Volume 64, Number 1, 2024

ACTA TECHNICA NAPOCENSIS
Electronics and Telecommunications

CHATBOT DESIGNED FOR INTERNATIONAL STUDENTS

Carla-Mihaela BULZESCU[1], Lacrimioara GRAMA[2]
[1,2]Faculty of Electronics, Telecommunications and Information Technology, Technical University of Cluj-Napoca, Cluj-Napoca, Romania
[1]Bulzescu.Ga.Carla@student.utcluj.ro, [2]Lacrimioara.Grama@bel.utcluj.ro

Abstract: The main purpose of this paper is to present a chatbot developed for international students. It provides guidance on a range of challenges, including safety, culture shock, housing, language, and academic notes. For this purpose, a dataset of 72 question-and-answer intents was developed from scratch, designed to assist students. The training phase uses the dataset to create the neural network model. The chatbot makes use of established rules and techniques for natural language processing to predict the appropriate responses to user's demand. Two Python-based versions of the chatbot are developed: one with a user interface, designed using a Flask server connected to a front-end section, and one with a compiler-based method that can handle speech input and output. Both versions achieved high accuracy during the training phase, suggesting good forecasting for the purpose upon which the chatbot was designed.

Keywords: *chatbot, artificial intelligence, neural network, natural language processing, text to speech.*

I. INTRODUCTION

Globally, there is a growing demand for chatbots, especially in the education sector. Students frequently spend time and energy on a lot of websites, and did not the needed information, mostly when studying abroad. Artificial Intelligence (AI) chatbots are developed to converse with people and respond to specific queries, saving time and effort.

Over the span of years, Python chatbots have grown exponentially, reaching almost the top of the charts in the technical and commercial world. Companies from a wide range of industries are deploying these intelligent bots since they are excellent at simulating human speech and cooperating with people [1].

A chatbot is an astute, conversational computer software which employs voice, text, or both to conduct conversations. It is versatile enough to carry out a variety of activities since it takes in a lot of data and reacts with conversational output [2].

Modern AI-based chatbots can detect similar conversations using data collection, machine learning, and natural language processing. They can perform tasks such as information provision, personalized product recommendations, and routine task automation [3]. Nowadays, chatbots have made great advancements and can now converse with one another like people. They can mimic human behavior by sharing personal experiences, being patient, confusing and even misleading.

In the 1960s, the first chatbot, ELIZA, was developed by Joseph Weizenbaum to imitate a psychologist using pattern matching [4]. Today, chatbots are used for communication on various online platforms, often for amusement or specialized purposes. Natural language processing is used in educational inquiry chatbot applications to understand and respond to user questions. Chatbots can be rule-based, self-learning, text-to-text, or speech-to-voice [5].

Since ELIZA, virtual assistants have evolved, leading to personal assistants like Apple's Siri, Amazon's Alexa, and ChatGPT. Siri, created in 2010, supports video, audio, and images, and uses various online resources to provide suggestions and answer user requests. However, it has limitations, such as the need for an internet connection, support for multiple languages, and difficulty hearing strong accents or background noise [6].

In [7] the opinions of educators and students in Krabi, Thailand, regarding the use of ChatGPT for educational purposes are examined. The study found that ChatGPT offers quick feedback and is generally viewed favorably in education. Although technology has the potential to lighten the burden, teachers expressed worries concerning the accuracy of the content and loss of interpersonal engagement, other major issues including data safety and confidentiality.

In 2019, a university chatbot, UNIBOT, was released to provide information about admission procedures and university activities. However, UNIBOT cannot fully understand customer queries due to SQL-generated questions [5].

Chatbots can be categorized into rule-based and self-learning models [8]. Rule-based chatbots follow pre-set rules and are often underpowered. Self-learning chatbots use machine learning algorithms and are divided into retrieval-based and generative models. Retrieval-based models use predefined answers and are precise, while generative models use modern algorithms and natural language processing to create dynamic conversations.

The goal of this work was to develop a chatbot which is intended to support international students who are moving to another country to further their studies. It is not intended to be a general bot, i.e. as ChatGPT, who can compose various written content (such as essays, post for social media, articles, code, etc.). The present chatbot uses

Manuscript received February 19, 2024; revised May 20, 2024

predetermined rules and natural language processing methods but lacks advanced features. It was designed to answer some general questions that students who study abroad might have.

Two Python-based implementations of the chatbot are presented: a user-friendly interface built on the Flask framework and a chatbot that processes speech input and output using a compiler method. Both versions offer valuable insights for international students studying in new countries. To support them with a variety of questions they could have about living arrangements, culture shock, language, university-related matters, and other topics, a database with a set of question-answer intents was built.

Both implementations are user-centered applications, like CiSA [9]. Compared to CiSA, our application presents also a chatbot that processes speech input and output, and it is not restricted only to the campus life.

The paper is organized as follows: first the dataset is described in Section II; then, in Section III, the implementation of the chatbot's versions is discussed; experimental results are presented in Section IV, while the conclusions are dragged in Section V.

II. DATASET

Various datasets can be found on open data sites and are accessible to the public, but in this case, the dataset was created entirely from scratch to meet the needs of the targeted audience.

Our dataset contains 72 intent objects with questions, each of which has multiple answers [10]. The dataset is in a JavaScript Object Notation (JSON) format, which is very convenient for data management and use in web applications. The code for creating and reading JSON data can be written using any programming language, making it adaptable and compatible with various systems and architectures [11]. The file is organized in a specific way and follows a particular scheme. It contains a list of intents, where *intents* are a key to an array of intent items. Each item includes keys for *tags*, *patterns*, and *responses*:

- *tags* represent a string corresponding to the label or category;
- *patterns* are an array of questions that the user can ask;
- *responses* represent a set of answers that the chatbot can deliver for the specified intent.

An example of such an intents object can be seen in Fig. 1. Here, the word "mental" marks the tag because it is an essential word that can be detected in every question of this intent item, so that the chatbot can use the tag to give proper answers.

The queries and worries that students frequently have when studying internationally are addressed by the dataset utilized for this chatbot. These questions center on major elements that have a big impact on the entire experience and their purpose is to help students with housing, language, culture shock, staying healthy, adjusting to a new university, and making new friends. By addressing these subjects, the bot tries to help individuals navigate the difficulties and unknowns involved with studying abroad.

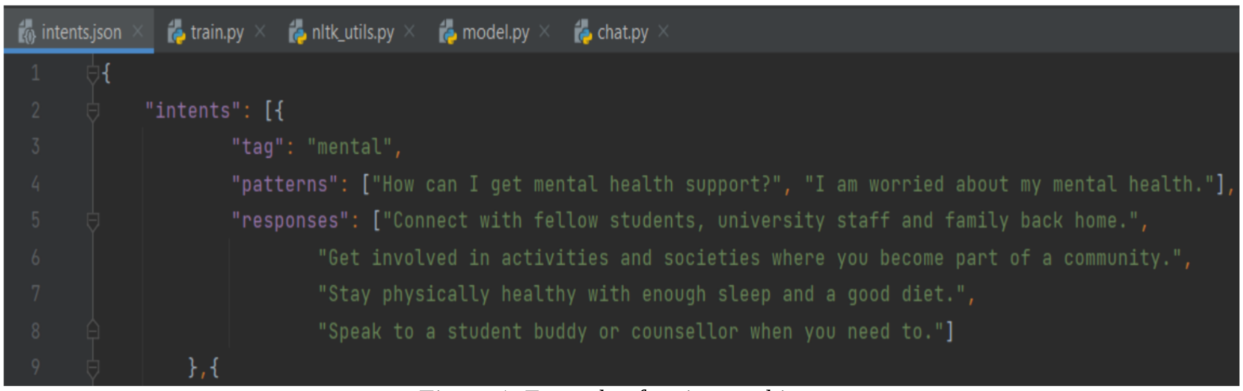
The 72 intent objects from the dataset [10] are: *academic, accommodation, affordable, bank, best, bring, club, cope, cultural, differences, disappointment, distracted, do, do not, easier, entertainment, events, exams, expectations, expenses, explore, exploring, facilities, favor, feedback, find classes, food, friends, goodbye, greetings, growing apart, habits, healthy, help, improve, interviews, involved, job, language, library, manage, mental, name, name1, need, need help, news, no answer, no help, notes, organized, phone, presentation, public, purpose, register, sense, shock, shops, sick, speak, stay, subjects, supervisor, support, thanks, things, tips, transportation, workplace, written.*

III. SOFTWARE ARHITECTURE

The chatbot uses predetermined rules and natural language processing (NLP) methods but lacks advanced features. Our chatbot application, based on Python, it is implemented in two different ways: one uses the Flask framework to set up the user-friendly interface, and the other is a compiler-based one, where the chatbot combines text-to-speech technology with speech recognition to record user input and deliver audio replies to the user.

Python was chosen as the primary programming language, and PyCharm as the integrated development environment for its creation. Sublime Text Editor was used for editing the front-end section, offering features like syntax highlight, automated indentation, document type recognition, and a sidebar.

The structure of the project, as depicted in Fig. 2 [10], consists of Python, the programming language used to write most of the code, the dataset that was developed in JSON format, some shared files across the two implementations, and the files that differ between the implementations.



```
{
    "intents": [{
            "tag": "mental",
            "patterns": ["How can I get mental health support?", "I am worried about my mental health."],
            "responses": ["Connect with fellow students, university staff and family back home.",
                    "Get involved in activities and societies where you become part of a community.",
                    "Stay physically healthy with enough sleep and a good diet.",
                    "Speak to a student buddy or counsellor when you need to."]
        },{
```

Figure 1. Example of an intent object.

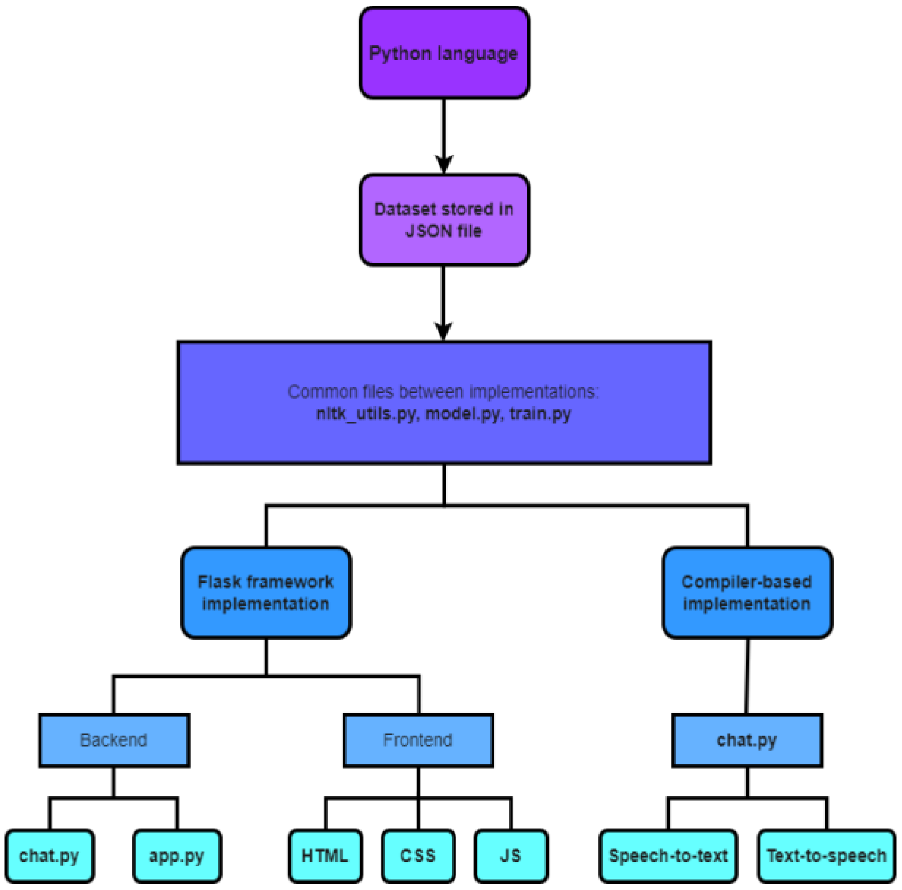

Python language
Dataset stored in JSON file
Common files between implementations:
nltk_utils.py, model.py, train.py
Flask framework implementation
Compiler-based implementation
Backend
Frontend
chat.py
chat.py
app.py
HTML
CSS
JS
Speech-to-text
Text-to-speech

Figure 2. Organizational structure.

A. Shared files across the two implementations

As illustrated in Fig. 2, the two versions share a set of common files written in Python that function in the same manner in both cases (*nltk_utils.py*, *model.py*, and *train.py*). Initially, to be pre-processed and used for the training phase, the text representing the intents in the JSON file went through several steps. A fully connected neural network (NN) model was built, and then the bot was trained on it and the prepared data.

Every distinct function in a fully linked layer is made up of a neuron that transforms the vector of inputs linearly using a weights matrix. The product undergoes a non-linear conversion using an activation function that encloses the dot product within the layer's input and weights matrix, and so the output vector is obtained.

In what follows we will try to describe the three common files between the two implementations.

1) The *nltk-utils.py* file contains text formatting functions for NLP and an AI field that converts data into strings. The file imports the required libraries and provides preprocessing functions like tokenization and stemming. The tokenize function divides a sentence into a collection of tokens/words using the *nltk.word_tokenize()* function as its input. The stem function stems from a word, meaning it finds the root of the word and converts the word to lowercase before returning it as a stemmed word. The next step in an NLP pipeline, after text processing, is the feature extraction stage, and in this case the bag-of-words technique has been used. With the help of the *bag_of_words* function, a tokenized sentence which is provided as input along with an array of recognized words, can be represented as a bag of words. At the beginning, the function fills an array of words with zeros. For every word that appears in the sentence, after it was stemmed, the associated place in the bag of words collection is changed to one. A tokenized sentence which is provided as input along with an array of recognized words, can be represented as a bag of words.

2) The *model.py* file contains an artificial NN model used for chatbot training. The NN module is loaded from a Python library to ofer the existing NN layers and functionalities; as an instance of *module* class a *NeuralNet* subclass is created. A constructor is built, and three linear transformation layers are generated. The same three arguments that were provided to the original constructor (input_size, hidden_size, and num_classes) are passed to each layer as input and output neurons. Repeating the process, a linear transformation layer is given to the input, and a ReLU activation function to the layer's output. The network's final output is restored, and it shows what is happening when the second ReLU function output is subjected to the linear transformation of the third layer.

3) The *train.py* file trains the NN model; intents are obtained from the JSON file and three lists are created for the words, tags and "xy" pairs, representing the patterns and tags associated with the intent objects.

Every sentence corresponding to patterns from the JSON file is then passed by the code. Each of the patterns are tokenized, its components are added to the words list, and the tokenized phrases and tags are added to the “xy” list. The words are stemmed, lowered, and sorted and duplicates are removed. The tag list stored the ordered tags. The bag-of-words format is used to transform sentences representing patterns while iterating through the “xy” list to build the data for training. Hyperparameters like input, hidden or output size, learning pace, and epochs count are set to control the NN's layout and training. Two primitives are included to enable the use of both prepared and newly generated datasets. A data loader is created to manage batching and shuffling of data used for training. The input, hidden, and output parameters already defined are supplied to a particular instance of the constructed model and the Adam optimizer is initialized. The machine learning performance of the classification model is measured by cross-entropy loss. The preset number of epochs are used for the training phase. The data used for training is fed into the model in batches, during each epoch and both forward pass and loss are determined. Next, the gradients of the optimizer are set to zero before beginning the second pass in reverse. The step function of the Adam optimizer is then used for updating the parameters that comprise the model and the loss amount is set to be printed after one hundred epochs.

B. Flask framework backend implementation

The Flask web application framework was used to develop the user interface, which is divided into two sections: the files which are shared by the two implementations (see subsection III.A), as well as a file titled *chat.py* for the chatbot’s interface functionality and a script *app.py* to set up the Flask web server (they are found in the application’s backend). The files that made up the front-end section (*base.html*, *style.css*, and *app.js*) were edited using the Sublime Text Editor.

1) The *chat.py* file develops the basic interface, includes the essential libraries, packages, and functions. The JSON file's intents are loaded in the same manner as in the *train.py* script. A console-based implementation links the interface to the server so that the user can test the chatbot before connecting to the Flask application. The chatbot asks for further assistance after an inactivity period; to keep track of user actions, a variable is updated with the current time.

2) The *app.py* script configures the Flask web application server. The *app* variable is initialized with a Flask class, and origins can be provided for content loading. The response variable stores the JSON entity as an answer, and the server can be tested locally using a function.

3) The *base.html* file is responsible for delivering an internet browser user interface. It includes a head section with metadata, a title bar, and an icon. The body section outlines the bot's interface and is styled with Cascading Style Sheets (CSS). A chat container displays user messages, and a JavaScript (JS) file handles user input. The HyperText Markup Language (HTML) file automatically refreshes the chat box with the bot's answer, ensuring a continuous conversation experience.

4) The *style.css* script is used to style the front-end section of the chatbot, applying styles to elements based on their type. The cascading approach allows multiple style rules to be applied to a single element, with higher specificity selectors preceding lower specificity selectors. Media queries can also be used to adjust the interface on different devices.

5) The *app.js* file is the final script from this part, defining a class called *Chatbox* that manages the chatbot's interaction with the user. It initializes properties and configurations, including the open and send buttons, event listeners, input field, and interval loop. The *Chatbox's* status is switched between active and inactive states, and user input is added to the array of messages.

C. Compiler-based implementation

The second implementation of the chatbot combines text-to-speech (TTS) technology with speech recognition to record user input and deliver audio replies to the user. The NN model is trained to receive user input and provide responses based on predetermined intents and can handle user inactivity. The compiler-based chatbot is based on the same shared files (see subsection III.A). The model is trained on the same JSON file, and it does not have a front-end section because it only runs in the console. These files are used in the creation of the bot, along with the chat.py script, which is modified to include the TTS and speech recognition features.

A speech engine was created using the *pyttsx3* package, and the chatbot would respond with an arbitrary response if the predicted tag had a likelihood greater than the predetermined threshold of 0.75.

The main loop continuously monitors the microphone for user input, converting audio into text with the speech recognition module, providing an answer, and breaking the loop if the identified phrase contains predefined sentences, such as “see you later” or “goodbye”. The bot interacts with the user, writes a response in the console, and plays audio. If voice recognition misses collecting input, the user is asked for additional assistance.

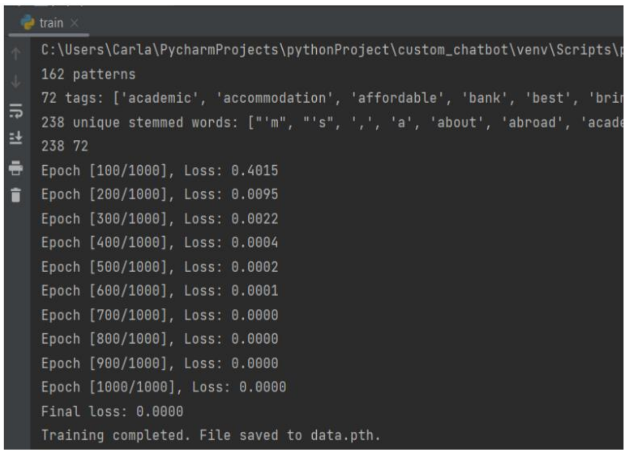
IV. EXPERIMENTAL RESULTS

The chatbot implementation process involves testing both versions, with the *nltk-utils.py* and *model.py* files being launched first.

A. Training phase

The chatbot is trained using the defined functions and the model. The results from the training phase can be seen in Fig. 3, which shows 162 patterns, 72 tags, and 238 stemmed words.

The model is trained using the entire dataset, with each tag fed into the NN model. The model runs for 1000 epochs, with the most recently reached loss value being displayed as “Final loss”. A value ranging from zero to one represents the loss, with zero representing the ideal model, so the objective is usually to bring the model as close to zero as possible.



```
C:\Users\Carla\PycharmProjects\pythonProject\custom_chatbot\venv\Scripts\p
162 patterns
72 tags: ['academic', 'accommodation', 'affordable', 'bank', 'best', 'bri
238 unique stemmed words: ["'m", "'s", ',', 'a', 'about', 'abroad', 'acade
238 72
Epoch [100/1000], Loss: 0.4015
Epoch [200/1000], Loss: 0.0095
Epoch [300/1000], Loss: 0.0022
Epoch [400/1000], Loss: 0.0004
Epoch [500/1000], Loss: 0.0002
Epoch [600/1000], Loss: 0.0001
Epoch [700/1000], Loss: 0.0000
Epoch [800/1000], Loss: 0.0000
Epoch [900/1000], Loss: 0.0000
Epoch [1000/1000], Loss: 0.0000
Final loss: 0.0000
Training completed. File saved to data.pth.
```

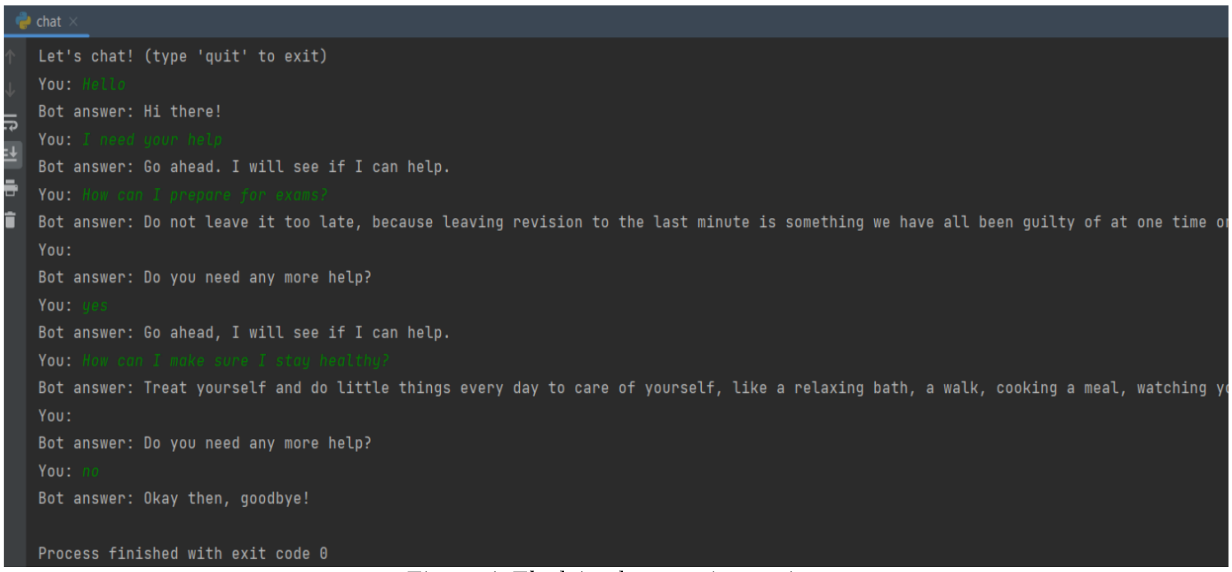
Figure 3. Training phase results.

From Fig. 3 it can be stated that the model exhibits high accuracy, since the loss is 0.0004 after 400 epochs and zero after 700 epochs. The user is informed of the completion of the training process.

B. Flask Framework testing

The user can test the chatbot in a console-based main loop before connecting to the Flask web application. The *data.py* file contains training data for the chatbot. The main loop is executed and, in the console, the “Let’s chat! (type ‘quit’ to exit)” message appears inviting the user to start the conversation. If at some point the client wants to finish the chat, they merely need to type the word “quit” in the terminal and the process will stop.

Any conversation will start with “You:” for the user’s perspective, followed by “Bot answer:” for the bot’s perspective. It can be noticed that the bot responds to greetings accurately and can respond to the questions it was taught to answer. Moreover, the bot will ask if additional help is required once the 10 second period of inactivity has passed. In case the response is affirmative, the dialogue can continue and if it is negative, the chatbot bids the user farewell and concludes the exchange. An illustration of the bot testing phase in the console is shown in Fig. 4.



```
Let's chat! (type 'quit' to exit)
You: Hello
Bot answer: Hi there!
You: I need your help
Bot answer: Go ahead. I will see if I can help.
You: How can I prepare for exams?
Bot answer: Do not leave it too late, because leaving revision to the last minute is something we have all been guilty of at one time o
You:
Bot answer: Do you need any more help?
You: yes
Bot answer: Go ahead, I will see if I can help.
You: How can I make sure I stay healthy?
Bot answer: Treat yourself and do little things every day to care of yourself, like a relaxing bath, a walk, cooking a meal, watching y
You:
Bot answer: Do you need any more help?
You: no
Bot answer: Okay then, goodbye!

Process finished with exit code 0
```

Figure 4. Flask implementation testing.

The front-end component of the web application needs to be linked to the Flask server through the *app.py* script. The Flask framework enters debug mode, allowing the front-end component to establish a connection between the application and the web browser. The scripts are changed using Sublime Text Editor, which also allows for the installation of a package called Browser Sync. This plugin synchronizes connections and code updates across devices, speeding up the workflow.

Fig. 5 shows what will be initially displayed on the page in the web browser after it has been loaded in a new window tab. In the browser’s top bar, users can see the title and chatbot icon that were defined in the HTML file. The button that displays the user interface and is in the lower right corner of the page initially displays a blank page. It needs to be pressed to turn it on and start the interface. If the button is pressed again while chatting with the bot, the interface can always be returned to its inactive state.

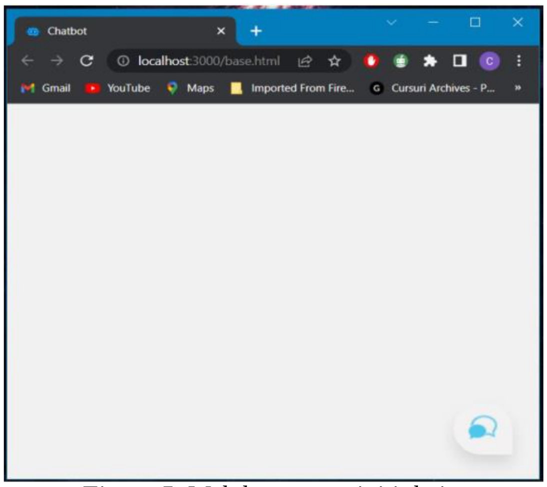
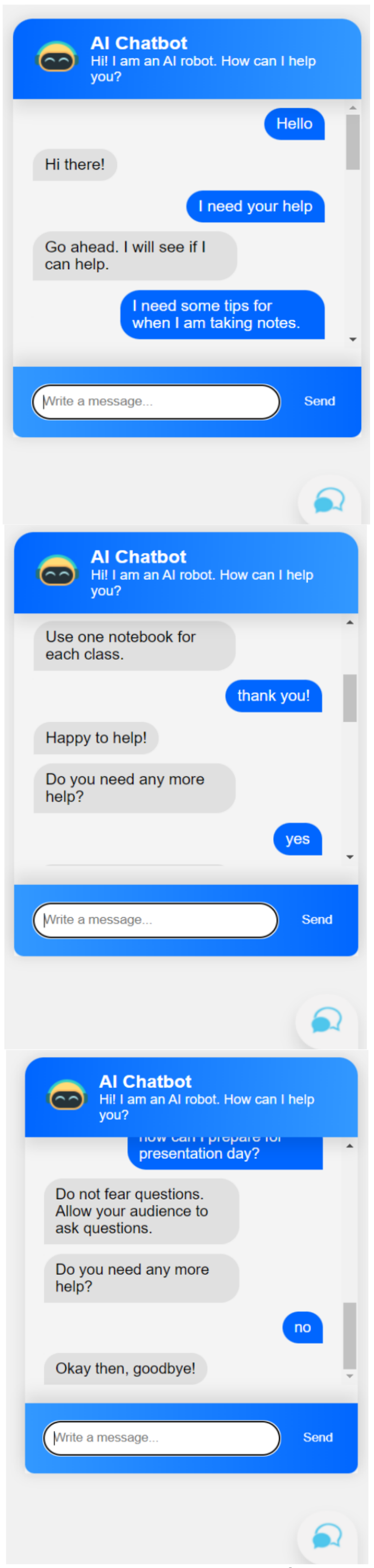


Figure 5. Web browser – initial view.

An interaction between the user interface and the bot can be observed in Fig. 6. A greeting message and an icon have been placed in the interface's top part.


AI Chatbot
Hi! I am an AI robot. How can I help you?

Hello
Hi there!
I need your help
Go ahead. I will see if I can help.
I need some tips for when I am taking notes.

Write a message... Send

AI Chatbot
Hi! I am an AI robot. How can I help you?

Use one notebook for each class.
thank you!
Happy to help!
Do you need any more help?
yes

Write a message... Send

AI Chatbot
Hi! I am an AI robot. How can I help you?

how can I prepare for presentation day?
Do not fear questions. Allow your audience to ask questions.
Do you need any more help?
no
Okay then, goodbye!

Write a message... Send

Figure 6. Chatbot conversation flow example.

The bot functions exactly as it would if a user was to test the interface in the terminal. It replies to greetings, offers responses to user input, requests more assistance after a certain period of inactivity, and deals with circumstances where the user needs more assistance or not.

The input field and send button can also be seen in the figure, along with a scrolling bar that was also implemented in the front-end section, and the button that changes the interface's state.

C. Compiler-based implementation testing

After the training phase is fulfilled, the final file that must be executed for the chatbot to function is *chat.py*. TTS and speech recognition tools are used in this version, which operates in the PyCharm terminal. An example of a user-chatbot interaction using TTS and voice recognition is shown in Fig. 7. The bot displays a greeting, invites the user to ask for help, and then processes user input and generates a response. If the user responds affirmatively, the conversation continues, otherwise the bot ends the interaction.

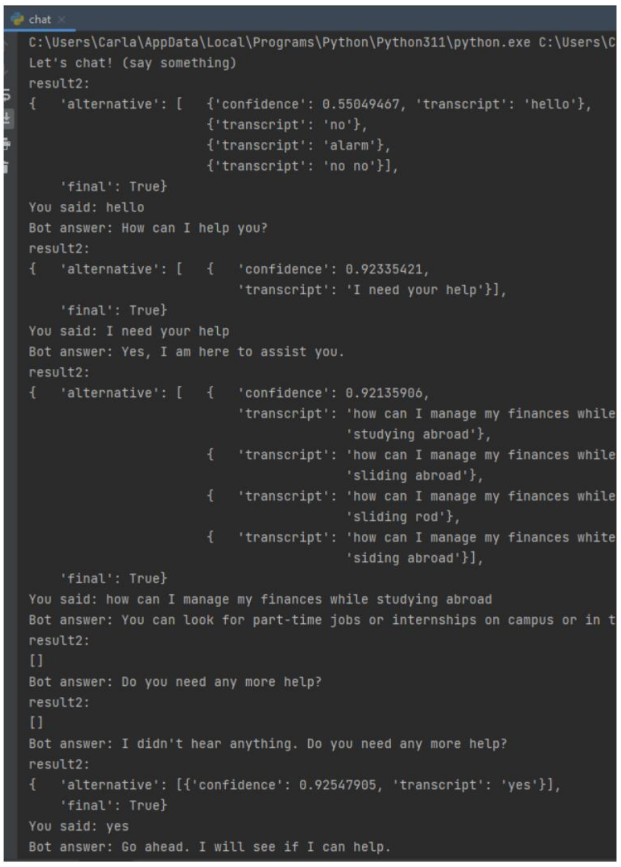

```
chat
C:\Users\Carla\AppData\Local\Programs\Python\Python311\python.exe C:\Users\C
Let's chat! (say something)
result2:
{   'alternative': [   {'confidence': 0.55049467, 'transcript': 'hello'},
                       {'transcript': 'no'},
                       {'transcript': 'alarm'},
                       {'transcript': 'no no'}],
    'final': True}
You said: hello
Bot answer: How can I help you?
result2:
{   'alternative': [   {   'confidence': 0.92335421,
                           'transcript': 'I need your help'}],
    'final': True}
You said: I need your help
Bot answer: Yes, I am here to assist you.
result2:
{   'alternative': [   {   'confidence': 0.92135906,
                           'transcript': 'how can I manage my finances while
                                          'studying abroad'},
                       {   'transcript': 'how can I manage my finances while
                                          'sliding abroad'},
                       {   'transcript': 'how can I manage my finances while
                                          'sliding rod'},
                       {   'transcript': 'how can I manage my finances white
                                          'siding abroad'}],
    'final': True}
You said: how can I manage my finances while studying abroad
Bot answer: You can look for part-time jobs or internships on campus or in t
result2:
[]
Bot answer: Do you need any more help?
result2:
[]
Bot answer: I didn't hear anything. Do you need any more help?
result2:
{   'alternative': [{'confidence': 0.92547905, 'transcript': 'yes'}],
    'final': True}
You said: yes
Bot answer: Go ahead. I will see if I can help.
```

Figure 7. Compiler-based chatbot interaction.

V. CONCLUSIONS AND FUTURE WORK

Through this paper a chatbot to assist international students moving to another country for further studies was discussed. To support students with a variety of questions they could have about living arrangements, culture shock, language, university-related matters, and other topics, a database with a set of 72 question-answer intents was built.

The proposed chatbot comes in two versions: a user interface-based one built using the Flask framework that is linked to the frontend section, and a compiler-based one that uses text-to-speech and speech recognition to streamline user interaction and relies on audio as a form of communication.

The dataset used for training was created from scratch and the chatbot makes use of established rules and techniques for natural language processing to predict the appropriate responses to the user's queries. Both bot versions achieved high accuracy during the training phase, indicating good forecasting for the purposes upon which it was designed.

In intelligent messaging systems used in education, chatbots are becoming more prevalent, and will inevitably develop in power and grow more intelligent as artificial intelligence and machine learning fields are progressing. The achieved results and accuracy levels support the assertion that there are many implementation options for chatbots and depending on the dataset, they may be exploited in various industries.

Future development of the chatbot described in this paper could benefit from enhancing the user interfaces and expanding the database to provide more sophisticated assistance for international students. Later, the bot might be incorporated in an Android application that would be handier for the users and it could also include the input-output audio features.

REFERENCES

[1] K. Goyal, “Top 12 Commerce Project Topics & Ideas in 2023 [For Freshers],” upGrad blog.
[2] N. K. Chauhan, “Create a voice chatbot in python using NLTK, Speech Recognition, Google (text-to-speech) & Scikit-learn,” Medium, Dec. 14, 2021.
[3] T. Capacity, “How does an AI chatbot work, and what does it mean for the future?,” Capacity, Apr. 11, 2023.
[4] M. Lundell Vinkler, P. Yu, “Conversational Chatbots with Memory-based Question and Answer Generation,” Dissertation Thesis, Department of Science and Technology, Linköping University, The Institute of Technology, 2020.
[5] R. Parkar, Y. Payare, K. Mithari, J. Nambiar and J. Gupta, “AI And Web-Based Interactive College Enquiry Chatbot,” 13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Pitesti, Romania, 2021, pp. 1-5, doi: 10.1109/ECAI52376.2021.9515065.
[6] E. Adamopoulou, L. Moussiades, “Chatbots: History, technology, and applications,” Machine Learning with Applications, vol. 2, 2020, doi: 10.1016/j.mlwa.2020.100006..
[7] P. Limna, T. Kraiwanit, K. Jangjarat, P. Klayklung, P. Chocksathaporn, “The use of ChatGPT in the digital era: Perspectives on chatbot implementation,” Journal of Applied Learning & Teaching, vol. 6, no. 1, May 2023.
[8] D. Maina, “How to create an AI Chatbot in Python and Flask”, DEV Community, Feb. 2022.
[9] J. Heo, J. Lee, “ CiSA: An Inclusive Chatbot Service for International Students and Academics,” HCI International 2019 – Late Breaking Papers, Lecture Notes in Computer Science, vol 11786. Springer, Cham, doi: 10.1007/978-3-030-30033-3_12.
[10] C.M. Bulzescu, “Chatbot support for international students,” Diploma Thesis, Faculty of Electronics, Telecommunications and Information Technology, Technical University of Cluj-Napoca, Romania, July 2023.
[11] L. Bassett, ”Introduction to JavaScript Object Notation: A To-the-Point Guide to JSON,” 2015. https://www.amazon.com/Introduction-JavaScript-Object-Notation-Point/dp/1491929480.