 أهلا وسهلا بك action_listen 1.00	
4		ما هي التخصصات المتاحة؟

_____

 5   expected_major_form 1.00

active_loop{"name":

"expected_major_form"}

slot{"requested_slot":

"branch_of_tawjihi"}

أدخل فرعك في الثانوية العامة

slot{"branch_of_tawjihi": "علمي"}

slot{"requested_slot": "mark_of_branch"}

أدخل معدلك في الثانوية العامة

slot{"mark_of_branch": 59.6}

slot{"requested_slot": "program"}

ما هو البرنامج الذي ترغب بالالتحاق به في

جامعة بوليتكنك فلسطين؟

Buttons:

1:

(بكالوريوس (بكالوريوس

2: (دبلوم (دبلوم

Type out your own message...

slot{"program": "بكالوريوس"}

slot{"requested_slot": null}

active_loop{"name": null}

action_tawjihi_branch_and_mark 1.00

نأسف لعدم استيفائك شروط القبول لاي من

التخصصات التابعة لبرنامج بكالوريوس

Current slots:

    colleges: None, major: None, branch_of_tawjihi: علمي, mark_of_branch: 59.6, program:

بكالوريوس, requested_slot: None, session_started_metadata: None

------

? The bot wants to run 'action_listen', correct? Yes

? Your input ->

4- rasa train log file content:

(venv) maher@maher-PC:~/Documents/admission-bot$ rasa train

/home/maher/Documents/venv/lib/python3.7/site-packages/rasa/shared/utils/io.py:97: UserWarning: Action 'utter_university_colleges' is listed as a response action in the domain file, but there is no matching response defined. Please check your domain.

  More info at https://rasa.com/docs/rasa/responses

2021-06-01 23:41:58 INFO   rasa.model - Data (domain) for Core model section changed.

2021-06-01 23:41:58 INFO   rasa.model - Data (messages) for NLU model section changed.

2021-06-01 23:41:58 INFO   rasa.model - Data (domain) for Core model section changed.

2021-06-01 23:41:58 INFO   rasa.model - Data (messages) for NLU model section changed.

Training NLU model...

/home/maher/Documents/venv/lib/python3.7/site-packages/rasa/shared/utils/io.py:97: UserWarning: Please configure the number of 'epochs' in your configuration file. We will change the default value of 'epochs' in the future to 1.

/home/maher/Documents/venv/lib/python3.7/site-packages/rasa/utils/train_utils.py:593: UserWarning: constrain_similarities is set to `False`. It is recommended to set it to `True` when using cross-entropy loss. It will be set to `True` by default, Rasa Open Source 3.0.0 onwards.

  category=UserWarning,

/home/maher/Documents/venv/lib/python3.7/site-packages/rasa/utils/train_utils.py:565: UserWarning: model_confidence is set to `softmax`. It is recommended to try using `model_confidence=linear_norm` to make it easier to tune fallback thresholds.

  category=UserWarning,

2021-06-01 23:42:00 INFO   rasa.shared.nlu.training_data.training_data - Training data stats:

2021-06-01 23:42:00 INFO   rasa.shared.nlu.training_data.training_data - Number of intent examples: 267 (40 distinct intents)

2021-06-01 23:42:00 INFO   rasa.shared.nlu.training_data.training_data -   Found intents: 'dploma_tahel', 'arabic_night_greet', 'registration', 'affirm', 'bot_condition', 'refund_system_for_new_student', 'arabic_goodbye', 'placement_tests', 'system_study', 'university_programs', 'Losing_a_seat', 'arabic_morning_greet', 'tawjihi_branch', 'continue_conversation', 'transfer_to_ppu', 'graduation_plans', 'choose_major', 'bot_challenge', 'deny', 'nlu_fallback', 'goodbye', 'parallel_study', 'continue_pajrot', 'abuse_words', 'pajrot_calculate_mark_and_major', 'filling_the_application', 'colleges', 'installments_and_credit_hours', 'choose_college', 'branch_mark', 'arabic_greet',

'pajrot_enrollment_Conditions', 'ppu_grants', 'restart', 'greet', 'admission_policy', 'thanks', 'language_study', 'required_documents', 'expected_major'

2021-06-01 23:42:00 INFO   rasa.shared.nlu.training_data.training_data - Number of response examples: 0 (0 distinct responses)

2021-06-01 23:42:00 INFO   rasa.shared.nlu.training_data.training_data - Number of entity examples: 0 (0 distinct entities)

/home/maher/Documents/venv/lib/python3.7/site-packages/rasa/shared/utils/io.py:97: UserWarning: Intent 'restart' has only 1 training examples! Minimum is 2, training may fail.

/home/maher/Documents/venv/lib/python3.7/site-packages/rasa/shared/utils/io.py:97: UserWarning: Intent 'continue_pajrot' has only 1 training examples! Minimum is 2, training may fail.

2021-06-01 23:42:00 INFO   rasa.nlu.model - Starting to train component WhitespaceTokenizer

2021-06-01 23:42:00 INFO   rasa.nlu.model - Finished training component.

2021-06-01 23:42:00 INFO   rasa.nlu.model - Starting to train component CountVectorsFeaturizer

2021-06-01 23:42:00 INFO   rasa.nlu.featurizers.sparse_featurizer.count_vectors_featurizer - 366 vocabulary slots consumed out of 1366 slots configured for text attribute.

2021-06-01 23:42:00 INFO   rasa.nlu.model - Finished training component.

2021-06-01 23:42:00 INFO   rasa.nlu.model - Starting to train component DIETClassifier

Epochs: 100%|████████████████| 300/300 [09:24<00:00, 1.88s/it, t_loss=1.46, i_acc=0.996]

2021-06-01 23:51:25 INFO   rasa.nlu.model - Finished training component.

2021-06-01 23:51:25 INFO   rasa.nlu.model - Starting to train component FallbackClassifier

2021-06-01 23:51:25 INFO   rasa.nlu.model - Finished training component.

2021-06-01 23:51:25 INFO   rasa.nlu.model - Successfully saved model into '/tmp/tmpj_0w3csd/nlu'

NLU model training completed.

Training Core model...

Processed story blocks:

100%|██

████████████████████████████████| 26/26 [00:00<00:00, 217.94it/s, # trackers=1]

Processed story blocks:

100%|██

████████████████████████████████| 26/26 [00:02<00:00, 9.74it/s, # trackers=25]

Processed story blocks:

100%|██

████████████████████████████████| 26/26 [00:06<00:00, 3.94it/s, # trackers=50]

Processed story blocks:

100%|██

██████████████████████| 26/26 [00:07<00:00, 3.38it/s, # trackers=50]

Processed rules:

100%|██

████████████████████████| 10/10 [00:00<00:00, 375.80it/s, # trackers=1]

Processed trackers:

100%|██

███████████████████████| 25/25 [00:00<00:00, 127.51it/s, # actions=135]

Processed actions: 135it [00:00, 1502.46it/s, # examples=135]

Processed trackers:

100%|██

███████████████████████| 525/525 [00:11<00:00, 44.32it/s, # actions=1960]

Epochs:

100%|██

████████████████| 100/100 [08:18<00:00, 4.98s/it, t_loss=7.88, loss=7.5, acc=0.904]

Processed trackers:

100%|██

██████████████████████| 10/10 [00:00<00:00, 338.96it/s, # actions=23]

Processed actions: 23it [00:00, 2582.98it/s, # examples=21]

Processed trackers:

100%|██

█████████████████████| 25/25 [00:00<00:00, 117.29it/s, # actions=138]

Processed trackers:

100%|██

██████████████████████████████| 10/10 [00:00<00:00,

156.09it/s]

Processed trackers:

100%|██

████████████████████████████████| 35/35 [00:00<00:00,

59.45it/s]

2021-06-02 00:01:55 INFO   rasa.core.agent - Persisted model to '/tmp/tmpj_0w3csd/core'

Core model training completed.

Your Rasa model is trained and saved at
'/home/maher/Documents/admission-bot/models/20210602-000157.tar.gz'.
(venv) maher@maher-PC:~/Documents/admission-bot$

5- rasa run actions log file content:

(venv) maher@maher-PC:~/Documents/admission-bot$ rasa run actions
2021-06-01 23:32:55 INFO   rasa_sdk.endpoint - Starting action endpoint server...
2021-06-01 23:32:56 INFO   rasa_sdk.executor - Registered function for 'action_session_start'.
2021-06-01 23:32:56 INFO   rasa_sdk.executor - Registered function for 'action_colleges'.
2021-06-01 23:32:56 INFO   rasa_sdk.executor - Registered function for 'action_college_major'.
2021-06-01 23:32:56 INFO   rasa_sdk.executor - Registered function for
'action_tawjihi_branch_and_mark'.
2021-06-01 23:32:56 INFO   rasa_sdk.executor - Registered function for 'action_major_details'.
2021-06-01 23:32:56 INFO   rasa_sdk.executor - Registered function for
'validate_expected_major_form'.
2021-06-01 23:32:56 INFO   rasa_sdk.endpoint - Action endpoint is up and running on
http://0.0.0.0:5055