ARTIFICIAL INTELLIGENCE POWERED CHAT-BOT FOR CURRENT AND FUTURE FIRST-GENERATION STUDENTS OF CSUDH

_____

A Project

Presented

to the Faculty of

California State University, Dominguez Hills

_____

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

_____

by

Karen Salinas

Spring 2020

PROJECT: ARTIFICIAL INTELLIGENCE POWERED CHAT-BOT FOR CURRENT AND FUTURE FIRST-GENERATION STUDENTS OF CSUDH.

AUTHOR: KAREN A. SALINAS

APPROVED:

_____
Jack (Jianchao) Han, Ph.D.
Project Committee Chair

_____
Bhrigu Celly, Ph.D.
Committee Member

_____
Mohsen Behesthi, Ph.D.
Committee Member

I dedicate my work to my father and mother, that although foreign to a country in their youth, they never gave up on providing the best they could in my education and dreams. With the tools of life, they provided me, I never gave up on my goals even when those goals felt far away. I am also in deep gratitude to my longtime partner and fiancé, Miller, for always being by my side when I faced academic and life struggles. I am blessed that we both had each other when times felt rough and gave each other words of encouragement whenever we were in dire situations.

I would also like to extend my gratitude to friends and colleagues I've made through the years in this journey where we would connect and help each other out whenever we did not understand complex material. I am also very thankful to all the opportunities I have received throughout the years, most especially last year's opportunity to go to the Great Minds In Stem Conference in Florida where I was able to obtain a job close to home. Finally, I am thankful to God for walking with me on this journey and letting me breathe the fresh air another day especially in troubling times faced around the globe. I pray that we all make it safe when that time is over.

ACKNOWLEDGEMENTS

I want to thank the members of my committee for giving me the opportunity to seal my journey in my academic path that will help transition from a student to a professional. A special thank you for Dr. Han, for accepting my proposal and giving me the opportunity to present my work this semester. I would also like to thank Dr. Celly, for giving me a head start on the project when I most urgently needed help. Finally, I would like to thank Dr. Beheshti for giving me the opportunity during the fall of 2019 to attend GMIS where I landed a job in a defense company.

I want to also thank California State University, Dominguez Hills for allowing me to transition early on from California State University, Bakersfield and allowing me to admit early in January 2019 to start on my master's path. Finally, I am thankful for having to pursue the master's program on Computer Science with the concentration in the Software Engineering track at CSUDH and meeting and learning from the CSUDH Computer Science faculty that teachings will help me in my future career path.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

The first year of college can be an intimidating experience for first-generation students who are the first of their family to attend college. Some of the challenges faced by first-generation college students is college readiness since they do not have the enough tools to navigate through the school's system. This can cause students to feel lost and results in the student taking longer to graduate. Therefore, the motive of the project aims to help first-generation students into easing their access of information by integrating into the school's website an artificial intelligence powered chat-bot that uses Natural Language Processing (NLP), a field in machine learning that aims to provide the student with answers on their queries. In addition, the chat-bot aims to provide student support to current students who may find retrieving information from the school's website complex.

CHAPTER 1

INTRODUCTION

An integrated chat-bot into the school's website would be an essential tool for incoming and recurrent students of CSUDH who have trouble navigating through the school's website. The chat-bot would give students the ability to enter their queries and get responses quickly without the need of human interaction. Artificial intelligence (AI) tools like the chat-bot would improve engagement with students by providing them with student support services that eliminates the hassle of navigating through every page to get one piece of information. Furthermore, the addition of a chat-bot into the school's website is an appealing concept to attract more potential students from various backgrounds to apply to CSUDH.

Throughout the development of CSUDHBot, the agile methodology was applied to break down several stages of the phases for continuous improvement and iteration of the bot. The process began by planning out the type of software technologies and areas of data collection that are to be utilized for the implementation for CSUDHBot. Several of the technologies applied in the project were a combination of Python, YML, Markdown File Extensions (md), HTML, CSS, JavaScript, and the PostgreSQL procedural language PL/pgSQL. Furthermore, the data collection targeted the subjects the bot would converse with the user. In this case, the bot would provide the user to enter inquiries, guide them with steps over the school's programs and application process, chit-chat, and allow users to ask questions. Once the planning was done, the design requirements were set out. This included identifying the stakeholders in the project that included the students, administration, and general users, and setting out writing out the stories as they would before being implemented in the bot. Also, the class and ER-diagrams were designed since they were tied with the stories. Since only one person in the team was working, a

hypothetical team was created in terms of how the bot would have been developed in a software engineering environment.

When the design was done, the implementation took place by setting up the operating system used for the project, which was Ubuntu 16.08 LTS and then setting up the Anaconda Distribution Package to create and use an environment called ‘rasax’. Inside the environment, all of the RasaX toolset was downloaded with all of its corresponding packages and framework aimed to improve the contextual assistant (chatbot). After RasaX was set up, a new project was created through the command line with ‘rasa init’ where a pre-built project with the rasa framework was ready to use for further implementation. The next technology downloaded was pgAdmin that contained PostgreSQL procedural languages in order to create or drop tables to insert, update, or delete columns and rows within the database. All of the tables created followed the ER-diagram and once all the data that was collected was inserted, queries were created that consisted primarily of the SELECT statement and INNER JOIN and FULL OUTER JOIN keywords to match values from several tables.

The implementation phase began once data was collected, all the technologies were downloaded, a new project was created using the rasa framework, and the database was implemented in the database management system that leads to the implementation phase. This included implementing in the rasa framework the intents, entities, domains, slots, stories, and actions as well as setting up the endpoints and configuration that contains the policies.

After the implementation came the testing where each use case was tested to see if the functionality worked in the bot. This was done with the command ‘rasa data validate’ command. Initially, test cases were not recorded, but after the presentation a review of the test case approach was reconsidered where a few test cases thereafter were generated. Furthermore,

RasaX offers different testing methods but were not used for the project such as continuous integration testing since it deals with checking the merge and pull requests from GitHub. After testing the bot, the bot deployed by pushing it onto GitHub. The pull request was not used since this was an individual project. Lastly, the overall performance of the project was reviewed by checking if there were problems with the intents or problems in the overall story structure to change it according to the conversation plan between bot and user. The entire process went to the next iterations for the entire development of the CSUDHBot.

The main functions of CSUDHBot includes the following: taking in student's inquiries, student querying chit-chatting keywords to randomly talk to chatbot in the middle of the conversation, student querying search keywords to ask over 'employment', 'campus', or 'financial aid', and students querying to bot whether they are an 'incoming undergraduate student' or 'incoming graduate student' only to be redirected to a conversation that walks the student through a conversation. Moreover, some of the major components of CSUDHBot is that it is portable since it can be used in both Windows and MacOS operating systems, has usability in which is easily launched, it is integrable since it can integrate with many custom integrations, and contains good performance since it can reply to a user's input quickly.

The following chapters will discuss the overall development of the CSUDHBot. Chapter 2 explains the problem and motivation of CSUDHBot, Chapter 3 goes over some of the historical background relating to the beginnings of artificial intelligence and the first applications of natural language processing chat-bots. Chapter 4 then heads off with the scope of the project. Chapter 5 goes overs the general description that explains the product's perspective, feasibility report, user class characteristics and general constraints. Chapter 6 overviews the user requirement definitions whereas Chapter 8 goes over the system design by providing the sequence diagrams followed by

the E-R diagrams and class diagrams of the database, as well as the database's tables. Chapter 7 overviews the system architecture that includes the software modules and software architecture of both Rasa and the CSUDHBot. Chapter 9 goes over the interface design that includes the system interfaces, non-functional requirements, operating system, hardware interfaces, and communication interfaces. Chapter 10 provides the implementation process of CSUDHBot after the overall design of the system. Chapter 11 goes over the CSUDHBot model that comes over training of the models, running and loading the models, and testing the model. Chapter 12 shows the execution results of the CSUDHBot that shows how it would interact with the user. Finally, Chapter 13 talks about the future work that could be expanded with CSUDHBot. Furthermore, appendices are added towards the end to display the code, tables, PostgreSQL statements, and glossary of CSUDHBot.

CHAPTER 2

PROBLEM AND MOTIVATION

The motivation of the project is to assist and support incoming and recurring students of CSUDH with a chat-bot that eases their experience of finding information they may have trouble retrieving from the school's main website. This comes from observation and research that many students in their first year of college have trouble navigating through the university system since it significantly differs from their high school experience. Students that coming out of high school are pre-conditioned to follow a structure where they have classes every period. Therefore, the first year for most college students is confusing and frustrating because that structure is no longer there, and many times have trouble finding the right resources online that leads them to get lost over how the university system works.

Furthermore, to provide a solution to the problem, a chatbot called CSUDHBot is developed to make the college experience easier for incoming students. The bot also extends its services to ongoing students who may want to take a shortcut from navigating the website by asking the bot to give them an answer quickly from their queries. In addition, having the chatbot in the school's main website can improve the response rate in comparison to the school's human support team and thus, minimizes the time a student spends retrieving information.

CHAPTER 3

BACKGROUND

The chat-bot is a descendant of a proposed question made by Alan Turing in the 1950's, "Can machines think?" [1]. His development and creation of the Turing Test measured whether one was talking to a human or to a computer (in our case, the chat-bot). This stemming idea brought various developments of chat-bot like software's that aimed to pass the Turing Test. One of the first chat-bot-like software's developed was Eliza, where she was created in 1966 by Joseph Weizenbaum [1]. Eliza was able to fool some users into thinking they were talking to a human, however Eliza failed to pass the Turing Test [1]. Although her failure, Eliza laid out a foundation to the future structures of chat-bots. Moving forward to the present from various developments of improving Eliza, present bots like Alexa have been underway of becoming more intelligent in order to communicate with anyone [1]. However, Alexa as of present has not been able to pass the Turing Test where other competitors like the Google Duplex have managed to almost pass it [1]. The real game changer to improving communication between a human and bot has been the application of neural networks and machine learning.

Neural networks and machine learning are subsets of artificial intelligence (AI) where through application to a chat-bot enables it to recognize patterns of the input sentences made from

the user to respond with sentences from a template. Chat-bots are classified as conversational and domain-based bots [2]. Conversational bots are designed to fulfill tasks such as chit-chatting without reaching a specific goal, while the domain-based bots are classified as two types such as open and closed domain bots [2]. The open domain bots, or horizontal bots are designed to answer any questions, in which bots currently in use like Cortana or Amazon's Alexa, fall into this category [2]. Furthermore, different developments of chat-bot are available, such as Deep Learning bots that require Neural Networks in order to learn the input sequence [2]. Other branches that exists is the Natural Language Processing (NLP) bot that falls under artificial intelligence branch [2]. NLP is set to recognize speech and text [2]. Eliza, as mentioned before, is an NLP system that simulated a conversation through a series of pattern matching. NLP has evolved and since the 1980's has been augmented into applying Machine Learning algorithms for enhancing language processing [2]. Furthermore, the flow of NLP follows to the Retrieval-Based Models, onto pre-defined responses [2]. Therefore, the NLP structure is followed in the development of the CSUDHBot to use NLP tools from an open source framework called Rasa to achieve communication between user and bot.

CHAPTER 4

SCOPE

CSUDHBot is intended to guide students with information they could easily retrieve with queries upon communication with the bot. The way it works is that the bot starts the conversation after the student types a greeting intent such as "hello". Thereafter, the bot listens to the intent and proceeds to predicting the next action. This action predicted is then uttered by the bot and gives the student two options to choose depending on the student's goal.

The first option provides the student to enter inquiries of problems they may be facing on their own where they may need further assistance. This starts off by the bot performing as a form action that enables students to write inquiries through answering a series of questions that is stored throughout the process in the backend. Once the student completes the inquiry, the bot utters the information stored by the student to reaffirm the information entered is correct.

In the case of the second option, the student is asked if they are an incoming or recurring student. This is meant so the bot can make better predictions that match the student's criteria to provide them with the best responses to their questions. If the student identifies themselves as an "incoming undergraduate" or "incoming graduate" student, the bot identifies their intent and proceeds to take them to a pattern of questions to provide the student with information retrieved from the database. If the student identifies themselves as a "recurring undergraduate" or "recurring graduate" student, they are both joined to a student support system where they are helped with questions they may have such as enrollment dates, courses available for the next semester, financial aid distribution dates or where to apply for financial aid.

CHAPTER 5

GENERAL DESCRIPTION

Product Perspective

The CSUDHBot intends to extend the idea of adding and integrating a chatbot to the school's main website to make accessing resources via the bot an easier experience for students. When the CSUDHBot is matured through a wide range of use cases, the bot can be deployed to the school's main website. This is done by obtaining the school's credentials and adding them to Rasa's credentials page. After the setup, real time messages are recorded from the web socket and can be viewed through Rasa X and gives the ability for anyone who is administrating the page to

view the conversations captured between the bot and the students. Figure 1 and Figure 2 provides an example of how a conversation would be recorded through a chat-widget and Figure 3 and Figure 4 shows how the messages are viewed through the Rasa X graphical user interface (GUI) conversation section.

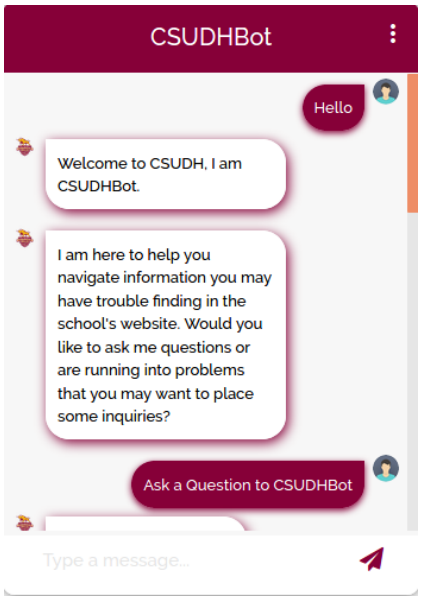


Figure 1: Student (User) starting the conversation with CSUDHBot

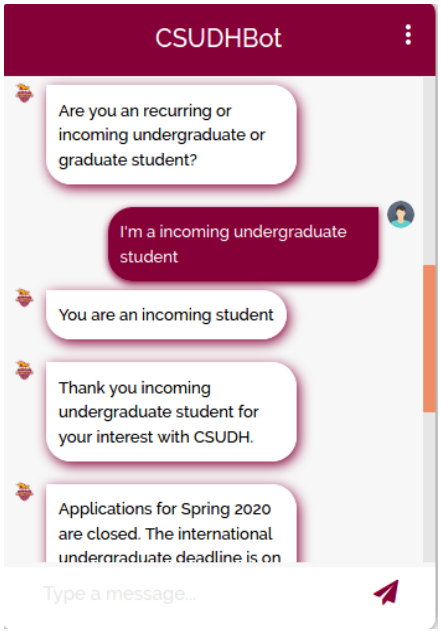


Figure 2: Student (User) messaging CSUDHBot

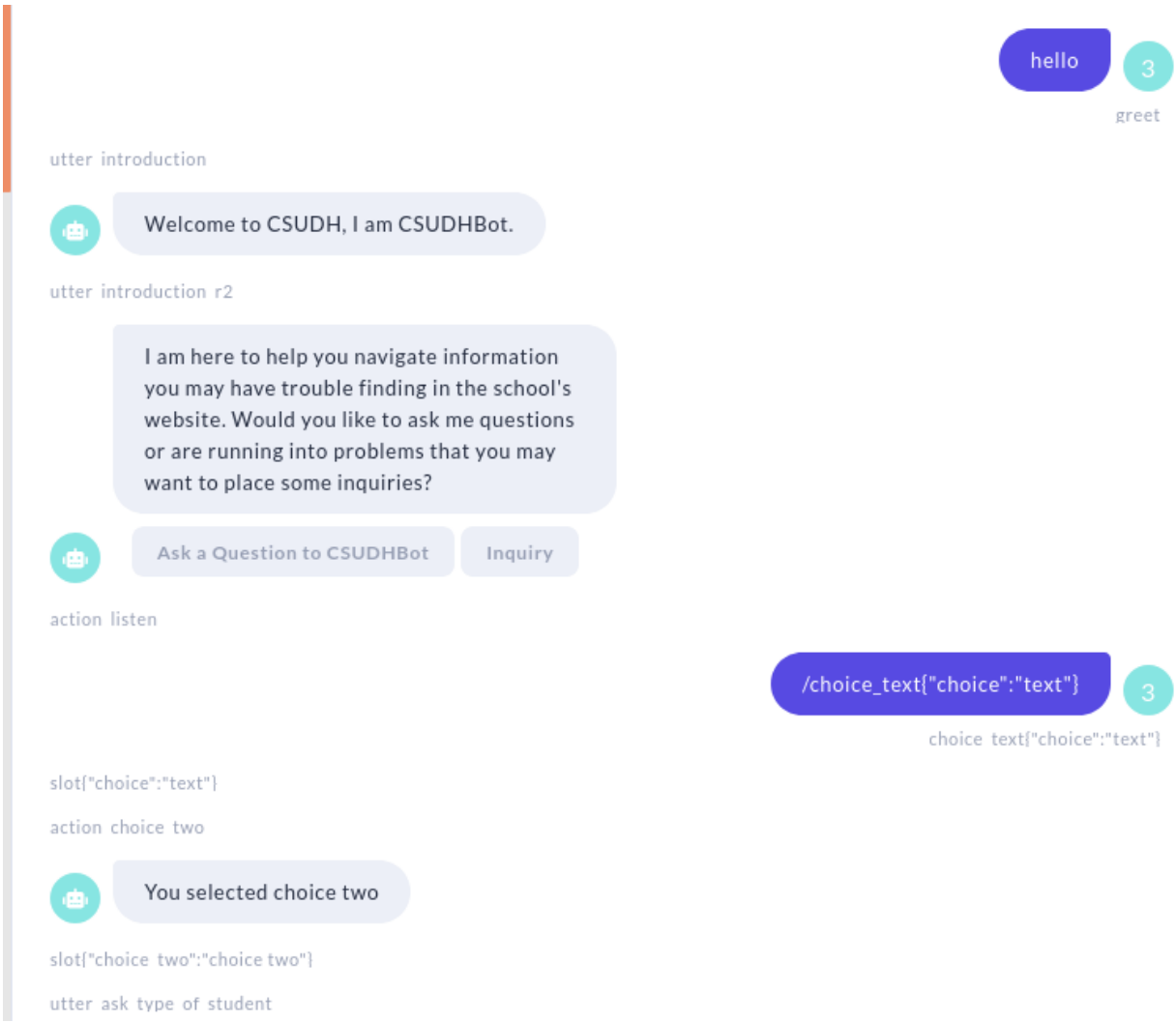


Figure 3: The conversation recorded after the student inputs their messages

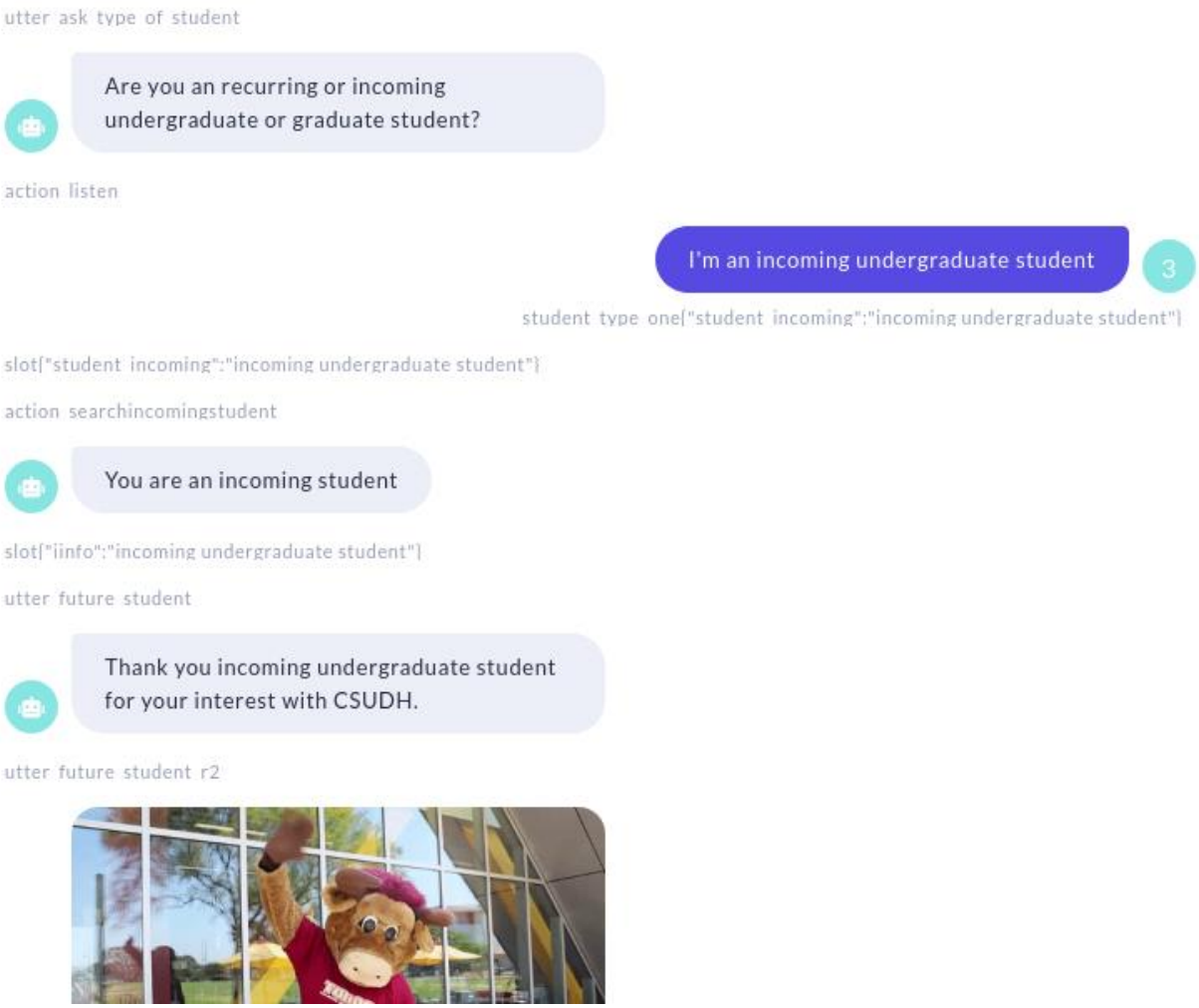


Figure 4: The second part of the conversation recorded from the student's messages

Feasibility Report

The CSUDHBot is currently being run locally, therefore performing a feasibility report with a wide range of subjects was not possible. However, only one subject that remained in close contact during the unprecedented time was able to attempt a conversation with the CSUDHBot. Therefore, a short feasibility report was recorded.

Student conversed with CSUDHBot inquiries:

a. **Feasibility study**: The student input their inquiry to check its ability to record data.
b. **Main evaluation**: The student input inquiries multiple times to check if the data was being recorded.

Student converses with CSUDHBot by searching:

a. **Feasibility study**: The student searched for the graduate program details.
b. **Main evaluation**: The student made different searches.

After performing the tasks, the subject gave feedback over the bot's response. The subject gave positive feedback that the bot was able to take in inquiries and display the data entered from the subject. In addition, the subject provided feedback on the searching where he identified that key words to querying the bot was limited.

User Class and Characteristics

The user classes and characteristics for CSUDHBot are identified in three separate classes. The first class is the incoming and ongoing undergraduate and graduate students followed by the general users, concluding with the administrators in charge maintaining the bot.

1. **Incoming and ongoing undergraduate, graduate students**: The students shall have the ability to use the services readily available upon opening the chat-bot widget through the

website. The technical experience of the users should not matter as the system would be straightforward and easy to use.

2. **General users**: The system will be available to the general user who is not registered through the school to help assist with their queries. No technical expertise is required for use of the bot.
3. **Administrators**: The administrators can maintain the bot by updating any data stored in the database or deleting past student conversations. The administrators are required to have some technical background.

General Constraints

The following contains the general constraints within the CSUDHBot that consists of framework constraints, minimal hardware requirements, linking requirements, IDE requirements, and querying constraints.

1. **Framework constraints**: The Rasa framework used in CSUDHBot is rapidly changing its versions and causes for some of the features to get deprecated with no forward compatibility leading a complete rewrite.
2. **Minimal Hardware Requirements**: With the CPU or Pentium dual core processors, the Tensor flow wheel will not fully be installed and will throw an import error.
3. **Linking Requirements**: Linking spaCy requires the user setting up the environment to be in admin mode where logging in as admin in the system may not be enough and may require the user to run the command prompt in admin mode. If the administrator decides to link spaCy through the Anaconda-Navigator, the admin mode can be run through the “Run as administrator” option inside the environment. The common mistake that occurs is that without admin access, a message appears as “Linking is successful” followed by a status

that identifies an error by prompting that there is not sufficient privilege to perform the operation.

4. **IDE requirements**: Installing an IDE to have Rasa work properly during installation is vital. An IDE that helps the Rasa installation is Visual Studio that requires pre-existing VC++ build tools to solve the RASA packages since they cannot be resolved accurately on their own and may lead to errors such as “ModuleNotFoundError”.
5. **Query Constraints**: The CSUDHBot is limited to the intents or query entered by the user if a word does not exist in the bot’s dictionary. This causes the fallback policy implemented in the bot’s action to notify the user that the query or keyword does not exist and asks the user to attempt again with a different query.

CHAPTER 6

USER REQUIREMENTS DEFITION

Functional Requirements

The following section extends CSUDHBot's functional requirements that includes chatting, searching, and inquiring as well as the bot's own use case of storing data in the tracker-store and an administrator viewing the data stored from the tracker-store in the database.

1. **Chatting**:
    a. The system should allow the users to chat.
    b. The system should inform the user if the query is not available.
    c. The system should respond back to the user.
    d. The system should end the session.

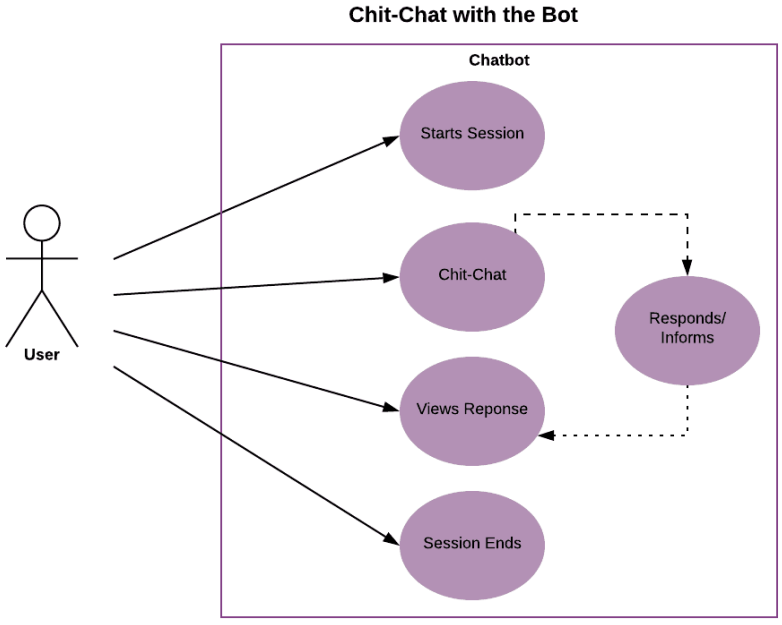


Figure 5: Chit-Chat with the Bot

2. **Searching**:

   a. The system should allow the users to search for information corresponding to their query.

   b. The system should be able to retrieve data from a database for their query.

   c. The user views the data retrieved.

   d. The system ends the session.

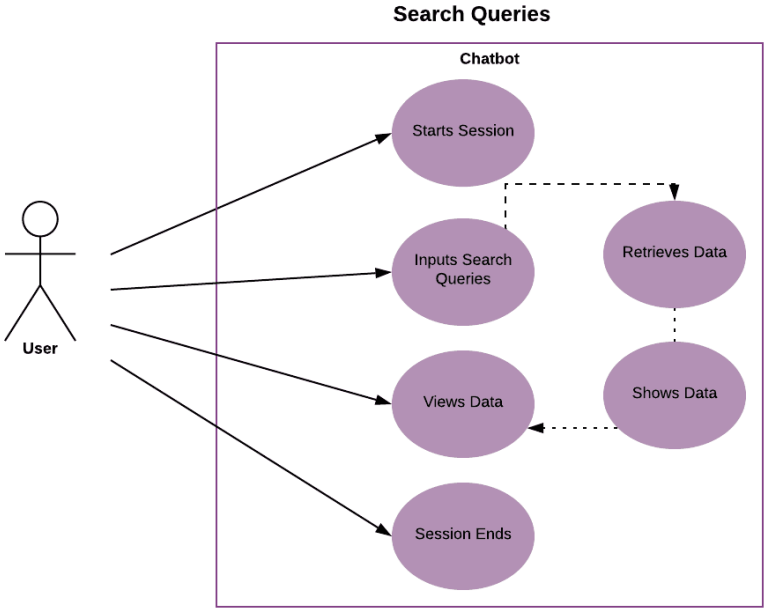


Figure 6: Search Queries

3. **Inquiring**:

   a. The user should be able to make an inquiry of their concern by providing the issues, their name, their phone number and their email.

   b. The system should be able to display the completed inquiry.

   c. The user should be able to review their inquiry.

   d. The system should be able to end the session.

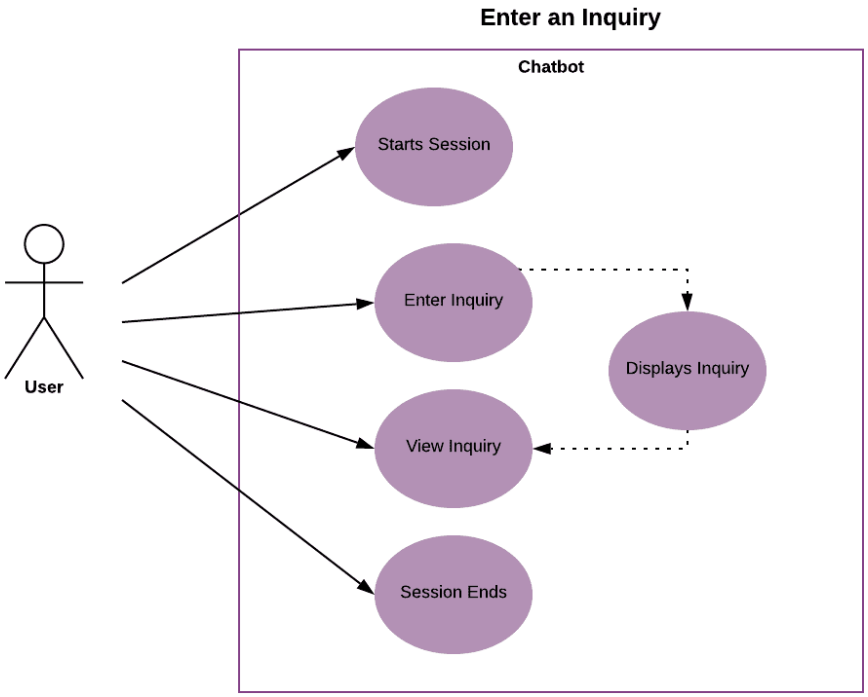


Figure 7: Enter an Inquiry

4. **Stores Data**:

   a. The system's tracker store should store data.

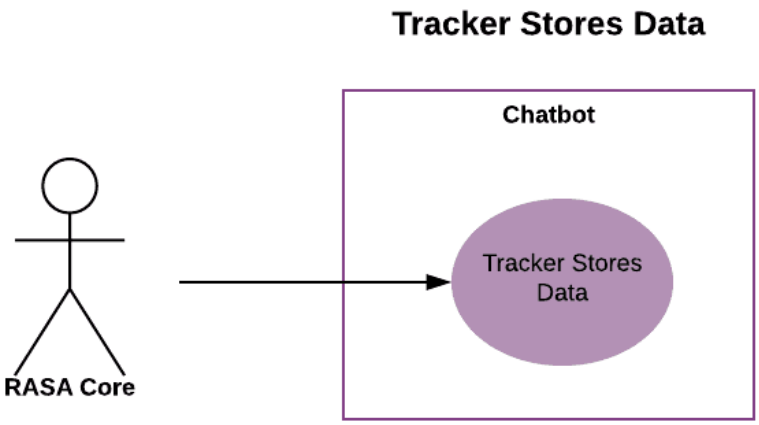


Figure 8: Tracker Stores Data

5. **View Data**:

    a. The system's administrator should be able to view data stored from the system's tracker store through a database.

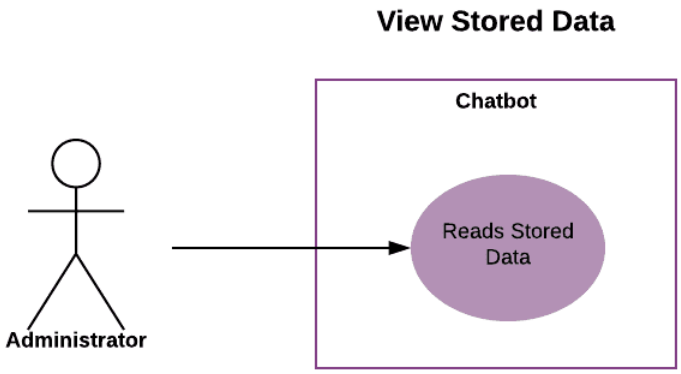


Figure 9: View Stored Data

Non-Functional Requirements

The following section will extend CSUDHBot's non-functional requirements which consists of portability, usability, integrability, and performance.

1. **Portability***:* The CSUDHBot can run in multiple operating systems. The operating systems that the chatbot can run are Linux, Windows 10, and Mac OS. In addition, the chatbot can be enhanced with multiple languages once established in the desired operating system and environment.
2. **Usability**: The CSUDHBot is intuitive since the user can instantly start communicating with the bot when launching the bot through the user interface.

3. **Integrability**: The CSUDHBot can integrate with multiple custom integrations with the REST Input or Callback Input channels. This is done by using the URLs provided by the channels to give the ability to post messages and receive responses directly, or asynchronously via a webhook.
4. **Performance**: The CSUDHBot is able to respond to the user's input quickly. Throughout the interaction, the bot mimics a human with typing indications with minimal delay depending on the integrated widget it is utilizing.

CHAPTER 7

SYSTEM ARCHITECTURE

System Modules

The CSUDHBot is composed of three different modules in the system's architecture. These modules consist of the user interface (UI), Rasa, and the backend.

1. **User Interface Module**: The user interface is where the user will be entering queries. The user also can view the responses made by the bot.

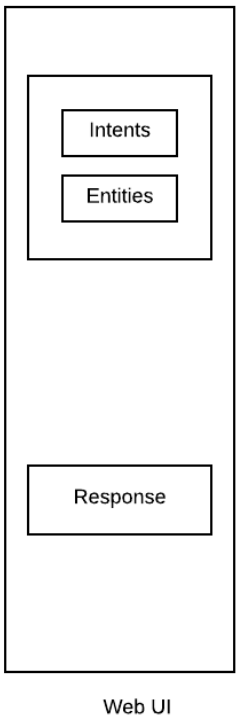


Figure 10: User Interface Module

2. **Rasa X Module**: Rasa X is the primary tool used for the implementation of CSUDHBot. Rasa X is an open source framework for building artificial intelligence (AI) assistants and chatbots. Two main modules exist within Rasa. The first module is the Rasa NLU, which

is meant to understand the user messages and performs intent classification, response retrieval, and entity extraction. The second module is the Rasa Core that holds the conversations and decides what to do next. In a better context, Rasa NLU is where Rasa tries to understand the user message by classifying the intents and entities from the message. The Rasa Core helps in the contextual message flow after it predicts dialogue as a reply and can trigger the Rasa Action Server. These two modules are merged together with Rasa X that helps build the AI Assistants that are powered by the Rasa framework. In addition, the tracker-store stores the data coming from the Rasa NLU and tells the dialogue policy the confidence level to an action, and the dialogue policy sends the Actions a prediction, in which the prediction is then determined by the actions [3].

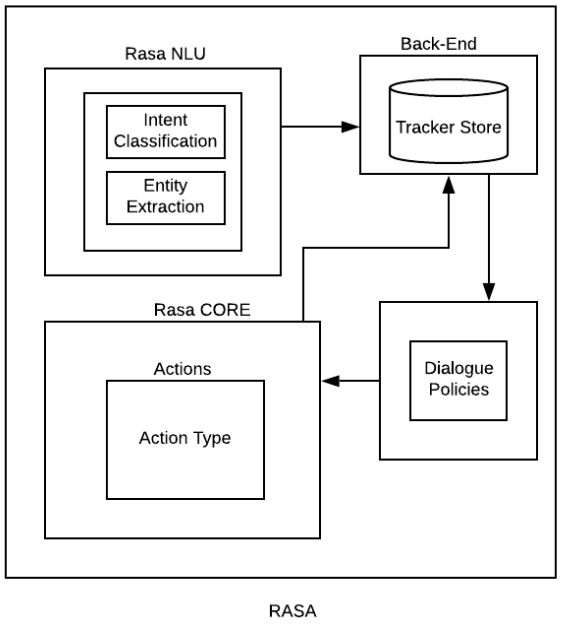


Figure 11: Rasa X Module

3. **Back-End Module**: The back-end module is where the tracker-store stores information. The backend also consists of a database that contains stored information over the school that is retrieved by the user through the actions.

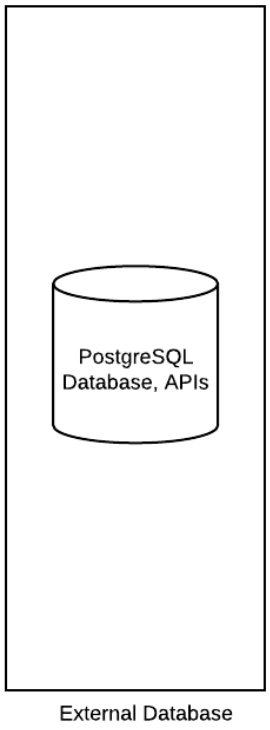


Figure 12: The Backend Module

Software Architecture

CSUDHBot is built in part of the Rasa X microservice architecture. The high-level architecture in

Figure 13 shows the services started when Rasa X is launched to begin interaction with the services. The microservice architecture breaks the services into two main components, which contains Rasa X and Rasa Open Source. Since both are different services, they both contain their own independent databases. The conversation events data flows from the Rasa Open Source to the

Rasa X event broker which allows Rasa X to make API calls to the Rasa Open Source to train and run the models, as well as triggering the conversation events.

Furthermore, the Rasa Open Source trains and runs the NLU. Whenever a conversation is handled by a model, it adds the events to the conversation tracker in the tracker store that collects it in the database [4]. Additionally, this also allows making API calls to the action server to run custom actions possible [4]. The conversation trackers are stored in the tracker store through the database being used in the project, which in this case is stored in the events table from PostgreSQL. When an event broker is configured through the endpoints, the events are then stored in the tracker store. Moreover, Rasa X depends on the Rasa Open Source since it does not work independently and needs Rasa Open Source service for handling conversation data, model training and running [4].

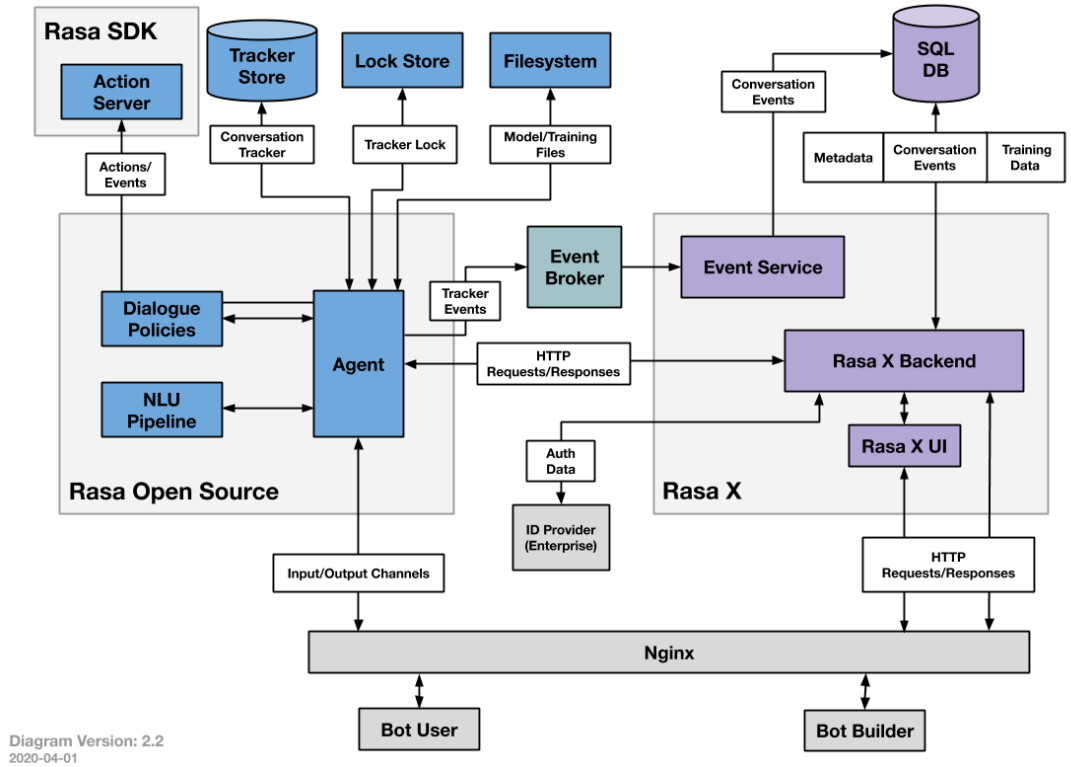


Figure 13: Rasa X and Rasa Open Source Microservice Architecture

In Figure 14, the CSUDHBot uses the architecture from Rasa and shows how it interacts with the overall system containing the user interface (UI) and the database that has the ability to act as an API. The way the architecture works is by taking a message made from the user and passing the intent and entities to the Rasa NLU that acts as an interpreter. The tracker store keeps track of the conversation state and receives the messages from the user. The tracker store sends the current state of the track data to the policy in order to determine the next action. When the policy selects the action based on the confidence which ranges from 0.0 to 1.0, the action is logged by the tracker and a response is sent to the user.

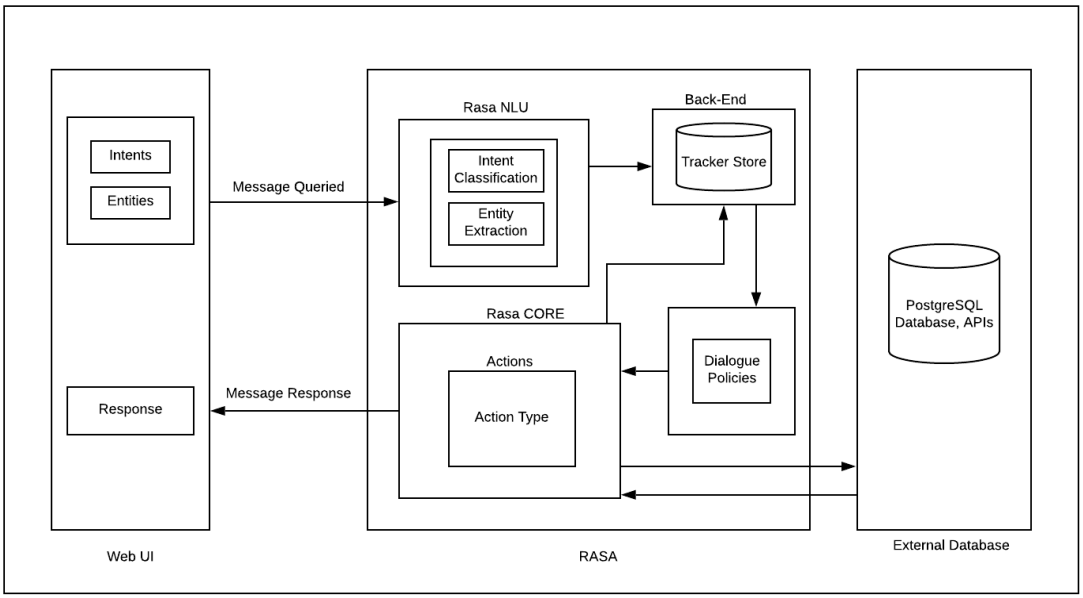


Figure 14: CSUDHBot Architecture

CHAPTER 9

SYSTEM DESIGN

Sequence Diagrams

The sequence diagrams show the interactions made from the user to the bot. A few examples and explanations show how the interactions work.

1. **User Chats with CSUDHBot**: The session starts off with the CSUDHBot listening to the user waiting for their message. The user enters their message that is then taken as an intent through the Rasa NLU and recorded to the tracker-store. The tracker store is part of the PostgreSQL database, and therefore stores the message inside the database management system (DBMS). The database updates, and the tracker-store considers the policy it should use to tell Rasa Core what action it should make [3]. Once the action is determined, the CSUDHBot responds with a predicted message that has been pre-determined based on the confidence level made from the policy. The user views the image, and the CSUDHBot ends the session.

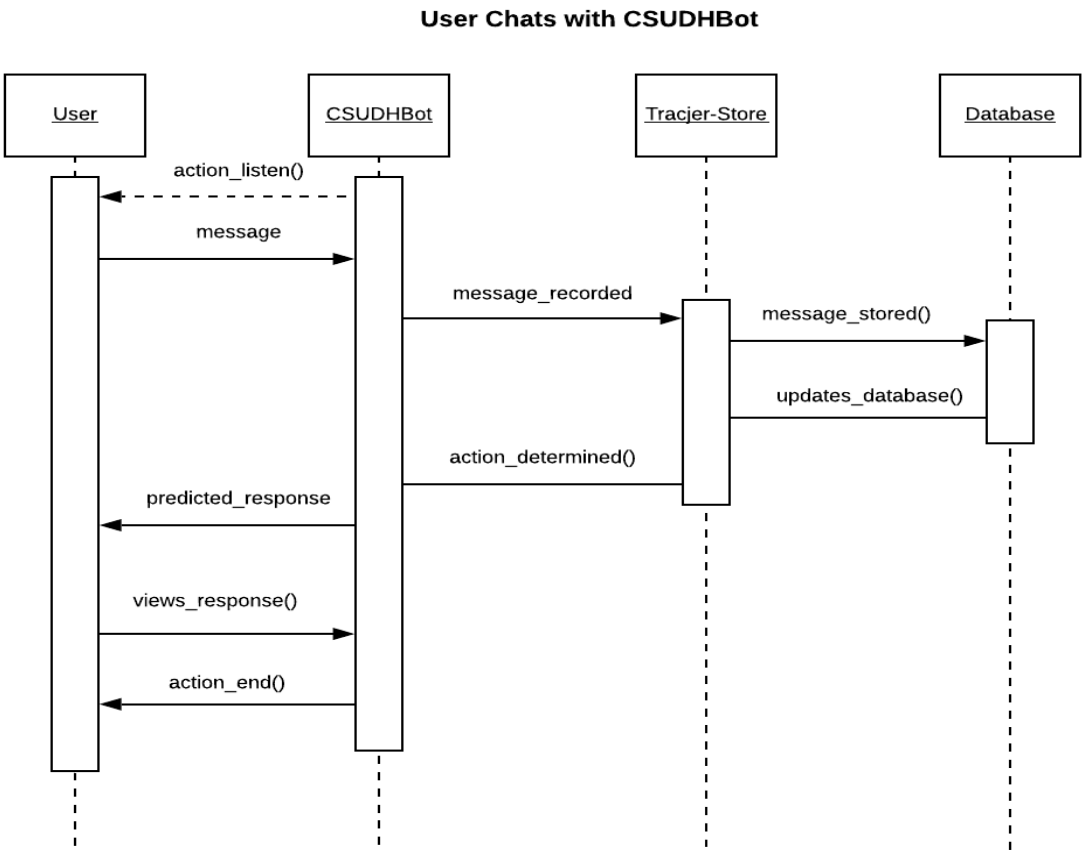


Figure 15: User Chats with CSUDHBot

2. **User Searches through Querying**: The session starts off with the CSUDHBot listening to the user waiting for their message. The user enters their message that is then taken as an intent through the Rasa NLU and recorded to the tracker-store. When the tracker-store takes the intent and identifies that the intent contains an action to retrieve information from the database, the tracker-store communicates with the database to retrieve the data requested from the user. The database retrieves the data and the tracker contains the data from the action determined. The CSUDHBot responds with the retrieved data and presents the data to the user. The user views the response with the data. The sessions are then closed.

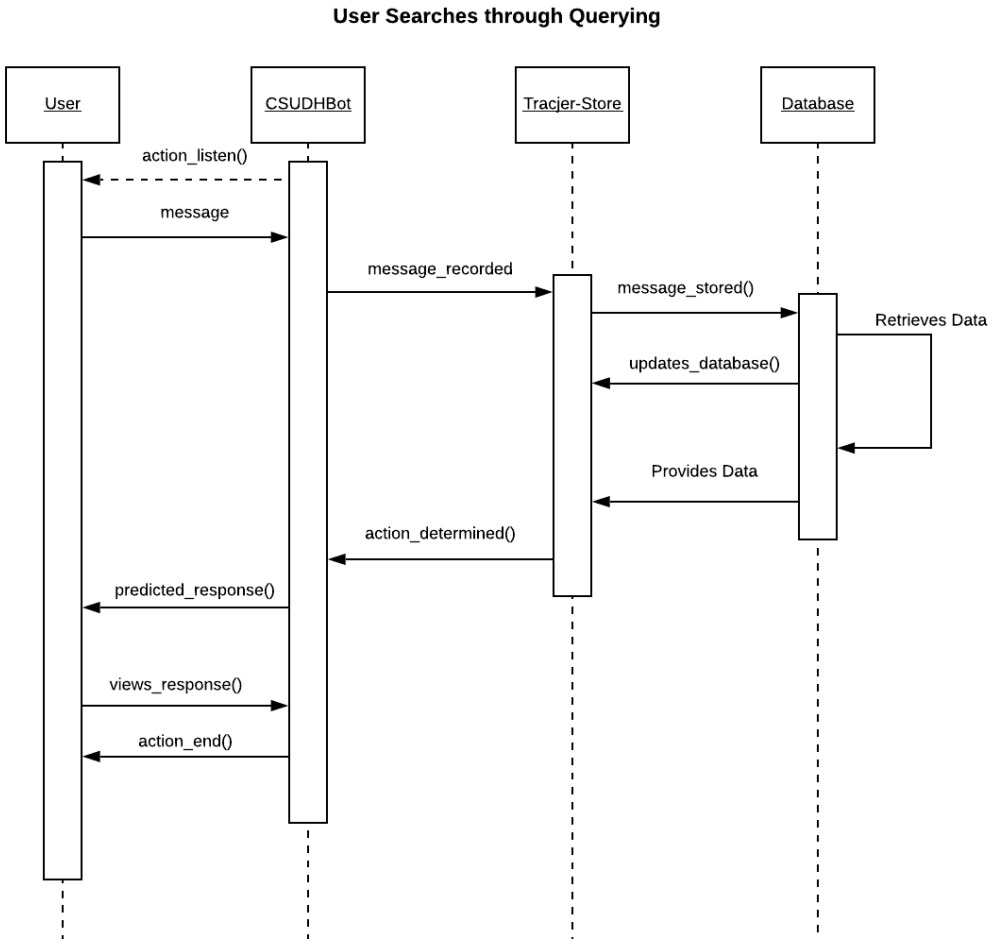


Figure 16: User Searches through Querying

3. **User Makes an Inquiry:** The session starts off with the CSUDHBot listening to the user, waiting for their message. The user enters their message that is then taken as an intent through the Rasa NLU and recorded to the tracker-store. Through the policy, the tracker-store tells the Rasa Core to retrieve the predicted response to the user. The user views the response and starts performing an inquiry through a series of questions. The answers to the questions are then stored to the tracker-store's database. The inquiries are then retrieved and through the predicted action, are displayed to the user's view to reaffirm the inquiry is correct. The session is then closed.

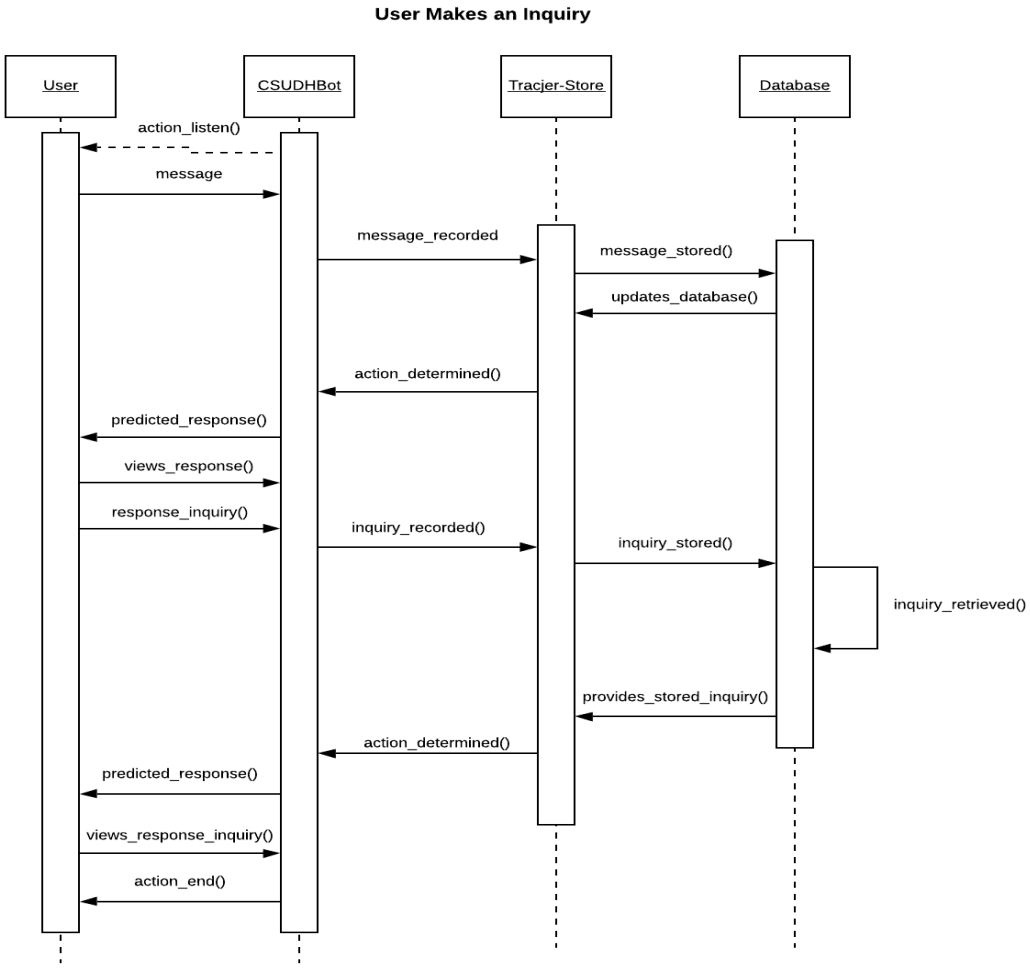
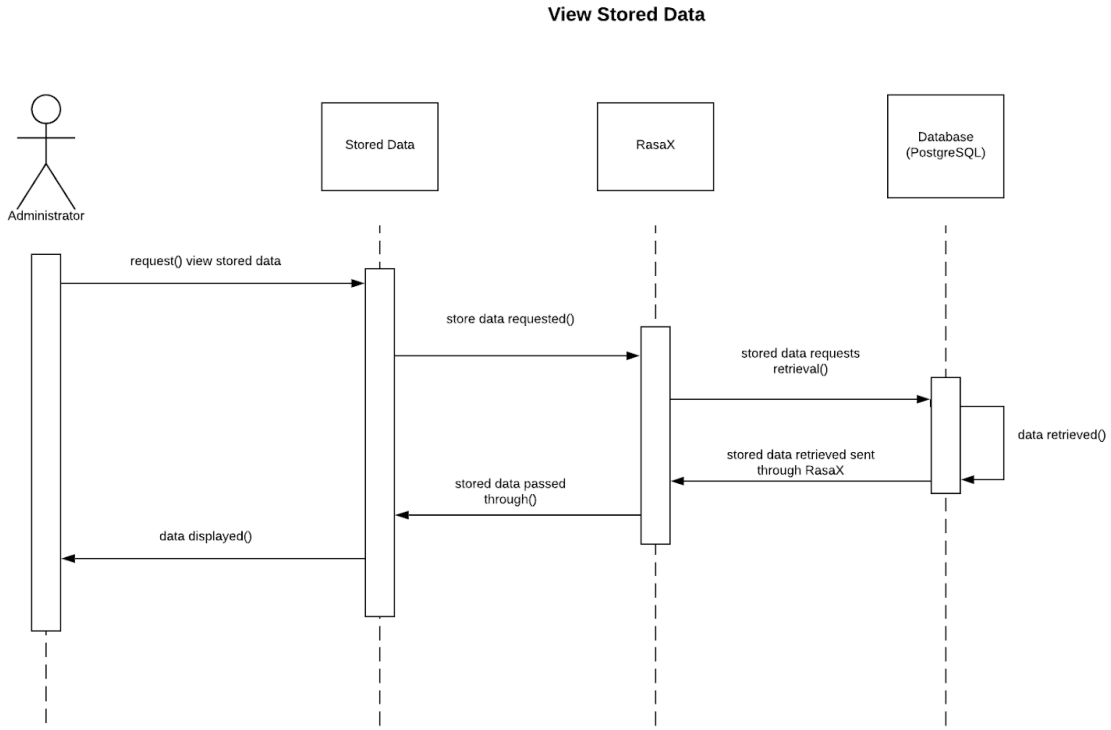
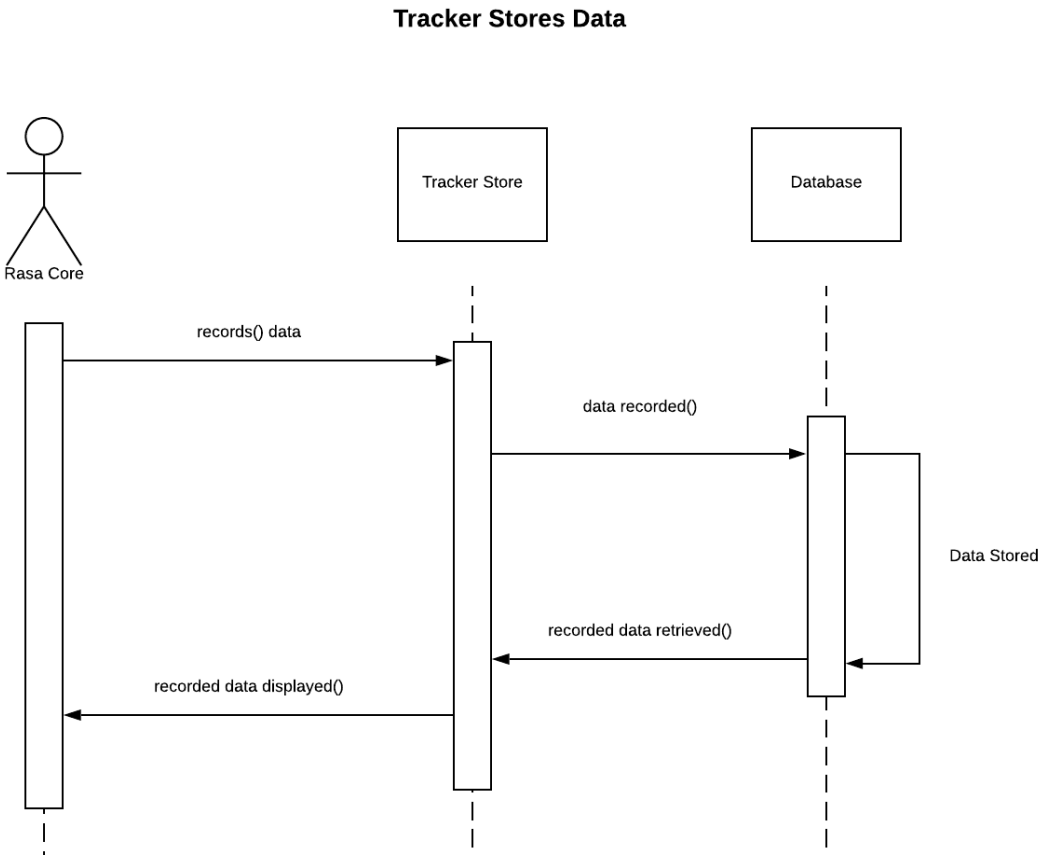


Figure 17: User Makes an Inquiry

4. **Administrator Views Stored Data:** The administrator requests to view the stored data from RasaX through the database. The request passes through RasaX where the request is retrieved through the database. When the stored data is retrieved from the database, the stored data is passed to RasaX and to the administrator so the administrator can view the data.



5. **Tracker Stores Data:** When the Rasa NLU passes the intents and entities to the tracker store, it is recorded into the database. This is the same with Rasa Core when the tracker store communicates with the dialogue pipeline to tell the Rasa Core what action to do next. When that action is determined, the Rasa Core stores data back again to the tracker store and is saved in the tracker's database. The data store can be seen in the database's table when it is opened or queried.



Class Diagrams

The development of a database was essential towards the success of CSUDHBot since CSUDH does not have an API of its own. A REST API was developed to take in the data from the local database but was not ultimately implemented towards the chatbot. Therefore, the chat-bot relied on the PostgreSQL queries. An E-R diagram was designed to display the entities and their relationships used for CSUDHBot, along with the class diagram that defines the attributes and behavioral characteristics of a mock CSUDH system. In addition, the class that stores the data inside the tracker store called ‘events’ will be the first class displayed below.

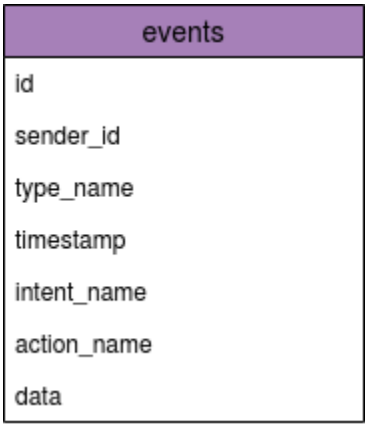


Figure 18: Tracker-Store Events Table

The events class stores the id, sender id, type of name, the timestamp to which the event was recorded, the name of the intent the user made, the action name that will trigger and predict the next action, and the data contained inside.

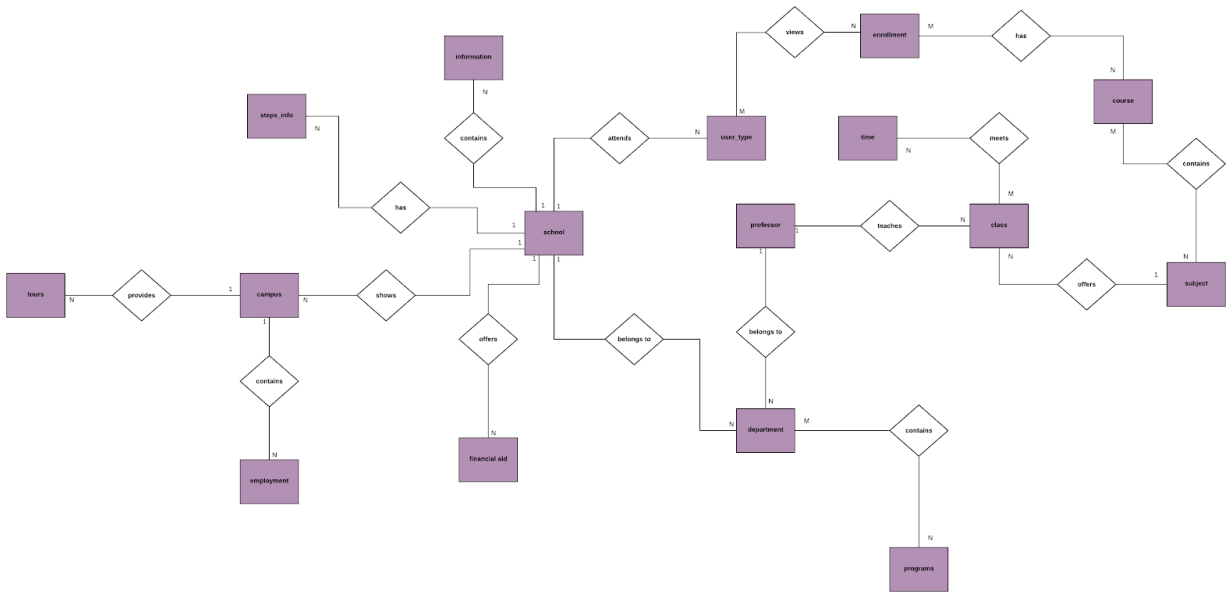


Figure 19: E-R Diagram

The following are the relationship types in the E-R Diagram:

user_type **attends** school; *Cardinality*: N:1;

user_type **views** enrollment; Cardinality: M:N;

department **belongs to** school; Cardinality: N:1;

school **offers** financial aid; Cardinality: 1:N;

school **contains** information; Cardinality: 1:N;

school **has** steps_info; Cardinality; 1:N;

school **shows** campus; Cardinality: 1:N;

campus **provides** tours; Cardinality: 1:N;

department **contains** programs; Cardinality: M:N;

professor **belongs to** department; Cardinality: 1:N;

professor **teaches** class; Cardinality: 1:N;

class **meets** time; Cardinality: M:N;

subject **offers** class; Cardinality: 1:N;

course **contains** subject; Cardinality: M:N;

enrollment **has** course; Cardinality: M:N;

In the following, the relationships between the entities is further explained. The user_type is the entity that identifies the student's user type, whether it'd be an incoming undergraduate student or a recurrent graduate student. The user_type entity has two relationships consisting of school and enrollment. In the relationship between user_type and school, the user_type attends school since the student attends one school, which is CSUDH. In the relationship between user_type and enrollment, the user_type views enrollment to see the semesters that are contained inside of it. The next entity is school, which represents the school CSUDH and contains five relationships. The first relationship is between departments, where the department belongs to school. The department can be anything from the College of Arts and Humanities to the College of Natural and Behavioral Science. The next relationship is between financial, where the school offers financial aid, and it is there where the user_type can view the financial aid that it offers. Next, the relationship is between information, by where the school contains information over the school. Following the information, there exists a relationship between the school that has step

info, which provides the steps over how to apply. The last relationship is between school and the campus, by where the user_type can select which information type from the campus they wish to choose. The entity campus has a relationship with tours, where the campus provides tours to a user_type. The department entity contains the programs, whereas mentioned before, contains the colleges. Furthermore, the entity professor belongs to the department, since a professor could be from a department such as the art department or the computer science department. The entity professor has a relationship with the class since it teaches the corresponding classes from the department. Furthermore, the class has a relationship with time, where the class meets times by providing the times the class is offered. The class consists of another relationship where the subject offers class. Moreover, the subject has a relationship with the course, where the course contains the subject. This could be the course number that is combined with the subject, such as CSC 521 for Advanced Programming Languages. The course also has another relationship, and this is enrollment where the enrollment has courses depending on the semester it is being offered.

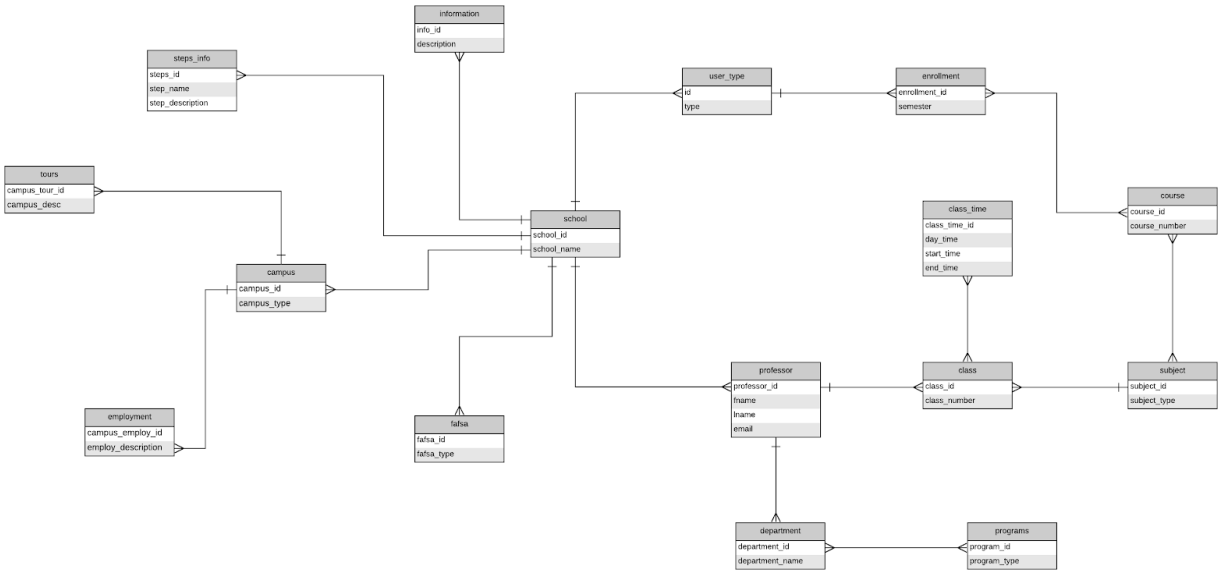


Figure 20: Class Diagram

The class diagram shows the classes and their attributes. A more detailed image is captured to view the classes and the attributes in better detail without their relationships below. The Boyd-Codd normal form (BCNF) was utilized in the classes to do database normalization.

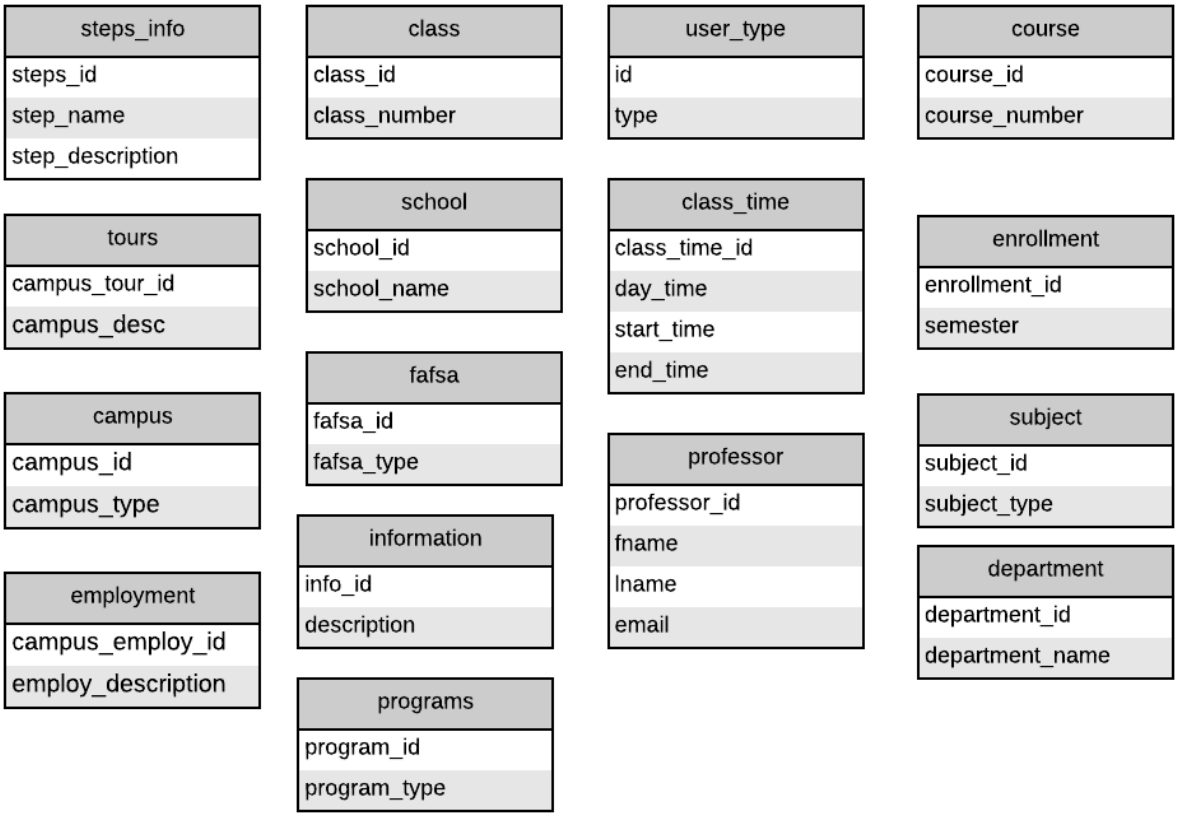


Figure 21: Classes

Class Diagrams

The tables represent the entities from the E-R diagram that were implemented in the database and were retrieved in queries. The statements used to create the tables include their primary keys, attributes, foreign keys, and relationships are found in Appendix B. The following are the tables in the database and their information. To view the relationship tables, they will be included in Appendix C.

School Table: School Table contains 5 different relationships with user type, fafsa, information, steps_info, and campus. Of all of these, the school table contains 5 foreign keys in each of the relationships. Furthermore, the school's table primary key is school_id that has a serial integer value. The only other attribute is school_name that has the character variable set at 150 characters. Only one row was inserted to represent the school, California State University, Dominguez Hills.

Table: school

school_id [PK] serial	school_name [varchar (150)]
1	California State University, Dominguez Hills

Figure 22: School Table

User_Type Table: The user_type table contains two different relationships with school and enrollment. This would require the user_type table to have two foreign keys relating to the relationships. Furthermore, the user_type primary key is id that has a serial integer value and has the attribute called type with the character varying value set to 150.

Table: user_type

id [PK] serial	type [varchar (150)]
1	Incoming Undergraduate Student
2	Incoming Graduate Student
3	Recurrent Graduate Student
4	Recurrent Undergraduate Student

Figure 23: User_Type Table

Department Table: The department table contains three different relationships with school, professor, and programs. The foreign keys for department are a total of three as they derive from the relationships. Furthermore, the primary key in department is department_id with a serial integer and the attribute department_name with the character varying value of 150.

Table: department

department_id [PK] serial	department_name [varchar (150)]
1	College of Arts & Humanities
2	College of Business Administration
3	College of Education
4	College of Extended & International Education
5	College of Health, Human Services, & Nursing
6	College of Natural & Behavioral Science

Figure 24: Department Table

Enrollment Table: The enrollment table only contains two relationships that relate to the user_type and the course and consists of two foreign keys. The primary key for enrollment is enrollment_id which is of a serial integer value, and the attribute semester which is a character varying value set to 150.

Table: enrollment

enrollment_id [PK] serial	semester [varchar (150)]
1	Spring 2019
2	Summer 2019
3	Fall 2019
4	Winter 2019
5	Spring 2020
6	Summer 2020

Figure 25: Enrollment Table

Course Table: The course table only contains two relationships that relate to the enrollment and the subject and consists of two foreign keys. The primary key for course is course_id which is of a serial integer value, and the attribute course_number which is an integer value.

Table: course

course_id [PK] serial	course_number integer
1	101
2	111
3	115
4	116
5	121
6	123
7	221
8	251
9	300
10	301
11	331
12	459
13	461
14	495
15	521
16	531
17	546
18	582
19	584
20	585
21	590
22	595
23	600

Figure 26: Course Table

Subject Table: The subject table only contains two relationships that relate to the course and the class both of which the subject contains their foreign keys. The primary key for subject is subject_id which is of a serial integer value, and the attribute subject_type which is of character varying value set to 150.

Table: subject

subject_id [PK] serial	subject_type [varchar (150)]
1	CSC
2	CTC

Figure 27: Subject Table

Fafsa Table: The fafsa table, which stands for financial aid, only contains one relationship that with school and is contains the school's foreign key. The primary key for fafsa is fafsa_id which is of a serial integer value, and the attribute fafsa_type which is of character varying value set to 150.

Table: fafsa

fafsa_id [PK] serial	fafsa_type [Varchar (150)]
1	grants
2	work-study
3	loans

Figure 28: FAFSA Table

Campus Table: The campus table contains three relationships shared with school, campus_employment and campus tours. The primary key for campus is campus_id which is of a serial integer value, and the attribute campus_type which is of character varying value set to 250.

Table: campus

campus_id [PK] serial	campus_type [Varchar(250)]
1	Campus Employment
1	Campus Tour

Figure 29: Campus Table

Program Table: The program table only contains one relationship that with department. The primary key for program is program_id which is of a serial integer value, and the attribute program_type which is of character varying value set to 150.

Table: program

program_id [PK] serial	program_type [Varchar (150)]
1	Computer Science
2	Computer Technology

Figure 30: Program Table

Class Table: The class table only contains three relationships that with class_time, subject, and professor. The primary key for class is class_id which is of a serial integer value, and the attribute class_number which is an integer value.

Table: class

class_id [PK] serial	class_number integer
1	21044
2	21046
3	21048
4	21052
5	21056
6	21065
7	21071
8	21073
9	21425
10	21429
11	21430
12	21435
13	21439
14	21440
15	21442
16	21444
17	22092
18	22122
19	22123
20	22124
21	22151
22	22153
23	22154
24	22156
25	22157
26	22159
27	20647
28	23907

Figure 31: Class Table

Class_Time Table: The class_time table only contains one relationship with class. The primary key for class_time is class_time_id which is of a serial integer value, and contains the following attributes of day_time with the character varying value of 150, start_time with the time value, and end_time with the time value.

Table: class_time

class_time_id [PK] serial	day_time [Varchar (150)]	start_time TIME	end_timeTIME
1	MoWe	11:30:00	12:45:00
2	TuTh	05:30:00	06:45:00
3	MoWe	04:00:00	05:15:00
4	MoWe	01:00:00	02:15:00
5	MoWe	02:30:00	03:45:00
6	TuTh	07:00:00	08:15:00
7	TuTh	01:00:00	02:15:00
8	Tu	07:00:00	09:45:00
9	Sa	09:00:00	11:45:00
10	Th	07:00:00	09:45:00
11	MoWe	05:30:00	06:45:00
12	Mo	07:00:00	09:45:00
13	Sa	12:00:00	02:45:00
14	Fr	07:00:00	09:45:00
15	We	07:00:00	09:45:00
16	Mo	08:30:00	11:15:00

Figure 32: Class_Time Table

Professor Table: The professor table only contains two relationships which are with department and class. The primary key for professor is professor_id which is of a serial integer value, and contains the following attributes of fname with the character varying value of 150, lname with the character varying value of 150, and email with the character varying value of 200.

Table: professor

professor_id [PK] serial	fname [Varchar (150)]	lname [Varchar (150)]	email [Varchar(200)]
1	Jason	Halasa	jhalasa@csudh.edu
2	Gerald	Fenwick	gmfenwick@csudh.edu
3	Brian	Smith	bsmith@csudh.edu
4	Malcolm	Mccullough	mmccullough@csudh.edu
5	Payman	Khani	pkhani@csudh.edu
6	Howard	Rosenthal	hrosenthal@csudh.edu
7	Marek	Suchenek	msuchenek@csudh.edu
8	Khondaker	Salehim	ksalehim@csudh.edu
9	Mehrdad	Sharbaf	msharbaf@csudh.edu
10	Roman	Tankelevich	rtankelevich@csudh.edu
11	Brad	Hollister	bhollister@csudh.edu
12	Jianchao	Han	jhan@csudh.edu
13	Mohsen	Beheshti	mbeheshti@csudh.edu
14	Bin	Tang	btang@csudh.edu
15	Alireza	Izaddoost	aizaddoost@csudh.edu
16	Bhrigu	Celly	bcelly@csudh.edu
17	Liudong	Zuo	lzuo@csudh.edu

Figure 33: Professor Table

Campus_Tours Table: The campus_tours table only contains one relationship with campus. The primary key for campus_tours is campus_tour_id which is of a serial integer value and the attribute campus_desc with the character varying value set to 600.

Table: campus_tours

campus_tour_id [PK] serial	campus_desc [Varchar(600)]
1	Individual students interested in attending the university are encouraged to bring their families and join us for a guided 55-minute walking tour of the campus. This tour is designed to familiarize the student with opportunities that CSUDH can offer. Tour guides will share about important offices and resources, and will help the student become familiar with this university.
2	Prospective tours are offered weekdays only.
3	A confirmation and reminder email will provide further tour details, including tour meeting location.
4	An $9 parking permit may be purchased at any of our university parking lots.
5	Groups and organizations are welcome to join us for a guided 55-minute walking tour of our campus. Advanced reservations are required for group tours. To request a group tour, please email campustours@csudh.edu. Due to an overwhelming response for group tours and tour guides educational obligations, group tours are scheduled based on the availability of our tour guides.
6	Tours starts promptly at time of the confirmed reservation. Plan to arrive 15 minutes before the start time to check in.

Figure 34: Campus_Tours Table

Campus_Employment Table: The campus_employement table only contains one relationship with campus. The primary key for campus_employment is campus_employment_id , with serial integer value and the attribute employ_description with the character varying value set to 600.

Table: campus_employment

8. Attend area job fairs sponsored by cities, newspapers, and private groups	employ_description
1	1. Use the services of the Career Center
2	a. On-line job listings at CSUDH Handshake
3	b. Website directory of other online job sites
4	c. On-Campus Interviews for graduating seniors and intern candidates
5	d. Employer Outreach Tables and Information Sessions
6	2. Pursue networking opportunities for your field
7	a. Professional association memberships
8	b. Create a profile on LinkedIn or other social networking websites
9	c. Seminars, conferences, and open-house events for your field
10	d. Informational Interviews
11	3. Subscribe to trade and professional association job sites in your field
12	4. Pursue internal advancement options
13	a. Temporary to permanent positions
14	b. Volunteer work or community service
15	c. Interships
16	d. Promotion
17	5. Review website job listings of your employees of choice
18	6. Target unsolicited resumes to hiring managers in selected organizations of choice
19	7. Place your resume on Internet resume databases
20	8. Attend area job fairs sponsored by cities, newspapers, and private groups
21	9. Contact Staffing Agencies (permanent and temporary job placement) and Search Firms
22	10. Visit an Employment Development Department: Workforce Services Branch

Figure 35: Campus_Employment Table

CHAPTER 9

INTERFACE DESIGN

System Interfaces

The CSUDHBot consists of integration of multiple systems. These systems communicate with each other when the user is interacting with the bot. In the prior section, the CSUDHBot's system architecture uses Rasa's microservice architecture and shows how the system interfaces interact with each other along with the microservice architecture. The system interfaces that interact with the Rasa microservice architecture consist of the user interface and the external databases. Figure 36 simplifies the system interface architecture design and shows how the interfaces communicate with one another.

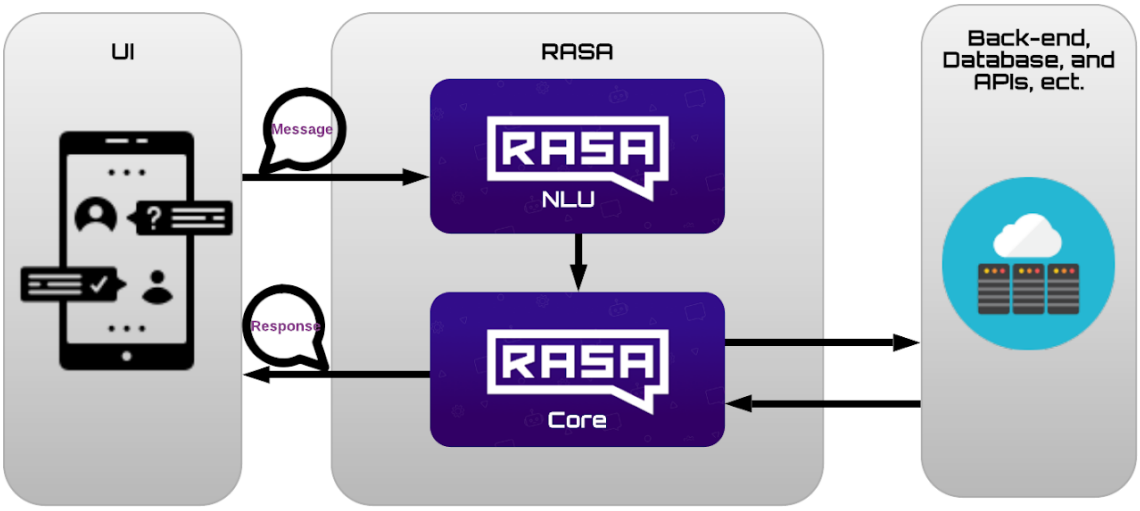


Figure 36: System Interface Architecture Design

User Interfaces

The CSUDHBot utilizes three types of user interfaces that allows the user to interact with the chat-bot. These user interfaces include the command-line interface, Rasa X user interface (UI), and the chat-widget used for a user interface (UI) mock website.

1. Command-line interface (CLI): The command-line interface from Rasa contains simple commands for common tasks that allows the administrator to create a project, train models using the NLU data and stories, loading train models combined with the actions server to communicate with bot through the command line, visualize the stories, and test the trained model using the test NLU data and stories. Figure 37 through Figure 42 displays a few of the CLI commands used when CSUDHBot was developed. Figure 44 shows a sample of the CLI interaction between the user and bot when the command ‘rasa shell && rasa run actions’ is utilized.

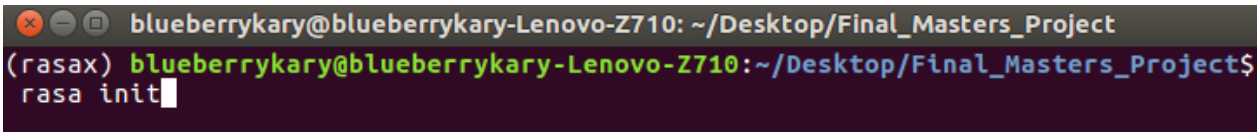


Figure 37: The command 'rasa init' is utilized to create a project

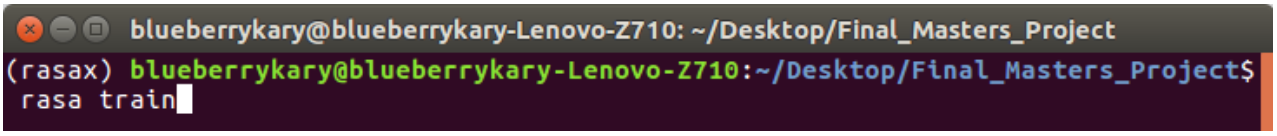


Figure 38: The command 'rasa train' trains the models using the NLU data and stories

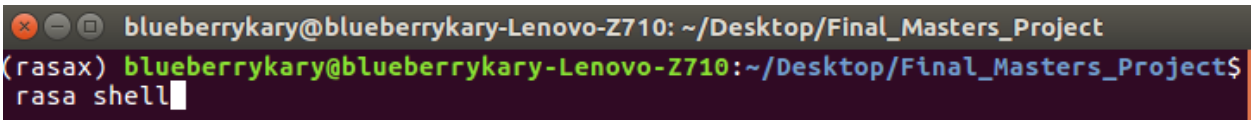


Figure 39: The command 'rasa shell' loads the trained model and allows use to talk to bot

```
blueberrykary@blueberrykary-Lenovo-Z710: ~/Desktop/Final_Masters_Project
(rasax) blueberrykary@blueberrykary-Lenovo-Z710:~/Desktop/Final_Masters_Project$
rasa shell && rasa run actions
```

Figure 40: The command 'rasa shell && rasa run actions' loads the trained model and starts the action server.

```
blueberrykary@blueberrykary-Lenovo-Z710: ~/Desktop/Final_Masters_Project
(rasax) blueberrykary@blueberrykary-Lenovo-Z710:~/Desktop/Final_Masters_Project$
rasa test
```

Figure 41: The command 'rasa test', tests the trained model

```
blueberrykary@blueberrykary-Lenovo-Z710: ~/Desktop/Final_Masters_Project
(rasax) blueberrykary@blueberrykary-Lenovo-Z710:~/Desktop/Final_Masters_Project$
rasa visualize
```

Figure 42: The command 'rasa visualize', visualizes the stories

The rasa visual command retrieves the stories developed for CSUDHBot. The following Figure 43 shows how the visualize command generates the graph.

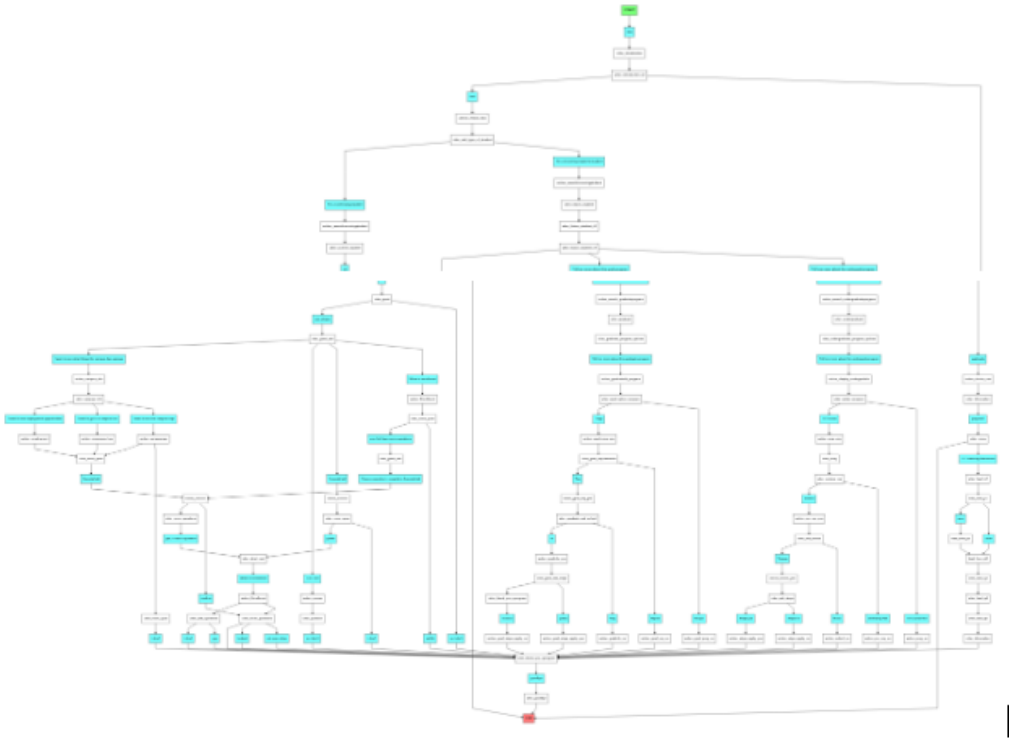


Figure 43: Visualized graphs from the stories

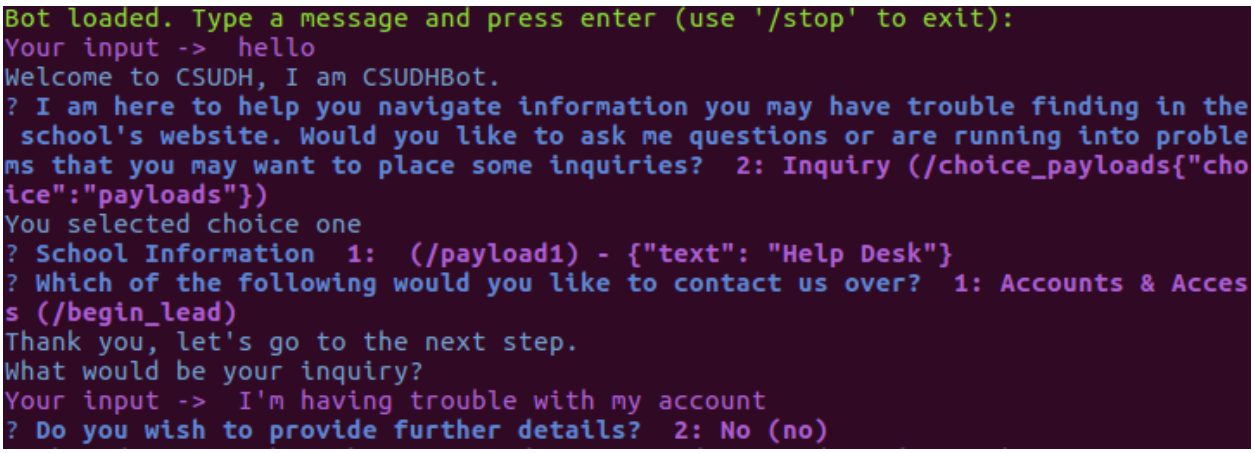


Figure 44: The CLI interaction when the rasa shell and actions run together

2. **Rasa X User Interface (UI):** The Rasa X UI is a web application built on top of Rasa and provides to quickly and easily be able to create and manage the chat-bot, NLU and Core components through the web interface. It also provides features for Rasa to train and load models and monitor the usage and view conversation logs between the user and bot. The Rasa X UI was used throughout the project to view the flow of the story and apply corrections. **Error! Reference source not found.** displays utilization of the Rasa X UI and its features.

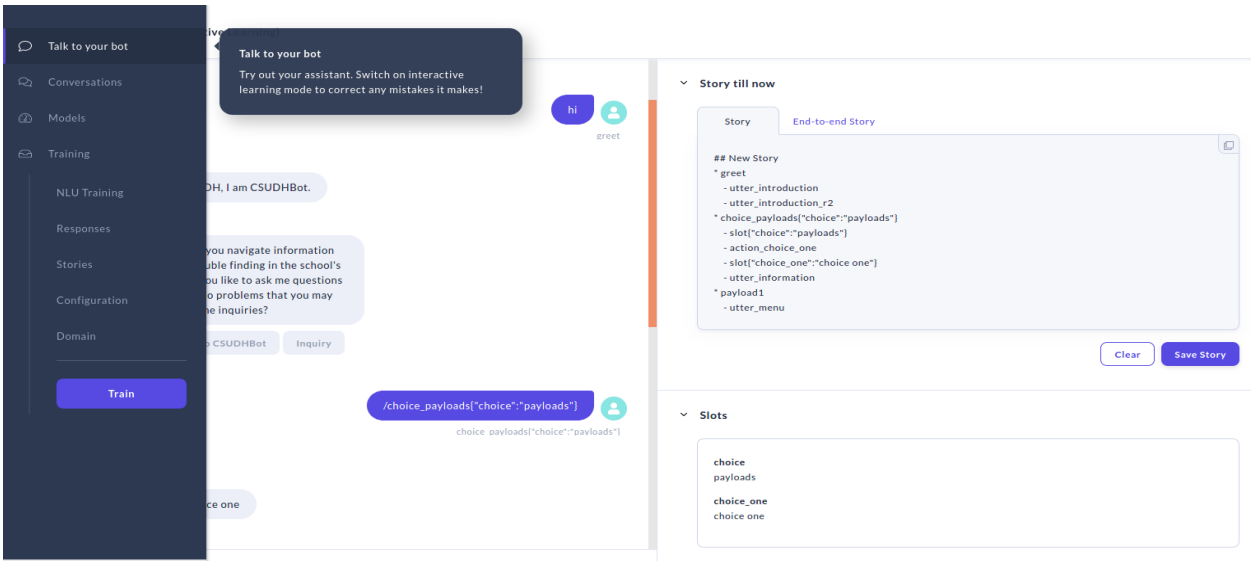


Figure 45: Rasa X UI utilization and its features

3. **Chat-Widget Interface:** The chat-widget connects and deploys the CSUDHBot for users to test out the chat interface. The chat-widget is the main interface used to showcase the interaction between the user and bot. The widget is connected to the project by adding the web socket into the credentials and setting up the widget's script inside the webpage code. Thereafter, the CLI runs the actions server for the bot to respond to user input combined with starting the chat widget by running the command "rasa run -m models --enable-api --cors "*" --debug". Figure 46 and Figure 47 display the commands in the CLI and Figure 11 contains the deployment of the chat-widget. Some of the features included with the widget is the send button and a drop-down button. Other features that can be included when running the bot is text, buttons, images, video, pdf attachments, quick replies, cards carousel, charts, bot typing indicator, and location access. Figure 48 and Figure 49 shows a few of the features mentioned that contains the drop-down menu in which the user can clear the message, restart the communication session, and close the widget. In addition, the send button is on the widget for the user to click and send messages.

```
blueberrykary@blueberrykary-Lenovo-Z710: ~/Desktop/Final_Masters_Project
(rasax) blueberrykary@blueberrykary-Lenovo-Z710:~/Desktop/Final_Masters_Project$
 rasa run -m models --enable-api --cors "*" --debug
```

Figure 46: Running the Rasa web-chat server with CORS headers

```
blueberrykary@blueberrykary-Lenovo-Z710: ~/Desktop/Final_Masters_Project
(rasax) blueberrykary@blueberrykary-Lenovo-Z710:~/Desktop/Final_Masters_Project$
 rasa run actions
```

Figure 47: Running the project's actions

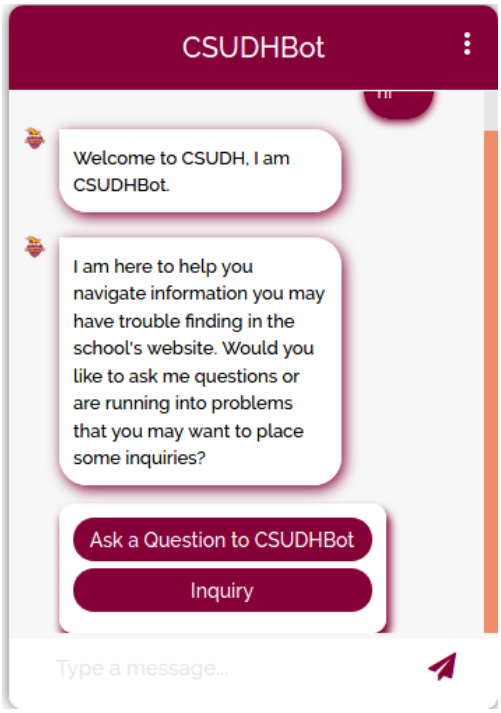


Figure 48: CSUDHBot web-chat interface

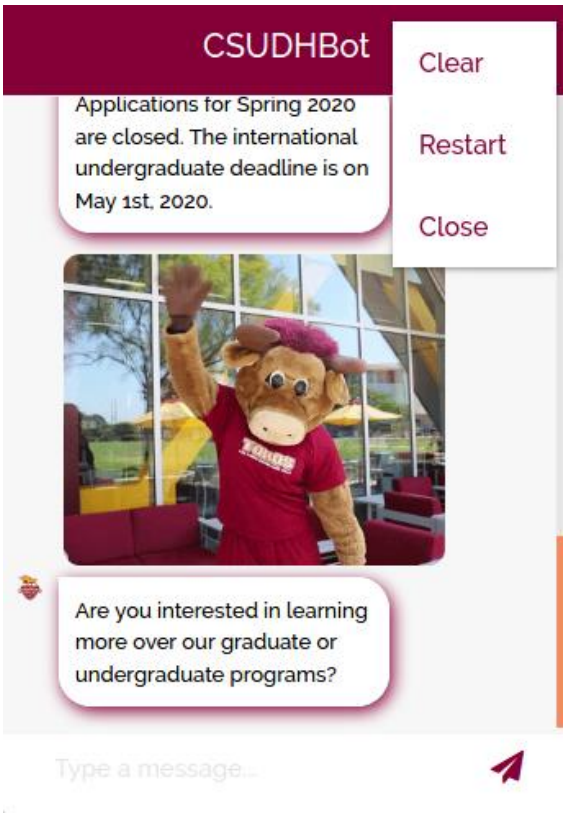


Figure 49: CSUDHBot chat-widget features

Operating System

The CSUDHBot is being run in the Ubuntu 16.04 LTS operating system. The CSUDHBot is portable and can be used in various operating systems such as Windows 10 and Mac OS.

Hardware Interfaces

The CSUDHBot is primarily composed of the software interface and has minimal hardware requirements. The minimal hardware requirements for better performance of the bot is a 2-6v CPUs and 8 GB RAM. At bare minimum, a 4 GB RAM is acceptable and at least 100 GB disk pace would be ideal.

Software Interfaces

The CSUDHBot uses several software interfaces for the success of the project. The first software interface is the Anaconda open-source distribution of the Python programming language that is utilized to set-up a virtual environment dedicated to installing Rasa X. Several packages must be installed in the virtual environment for installation success of Rasa X. The programming languages used for CSUDHBot was primarily python for editing actions, PL/pgSQL (Procedural Language/ PostgreSQL) to values from the school, followed by the JavaScript Object Notation (JSON) to parse query values, YAML which is a human-readable data-serialization language used for file configuration used for user intents and domain, JavaScript and HTML to edit the chat-widget, and Markdown which is a lightweight markup language with plain-text-formatting syntax used for editing stories.

Communication Interface

The communication between CSUDHBot and the web-chat widget follows the client-server model. The client and server utilize a web-socket API through the REST channel in the Rasa credentials for the web-chat service widget to run over HTTP. Since the application is not published publicly, there is no need to serve over HTTP Secure (HTTPS).

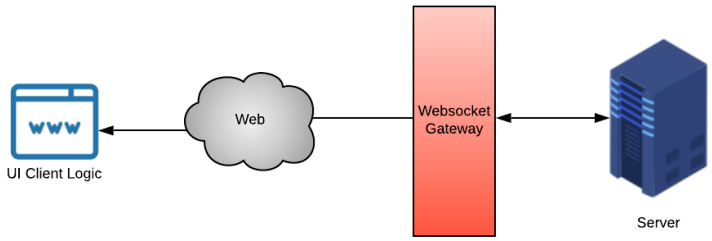


Figure 50: Communication Interface with web-socket

CHAPTER 10

IMPLEMENTATION PROCESS

First Component

The implementation process required five components of implementation for the success of the CSUDHBot. The first component is Rasa's domain that defines how the CSUDHBot operates. The domain specifies the intents, entities, slots, responses, and actions that each were used in the project [5].

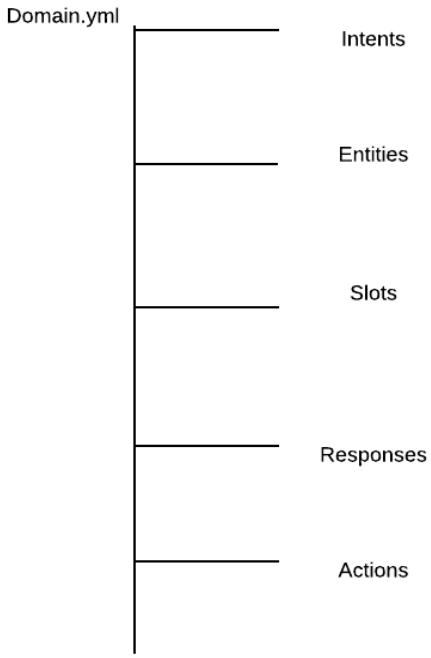


Figure 51: Domain file structure

1. **Intents**: Rasa uses the concept of intents to describe messages should be categorized, and the Rasa NLU deals with the classification [6]. The component used for classifying the intents is the supervised embeddings that is an intent classifier TensorFlow Embedding. This trains word embedding from scratch and supports multiple intents [6]. The classifier

learns separate embedding features and intent vectors. An example from the CSUDHBot project shows a sample snippet of the intents section.

```
session_config:
  session_expiration_time: 60
  carry_over_slots_to_new_session: true
intents:
- greet
- goodbye
- affirm
- deny
- bot_challenge
- current_student
- future_student
- instructor
- search_student_one
- search_student_two
- search_instruct
- go_back
- links
- links2
- links3
- links4
- links5
- schedule_appointment
- University
- By_Major
- back
- choice_payloads
- backtrace
- choice_text
- mood_great
- student_type_two
- student_type_one
- choose_incoming_undergrad_program
- choose_incoming_grad_program
- undergrad_programs
- undergrad_programs_info
- undergrad_programs_requirements
- undergrad_programs_school
- undergrad_programs_apply
- undergrad_programs_where_apply
- undergraduate_programs_deadline
- continue
```

Figure 52: Intents Example from domain file in CSUDHBot

2. **Entities**: Entities are extracted when the user sends messages to the Rasa NLU that get vital pieces of information [7] . When the entity is extracted, it is returned as a dictionary. Moreover, in the CSUDHBot, the spaCy library was utilized for entity recognition [7].

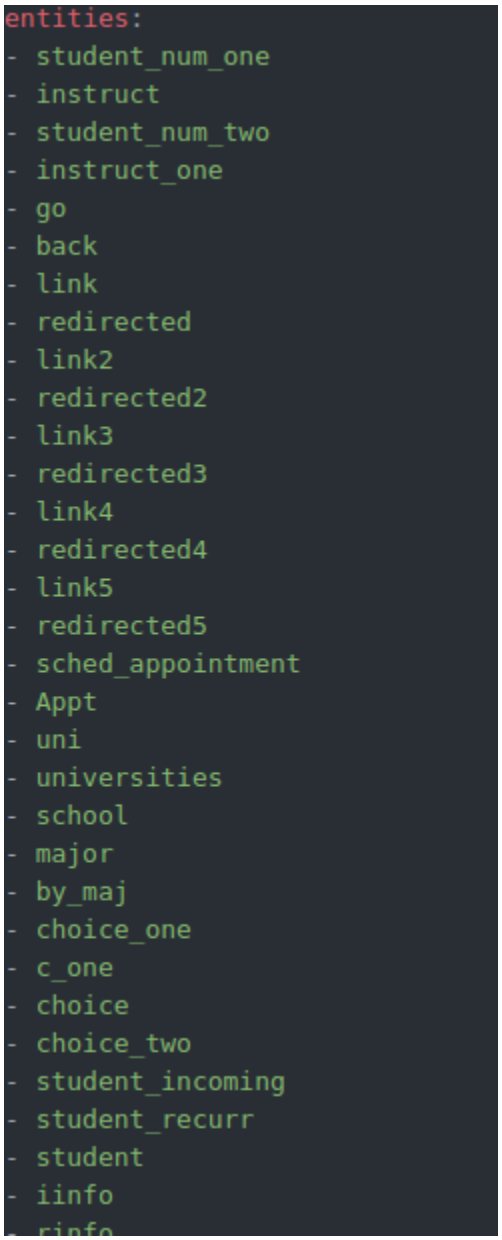


Figure 53: Entities in the domain file for CSUDHBot

3. **Slots**: Slot's in the CSUDHBot is the bot's memory [8]. It stores information provided from the user as well as information gathered from a database query. The goal for using

slots in the project was to influence the dialogue process to determine where the conversation would lead [8]. The most commonly used type of slot in the CSUDHBot was text that tells the Rasa Core whether the slot has a value.

Figure 54: Slots in the domain file for CSUDHBot

4. **Responses**: Another feature that was used heavily is the responses, or in other words, the template that defined what the bot could say to the user [5].

```
utter_menu:
- text: Which of the following would you like to contact us over?
  buttons:
  - title: Accounts & Access
    payload: /begin_lead
  - title: Hardware & Software
    payload: /begin_lead
  - title: IT Continuity Resources
    payload: /begin_lead
  - title: Network
    payload: /begin_lead
  - title: Teaching & Learning
    payload: /begin_lead
  - title: TV & Media Production
    payload: /begin_lead
  - title: University Printing Services

utter_lead_q1:
- text: Thank you, let's go to the next step.

utter_ask_inquiry:
- text: What would be your inquiry?

utter_more_details:
- text: Do you wish to provide further details?
  buttons:
  - title: Yes
    payload: yes
  - title: No
    payload: no

utter_ask_details:
- text: Please type in further details over your inquiry here.

utter_ask_timeline:
- text: When do you wish to be contacted?
  buttons:
  - title: Immediately
    payload: Immediately
  - title: Next Week
    payload: Next Week
  - title: Until the next available time
```

Figure 55: Responses in the domain file for CSUDHBot

5. **Actions**: Actions in the CSUDHBot are the things that the bots could do. Some of the actions that it could do is respond to the user, make external API calls, or query a database. The actions module path had to be added in the domain along with the action's slots [5].

```
actions:
- utter_introduction
- utter_introduction_r2
- utter_cheer_up
- utter_did_that_help
- utter_happy
- utter_goodbye
- utter_iamabot
- utter_view
- action_search_student_one
- action_search_student_two
- action_search_instruct
- utter_next
- action_goback
- action_goback2
- action_goback3
- action_goback4
- action_goback5
- action_schedule_appt
- action_schedule_university
- action_schedule_major
- action_choice_one
- action_choice_two
```

Figure 56: The actions in the domain file for CSUDHBot

Second Component

The second component in the implementation process was the actions of the project. The actions are referenced by the domain, but custom actions were created to retrieve data from the database to provide the data to the user. The first objective was to create the database and connect the database to the project. After connecting to the database, the custom actions were implemented to get the data from the database. Inside the actions, PostgreSQL statements had to be implemented

to call on the database to retrieve the information. The JSON encoder and decoder had to be used to parse the data from the database since the data was being read as integers. Therefore, the json loads and dumps methods were used to return the strings.

```python
# Connect to PostgreSQL database
import psycopg2
import traceback

def getData(query:str):
    try:
        mydb = psycopg2.connect(
        host = "localhost",
        database = "api",
        user = "postgres",
        password = "Ks126352"
        )

        cursor = mydb.cursor()
        cursor.execute(query)

        results = cursor.fetchall()
        return results
    except:
        print("Error occured while connecting to database or fetching data from database. Error Trace: {}".format(traceback.format_exc()))
        return []
```

Figure 57: PostgreSQL connection to CSUDHBot Project

```python
class Action_view_summer2020_course1(Action):

    def name(self) -> Text:
        return "action_view_summer2020"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #school information slot
        summer_c1 = tracker.get_slot("summer_2020_course1")

        q = 'select  e.semester, s.subject_type, c.course_number, cl.class_number, t.day_time, t.start_time, t.end_time, p.fname, p.lname, p.emai`
        hc.course_id = c.course_id FULL OUTER JOIN contains_subject as csub ON c.course_id = csub.course_id FULL OUTER JOIN subject as s ON csub.s
        professor as p ON tt.professor_id = p.professor_id WHERE ut.id = 1 AND e.enrollment_id = 6 AND t.class_time_id = 17 AND c.course_id = 24 A
        #pass the sql query to the getData method and store the results in `data` variable.
        data = getData(q)


        #Print out the course details for 595
        opt = "============================="
        details = "Course Details"
        set_one = str(data[0][0])
        set_two = str(data[0][1])
        set_three = str(data[0][2])
        set_four = "    Class Number = " + str(data[0][3])
        set_five = "On " + str(data[0][4]) + " at " + str(data[0][5]) + " to " + str(data[0][6])
        set_six = "    Professor: " + str(data[0][7]) + " " + str(data[0][8]) + "\n" + "   Email: " + str(data[0][9])
        opt2 = "============================="

        set_combined  = opt + "<br/><br/>" + details + "<br/><br/>" +  set_one + "<br/><br/>" + set_two + "<br/><br/>" + set_three
        set_combined2 = " " + set_four + "<br/><br/>" + set_five + "<br/><br/>" + set_six + "<br/><br/>" + opt2

        set_com = set_combined + set_combined2
```

Figure 58: Actions code to retrieve data from database

Third Component

The third component in the implementation process was in Rasa's Core stories. This is where the dialogue representation between the user and the CSUDHBot took place. Stories are a form of training data used to train in the Rasa Core dialogue management models. The stories implemented start off with a prefix '##' to specify the purpose of the story structure [9]. The user input start with the '*' followed by the intent name [9]. The responses and actions that come from the CSUDHBot are prefixed with '-'. Moreover, stories in the CSUDHBot learn from the current user input and the previous states of conversations that influences the prediction of the next response.

```
## Story 1a: The Incoming Graduate Student WANTS more details on the grad programs
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_one{"student_incoming":"incoming student", "student_incoming":"prospective student", "student_incoming": "applicant", "student_incoming":"incoming
undergraduate student", "student_incoming":"incoming undergrad student", "student_incoming": "incoming graduate student", "student_incoming": "incoming grad student",
"student_incoming": "incoming freshman", "student_incoming":"transfer student", "student_incoming":"incoming first year student", "student_incoming":"incoming master's
student"}
  - action_searchincomingstudent
  - slot{"iinfo":"istudent"}
  - utter_future_student
  - utter_future_student_r2
  - utter_future_student_r3
* choose_incoming_grad_program{"program_graduate":"graduate program", "program_graduate":"grad program"}
  - action_search_graduateprogram
  - slot{"gprograms":"program_graduate"}
  - utter_graduate
  - utter_graduate_program_options
* grad_program_info{"program_masters":"graduate program"}
  - action_graduatedd_program
  - slot{"graduate_program_details":"grad_program"}
  - utter_grad_option_program
* select_grad_program_details_yes{"program_grad_details_yes":"Yupp"}
  - action_grad_prog_yes
  - slot{"graduate_prog_yes":"grad_prog_yes"}
  - utter_grad_requirements
* select_grad_requirements_yes{"program_gradrequirement_details_yes":"Yas"}
  - action_grad_req_yes
  - slot{"graduate_progreq_yes":"grad_progreq_yes"}
  - utter_graduate_ask_school
* select_gradschool_yes{"view_gradschool_yes":"Si"}
  - action_gradinfo_yes
  - slot{"graduateinfo_yes":"gradinfo_yes"}
  - utter_grad_ask_steps
* select_gradsteps_yes{"view_grad_steps_yes":"yasss"}
  - action_grad_steps_apply_yes
  - slot{"graduatesteps_app_yes":"gradsteps_yes"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart
```

Figure 59: Rasa Core Stories for CSUDHBot

Fourth Component

The fourth component in the project that ties all the other files is the Rasa's NLU nlu.md. This file contains all the intents that will be made from the user structured as lists.

```
## intent: student_type_one
- I am an [incoming student](student_incoming)
- I'm a [prospective student](student_incoming)
- I am an [applicant](student_incoming)
- I'm a [incoming undergraduate student](student_incoming)
- I'm an [incoming undergrad student](student_incoming)
- I'm an [incoming graduate student](student_incoming)
- I'm an [incoming grad student](student_incoming)
- I'm an [incoming freshman](student_incoming)
- I'm a [transfer student](student_incoming)
- I'm a [incoming first year student](student_incoming)
- I'm a [incoming master's student](student_incoming)

## intent: student_type_two
- I am an [ongoing student](student_recurr)
- I'm a [freshman](student_recurr)
- I'm a [sophmore](student_recurr)
- I'm a [junior](student_recurr)
- I'm a [senior](student_recurr)
- I'm a [master's student](student_recurr)
- I'm a [first year student](student_recurr)
- I'm a [second year student](student_recurr)
- I'm a [third year student](student_recurr)
- I'm a [fourth year student](student_recurr)
- I'm a [fifth year student](student_recurr)
- I'm a [graduate student](student_recurr)
- I'm a [undergraduate student](student_recurr)
- I am an [international student](student_recurr)
- I'm a [recurring student](student_recurr)
- I'm a [continuing student](student_recurr)

## intent: choose_incoming_undergrad_program
- I am interested in learning more about the [undergraduate program](program_undergrad)
- Tell me more about the [undergrad program](program_undergrad)
- I want to know more about the [undergraduate program](program_undergrad)
```

Figure 60: Rasa NLU Intents for CSUDHBot

Fifth Component

The fifth part that was implemented and set up is the tracker-store broker that stores conversations into a database. The database used to store all the data from the tracker is PostgreSQL.

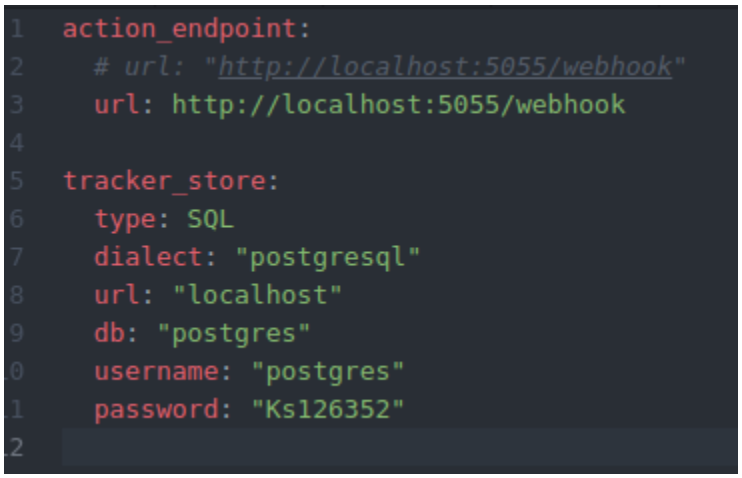
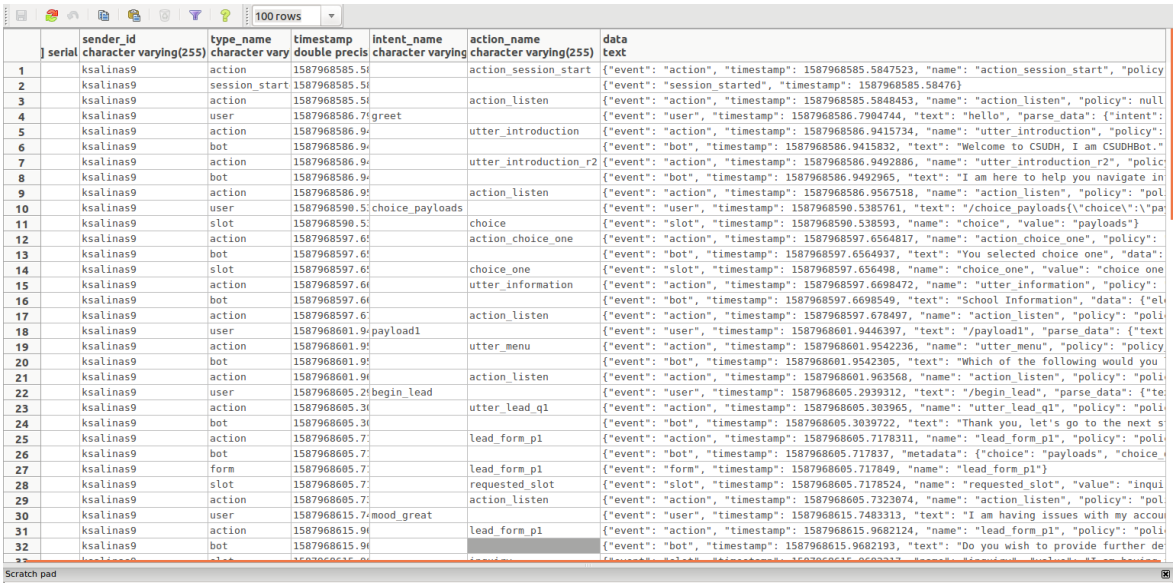


Figure 61: Tracker-Store using PostgreSQL

After the broker is set-up, when conversations take place between the user and CSUDHBot, the conversations are stored inside the database in a table called ‘events’.



100 rows

	] serial	sender_id character varying(255)	type_name character vary	timestamp double precis	intent_name character varying	action_name character varying(255)	data text
1		ksalinas9	action	1587968585.5		action_session_start	{"event": "action", "timestamp": 1587968585.5847523, "name": "action_session_start", "policy
2		ksalinas9	session_start	1587968585.5			{"event": "session_started", "timestamp": 1587968585.58476}
3		ksalinas9	action	1587968585.5		action_listen	{"event": "action", "timestamp": 1587968585.5848453, "name": "action_listen", "policy": null
4		ksalinas9	user	1587968586.7	greet		{"event": "user", "timestamp": 1587968586.7904744, "text": "hello", "parse_data": {"intent":
5		ksalinas9	action	1587968586.9		utter_introduction	{"event": "action", "timestamp": 1587968586.9415734, "name": "utter_introduction", "policy":
6		ksalinas9	bot	1587968586.9			{"event": "bot", "timestamp": 1587968586.9415832, "text": "Welcome to CSUDH, I am CSUDHBot."
7		ksalinas9	action	1587968586.9		utter_introduction_r2	{"event": "action", "timestamp": 1587968586.9492886, "name": "utter_introduction_r2", "polic
8		ksalinas9	bot	1587968586.9			{"event": "bot", "timestamp": 1587968586.9492965, "text": "I am here to help you navigate in
9		ksalinas9	action	1587968586.9		action_listen	{"event": "action", "timestamp": 1587968586.9567518, "name": "action_listen", "policy": "pol
10		ksalinas9	user	1587968590.5	choice_payloads		{"event": "user", "timestamp": 1587968590.5385761, "text": "/choice_payloads{\"choice\":\"pa
11		ksalinas9	slot	1587968590.5		choice	{"event": "slot", "timestamp": 1587968590.538593, "name": "choice", "value": "payloads"}
12		ksalinas9	action	1587968597.6		action_choice_one	{"event": "action", "timestamp": 1587968597.6564817, "name": "action_choice_one", "policy":
13		ksalinas9	bot	1587968597.6			{"event": "bot", "timestamp": 1587968597.6564937, "text": "You selected choice one", "data":
14		ksalinas9	slot	1587968597.6		choice_one	{"event": "slot", "timestamp": 1587968597.656498, "name": "choice_one", "value": "choice one
15		ksalinas9	action	1587968597.6		utter_information	{"event": "action", "timestamp": 1587968597.6698472, "name": "utter_information", "policy":
16		ksalinas9	bot	1587968597.6			{"event": "bot", "timestamp": 1587968597.6698549, "text": "School Information", "data": {"el
17		ksalinas9	action	1587968597.6		action_listen	{"event": "action", "timestamp": 1587968597.678497, "name": "action_listen", "policy": "poli
18		ksalinas9	user	1587968601.9	payload1		{"event": "user", "timestamp": 1587968601.9446397, "text": "/payload1", "parse_data": {"text
19		ksalinas9	action	1587968601.9		utter_menu	{"event": "action", "timestamp": 1587968601.9542236, "name": "utter_menu", "policy": "policy
20		ksalinas9	bot	1587968601.9			{"event": "bot", "timestamp": 1587968601.9542305, "text": "Which of the following would you
21		ksalinas9	action	1587968601.9		action_listen	{"event": "action", "timestamp": 1587968601.963568, "name": "action_listen", "policy": "poli
22		ksalinas9	user	1587968605.2	begin_lead		{"event": "user", "timestamp": 1587968605.2939312, "text": "/begin_lead", "parse_data": {"te
23		ksalinas9	action	1587968605.3		utter_lead_q1	{"event": "action", "timestamp": 1587968605.303965, "name": "utter_lead_q1", "policy": "poli
24		ksalinas9	bot	1587968605.3			{"event": "bot", "timestamp": 1587968605.3039722, "text": "Thank you, let's go to the next s
25		ksalinas9	action	1587968605.7		lead_form_p1	{"event": "action", "timestamp": 1587968605.7178311, "name": "lead_form_p1", "policy": "poli
26		ksalinas9	bot	1587968605.7			{"event": "bot", "timestamp": 1587968605.717837, "metadata": {"choice": "payloads", "choice_
27		ksalinas9	form	1587968605.7		lead_form_p1	{"event": "form", "timestamp": 1587968605.717849, "name": "lead_form_p1"}
28		ksalinas9	slot	1587968605.7		requested_slot	{"event": "slot", "timestamp": 1587968605.7178524, "name": "requested_slot", "value": "inqui
29		ksalinas9	action	1587968605.7		action_listen	{"event": "action", "timestamp": 1587968605.7323074, "name": "action_listen", "policy": "pol
30		ksalinas9	user	1587968615.7	mood_great		{"event": "user", "timestamp": 1587968615.7483313, "text": "I am having issues with my accou
31		ksalinas9	action	1587968615.9		lead_form_p1	{"event": "action", "timestamp": 1587968615.9682124, "name": "lead_form_p1", "policy": "poli
32		ksalinas9	bot	1587968615.9			{"event": "bot", "timestamp": 1587968615.9682193, "text": "Do you wish to provide further de

Scratch pad

Figure 62: Events database table

In the last phase of implementation, the chat-widget was set up with the web-socket that was added through Rasa's REST channel in the credentials.yml file for the chat-widget to run on the HTTP protocol.

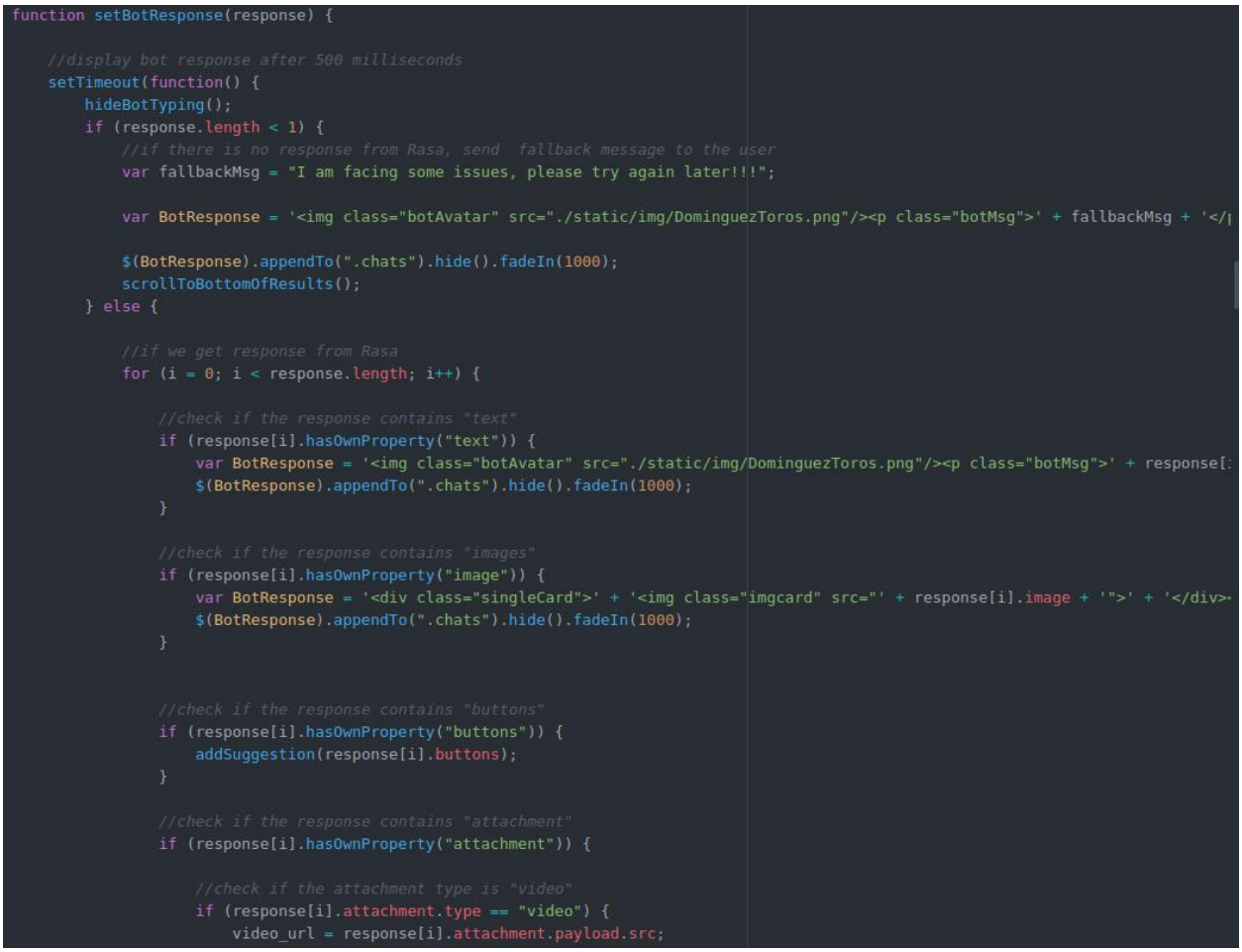
```
rest:

socketio:
  user_message_evt: user_uttered
  bot_message_evt: bot_uttered
  session_persistence: true/false

rasa:
  url: "http://localhost:5002/api"
```

Figure 63: Web-socket connection in the REST Channel

Along with that, the html and JavaScript files were edited to make a visual presentation of the CSUDHBot. These files were provided by the original author of the widget with permission to use it to showcase the chat-bot.

```
<head>
    <title>Chatbot Widget</title>

    <!--Let browser know website is optimized for mobile-->
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <!--Import Google Icon Font-->
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
    <link href="https://fonts.googleapis.com/css?family=Raleway:500&display=swap" rel="stylesheet">

    <!--Import Font Awesome Icon Font-->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css" integrity="sha25

    <!--Import materialize.css-->
    <link rel="stylesheet" type="text/css" href="static/css/materialize.min.css">

    <!--Main css-->
    <link rel="stylesheet" type="text/css" href="static/css/style.css">
    <meta name="viewport" content="width=device-width, initial-scale=1">

</head>

<body>
    <div class="container">

        <!-- Modal for rendering the charts, declare this if you want to render charts,
         else you remove the modal -->
        <div id="modal1" class="modal">
            <canvas id="modal-chart"></canvas>
        </div>

        <!--chatbot widget -->
        <div class="widget">
            <div class="chat_header">

                <!--Add the name of the bot here -->
                <span class="chat_header_title">CSUDHBot</span>
                <span class="dropdown-trigger" href='#' data-target='dropdown1'>
                  <i class="material-icons">
                  more_vert
                  </i>
                </span>
```

Figure 64: HTML open source code provided for chat-widget interface



```
function setBotResponse(response) {

    //display bot response after 500 milliseconds
    setTimeout(function() {
        hideBotTyping();
        if (response.length < 1) {
            //if there is no response from Rasa, send  fallback message to the user
            var fallbackMsg = "I am facing some issues, please try again later!!!";

            var BotResponse = '<img class="botAvatar" src="./static/img/DominguezToros.png"/><p class="botMsg">' + fallbackMsg + '</p

            $(BotResponse).appendTo(".chats").hide().fadeIn(1000);
            scrollToBottomOfResults();
        } else {

            //if we get response from Rasa
            for (i = 0; i < response.length; i++) {

                //check if the response contains "text"
                if (response[i].hasOwnProperty("text")) {
                    var BotResponse = '<img class="botAvatar" src="./static/img/DominguezToros.png"/><p class="botMsg">' + response[
                    $(BotResponse).appendTo(".chats").hide().fadeIn(1000);
                }

                //check if the response contains "images"
                if (response[i].hasOwnProperty("image")) {
                    var BotResponse = '<div class="singleCard">' + '<img class="imgcard" src="' + response[i].image + '">' + '</div>
                    $(BotResponse).appendTo(".chats").hide().fadeIn(1000);
                }


                //check if the response contains "buttons"
                if (response[i].hasOwnProperty("buttons")) {
                    addSuggestion(response[i].buttons);
                }

                //check if the response contains "attachment"
                if (response[i].hasOwnProperty("attachment")) {

                    //check if the attachment type is "video"
                    if (response[i].attachment.type == "video") {
                        video_url = response[i].attachment.payload.src;
```

Figure 65: JavaScript code from open source with permission to edit

CHAPTER 11

CSUDHBOT MODELS

Training Models

Training the CSUDHBot's models mostly took place in the command-line interface. The command trains the model that combines both Rasa NLU and Rasa Core as one. Since the project used both Rasa NLU and Rasa Core, training the model was simple. However, the time it took from making changes to training the model was a slow process. Moreover, after the model is trained, the models are saved in the model's folder. This gives the ability to retrieve and test them in the Rasa X GUI.

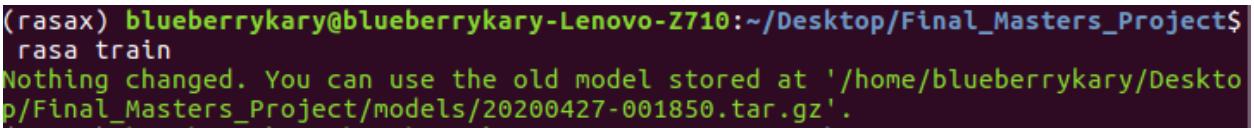


Figure 66: Training model command

Loading and Running the Model

During the implementation, loading and running the model mostly took place in the command line. The commands used were 'rasa shell' and 'rasa run actions' in two separate terminal windows. If the action server was not running, the project would not function properly since the project contains many actions.

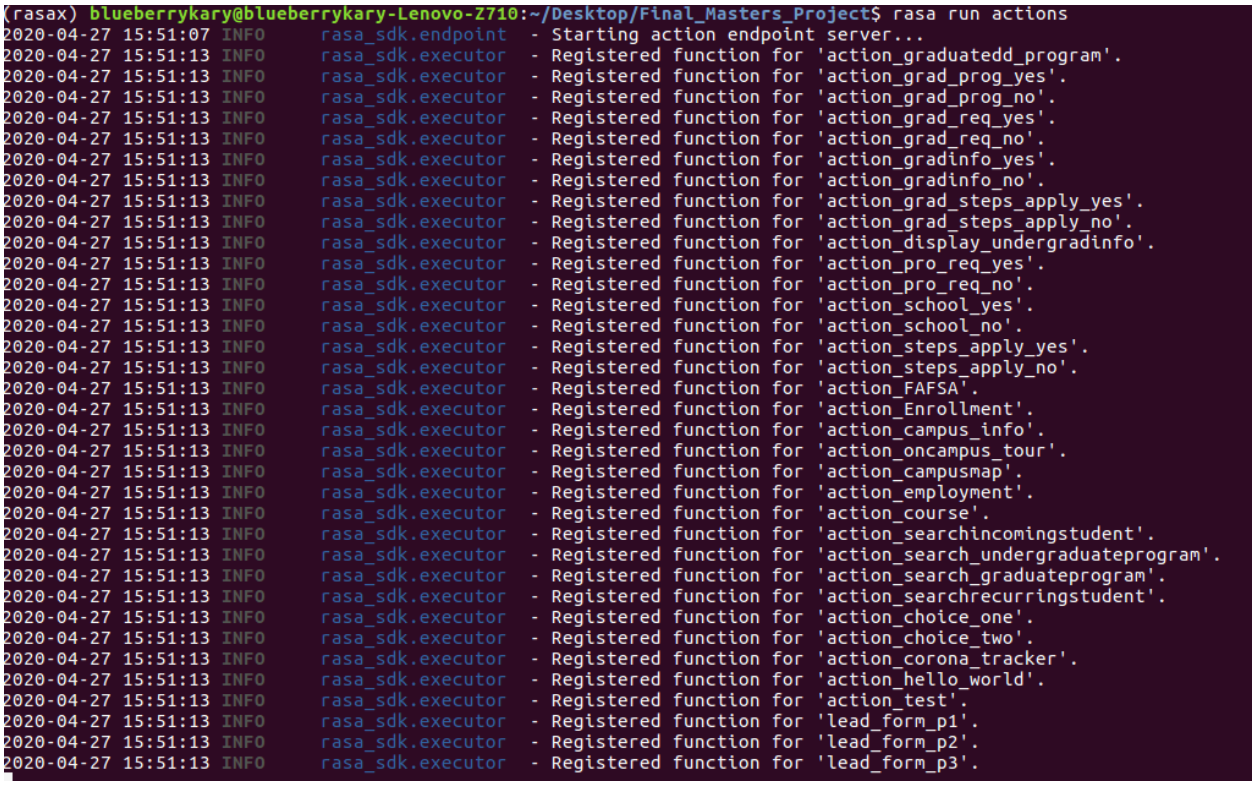


Figure 67: Actions server running

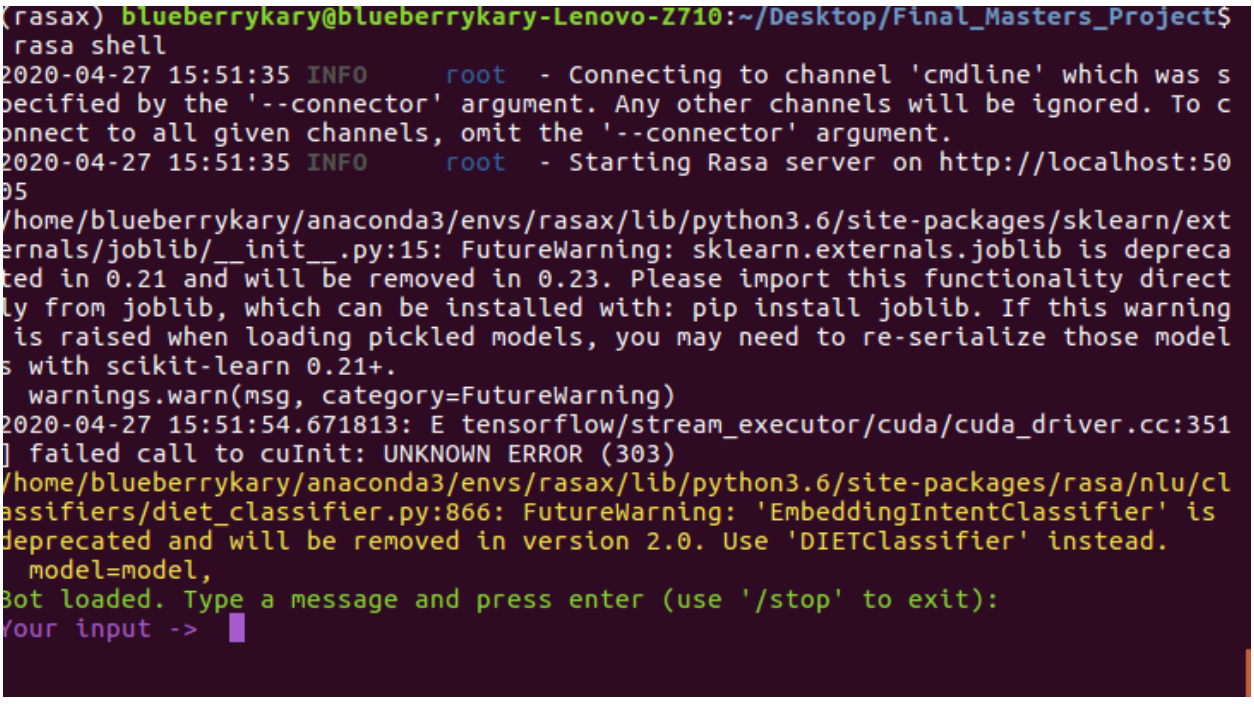


Figure 68: Rasa shell running

After connecting the chat-widget with the CSUDHBot, the ‘rasa run -m models --enable-api --cors "*" --debug’ command would have to start, along with the action server. This would launch the main web page where the chat-widget is connected.

```
(rasax) blueberrykary@blueberrykary-Lenovo-Z710:~/Desktop/Final_Masters_Project$
 rasa run -m models --enable-api --cors "*" --debug
2020-04-27 15:57:52 DEBUG    rasa.cli.utils  - Parameter 'endpoints' not set. Us
ing default location 'endpoints.yml' instead.
2020-04-27 15:57:52 DEBUG    rasa.cli.utils  - Parameter 'credentials' not set.
Using default location 'credentials.yml' instead.
2020-04-27 15:57:57 DEBUG    rasa.core.utils  - Available web server routes:
/conversations/<conversation_id>/messages          POST
  add_message
/conversations/<conversation_id>/tracker/events    POST
  append_events
/webhooks/rasa                                     GET
  custom_webhook_RasaChatInput.health
/webhooks/rasa/webhook                             POST
  custom_webhook_RasaChatInput.receive
/webhooks/rest                                     GET
  custom_webhook_RestInput.health
/webhooks/rest/webhook                             POST
  custom_webhook_RestInput.receive
/model/test/intents                                POST
  evaluate_intents
/model/test/stories                                POST
  evaluate_stories
/conversations/<conversation_id>/execute           POST
  execute_action
/domain                                            GET
  get_domain
/socket.io                                         GET
  handle_request
/                                                  GET
  hello
/model                                             PUT
  load_model
/model/parse                                       POST
  parse
/conversations/<conversation_id>/predict           POST
  predict
/conversations/<conversation_id>/tracker/events    PUT
  replace_events
/conversations/<conversation_id>/story             GET
  retrieve_story
/conversations/<conversation_id>/tracker           GET
  retrieve_tracker
/webhooks/socketio                                 GET
  socketio_webhook.health
/status                                            GET
```

Figure 69: Web-socket connection

Figure 70: Launching the chat-widget after web-socket connection

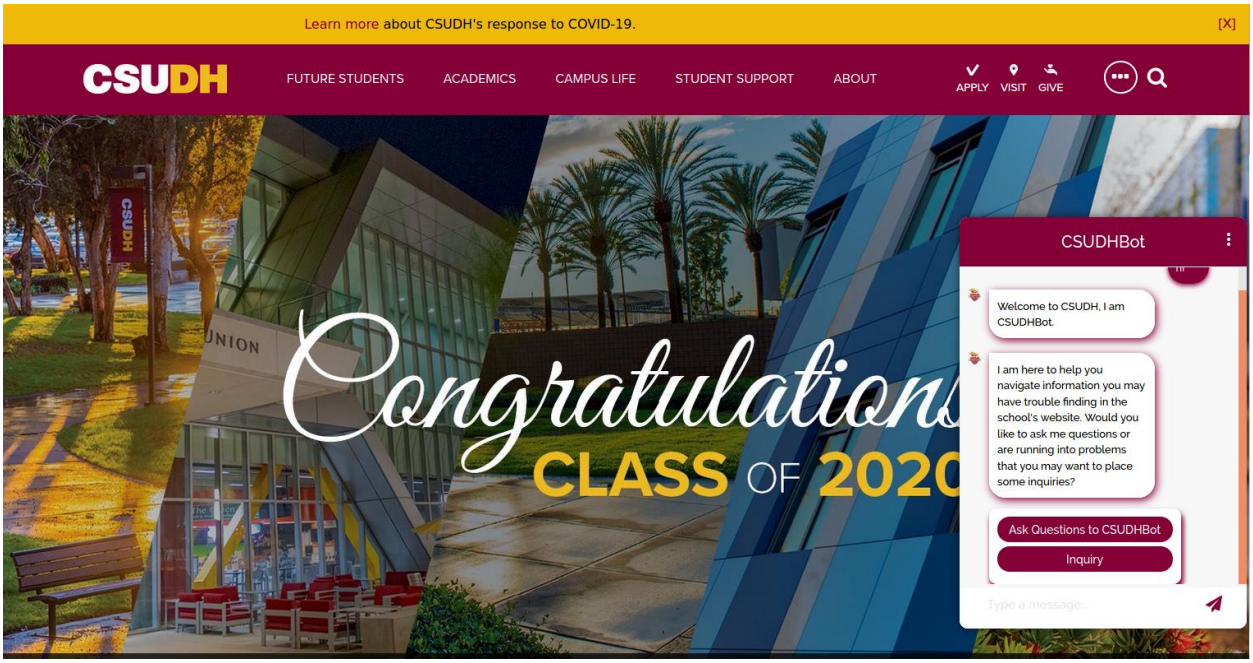


Figure 71: CSUDH Website with CSUDHBot

Testing Models

While testing the model, it ran into some issues when the command ‘run test’ was not working properly. Instead, the model was tested manually by the command line by seeing the flow of the conversation between the user and the bot. Most of the mistakes were noticed in the actions server that would indicate if an action was not registered. Moreover, though the command ‘run test’ does not work properly, the version of the Rasa X installed does not run tests with that command. Instead, the command used to run the test is the ‘run data validate.

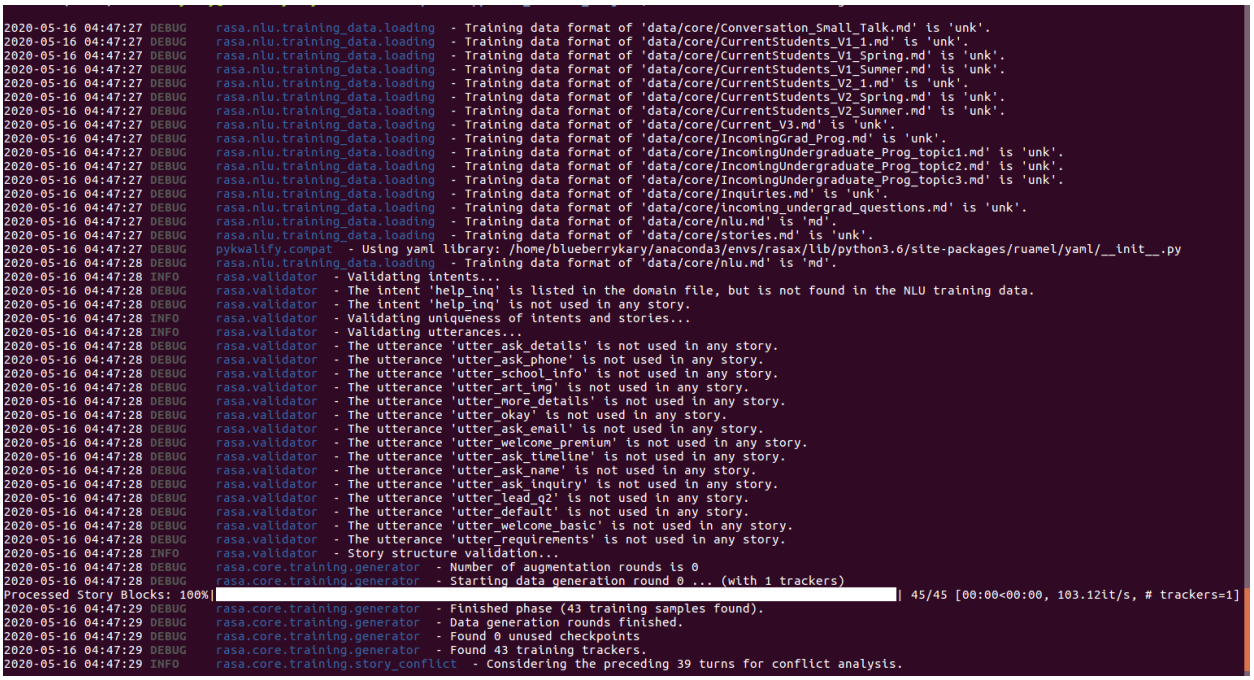


Figure 72: The command 'rasa data validate' output to test model

After the presentation, one of the concerns brought by a professor was the utilization of test cases. Test cases in the chat-bot are meant to allow the person administering the bot to set the test cases to get results to figure out where in the design is working and where it isn't in order to go back to the original design and make changes to make it more effective. One of the issues ran for building test cases was not having a group of individuals to test the bot. However, other alternatives to do test cases consisted of testing each use case in the bot. The following contains the tables of the test cases in **Figure 73** through **Figure 77**, and then the results are shown in **Figure 78**.

Place an Inquiry

ID	Test Case 1
Item or Feature:	Input Inquiry as queries
Objective	The user places an inquiry onto the CSUDHBot
Set-Up	The user places their inquiry by answering all of the queries asked by the bot and providing their information.
Expected Output	The CSUDHBot prints out the user's input
Test Procedure	1. Begin CSUDHBot Session 2. Select option to make an inquiry 3. Fill in inquiry 4. Have CSUDHBot output inquiry information 5. End Session

Figure 73: Test Case 1

Chit-Chatting

ID	Test Case 2
Item or Feature:	Chit-Chat with bot in middle of conversation
Objective	The user chats with the bot by entering queries
Set-Up	The user chats with the bot by entering random queries, such as 'I am tired'
Expected Output	The CSUDHBot responds to the user's query
Test Procedure	1. Begin CSUDHBot Session 2. Enter an inquiry and get response 5. End Session

Figure 74: Test Case 2

Querying data with word intents

ID	Test Case 3
Item or Feature:	Query data with word intents
Objective	The user queries data with words intents that are already pre-trained in the bot.
Set-Up	The user queries the data by entering a intent such as 'campus'
Expected Output	The CSUDHBot returns result after user queries data.
Test Procedure	1. Begin CSUDHBot Session 2. Select option to ask a question to bot 3. Follow through the conversation 4. Enter word intents to query data and have a returned result 5. End Session

Figure 75: Test Case 3

-

View Stored Data

ID	Test Case 4
Item or Feature:	Viewing Stored Data in Database
Objective	The administrator views the data stored in user/bot conversation in the database
Set-Up	The administrators opens up the database to view stored data. Tracker must already be set-up
Expected Output	The administrator views the data table called "events" to view the stored data
Test Procedure	1. Open Datatbase 2. Open table "events" 3. View "events" stored data

Figure 76: Test Case 4

Bot Stores Data

ID	Test Case 5
Item or Feature:	Bot stores data in the database
Objective	The CSUDHBot tracker's store stores data in the database
Set-Up	By including the tracker into the endpoints
Expected Output	Bot has to store data inside database and is confirmed when it is run, no error shown of missing tracker
Test Procedure	1. Run program with actions and shell 2. View if no tracker error. 3. View data in database after conversation

Figure 77: Test Case 5

Test Results

S No.	Use Case	Description	Test Case	Expected Results	Actual Result	Pass/Fail
1	Inquiries	Input an inquiry for an issue	Place an Inquiry	The CSUDHBot prints out the user's input	Input accepter and printed	Pass
2	Chatting	Chit-Chats with Bot	Chit-Chatting	The CSUDHBot responds to the user's query	Bot response after random chit-chat query	Pass
3	Searching	Input Search Inquiries	Querying data with word intents	The CSUDHBot returns result after user queries data.	Result printed after retrieved from database.	Fail, Re-Test, Fail, Re-Test, Pass
4	View Stored Data	Administrator view the stored data in the database	View Stored Data	The stored data is inside the database after each interaction between bot and user	Administrator is able to view all the stored data	Pass
5	Bot Stores Data	The system's tracker store should store data	Stores Data	Bot has to store data inside database and is confirmed when it is run, no error shown of missing tracker	Bot stores data inside the database	Pass

Figure 78: Test Case Results

CHAPTER 12

CSUDHBOT EXECUTION RESULTS

CSUDHBot Introduction

CSUDHBot utters (responds) to the student's message after the student greets the bot with a simple keyword such as "Hello". The opening window provides the user with two options. One allowing the student to ask a question to CSUDHBot, or second to make an inquiry.

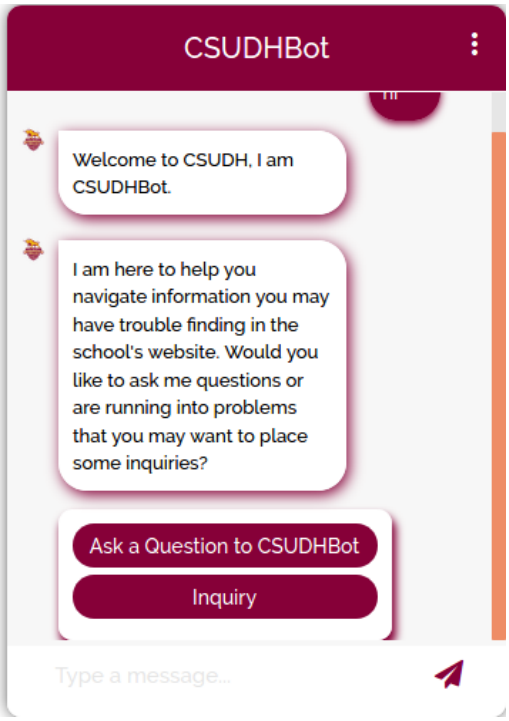


Figure 79: CSUDHBot Introduction

Guiding Student in Conversation

The CSUDHBot asks a student if they are a ‘recurring’ or ‘incoming student’. Once the user identifies who they are, the bot redirects them to a story that corresponds to the query they entered. In this case, the user wants to follow the ‘incoming undergraduate student’ story. **Figure 80** through **Figure 87** displays how the bot guides the user into the conversation.

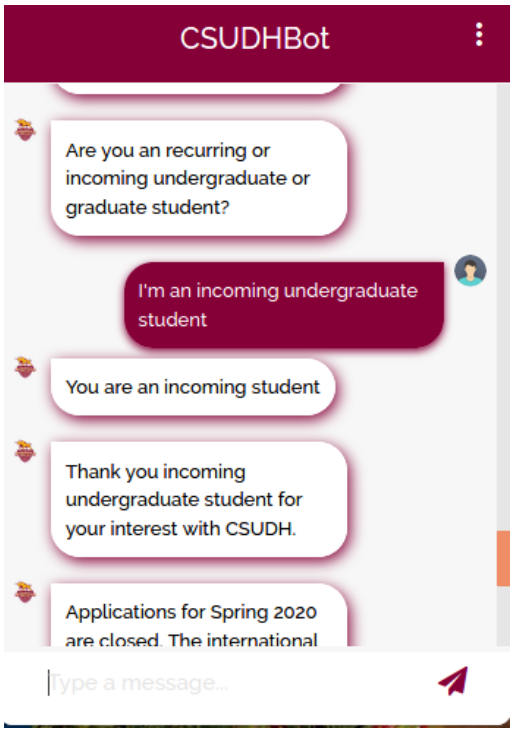


Figure 80: Incoming Undergraduate Student – 1

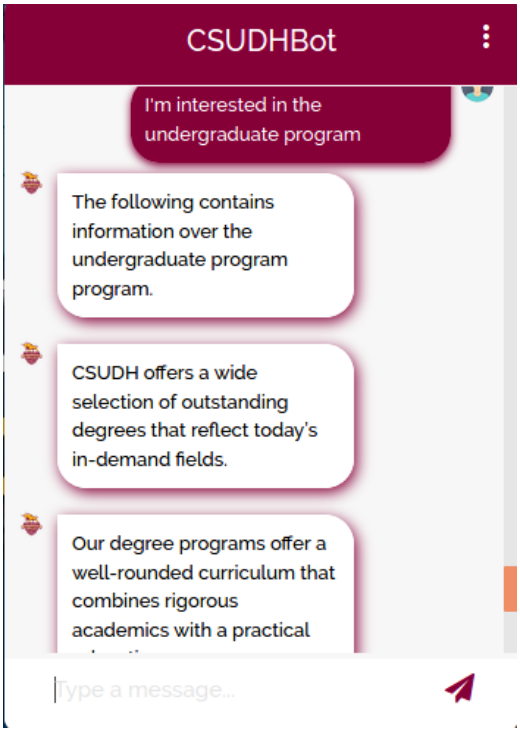


Figure 81: Incoming Undergraduate Student – 2

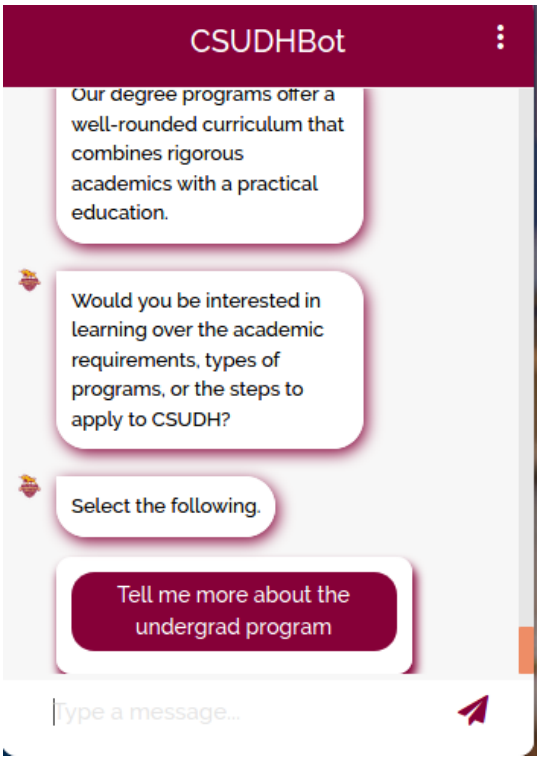


Figure 82: Incoming Undergraduate Student – 3

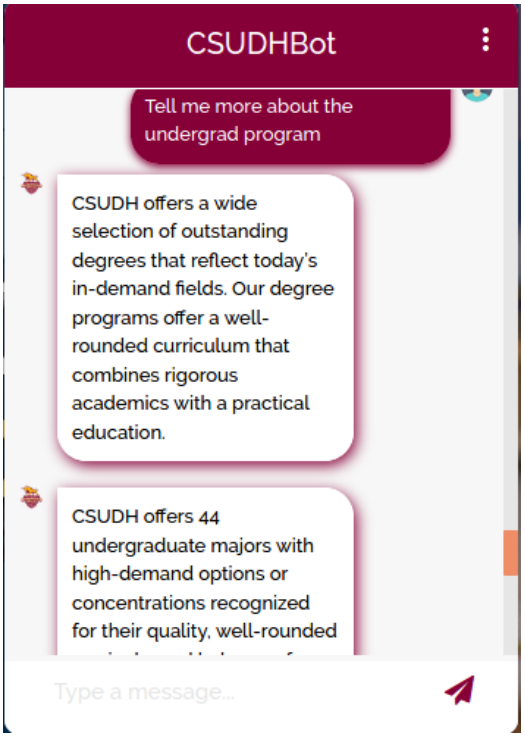


Figure 83: Incoming Undergraduate Student – 4

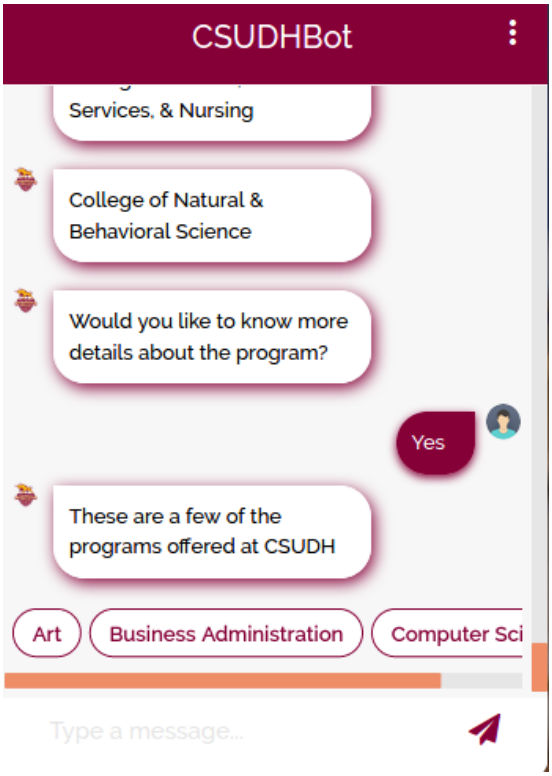


Figure 84: Incoming Undergraduate Student – 5

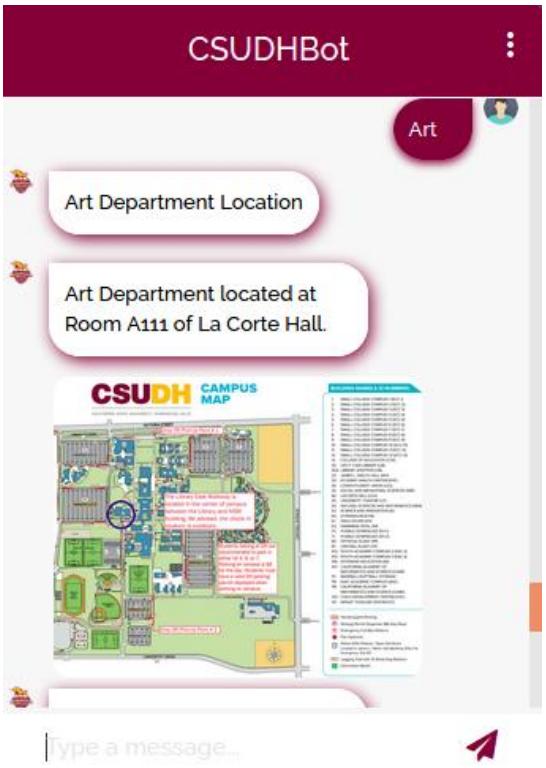


Figure 85: Incoming Undergraduate Student – 6

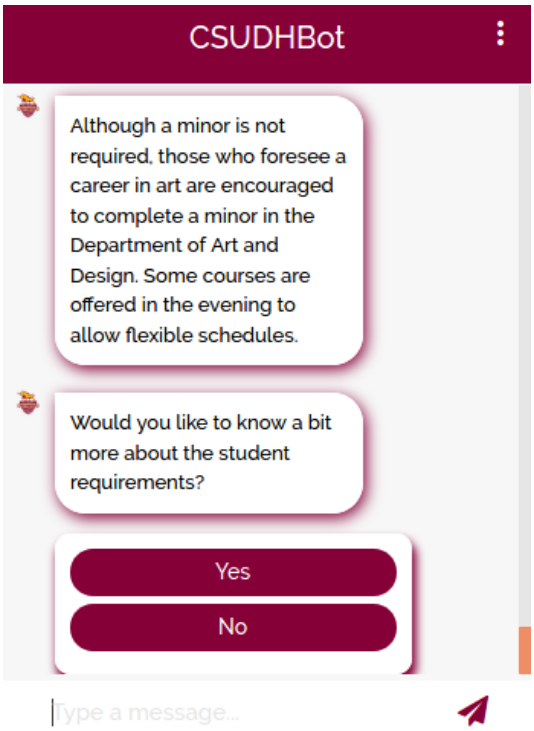


Figure 86: Incoming Undergraduate Student – 7

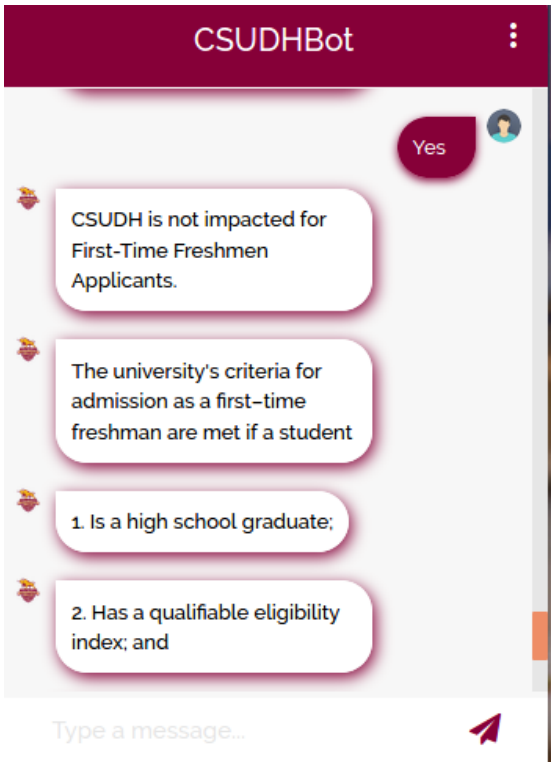


Figure 87: Incoming Undergraduate Student – 8

Making an Inquiry

When the user selects the ‘inquiry’ option, they are redirected to make inquiries over their problem. As the user enters their inquiries, the slots in the bot (which are the memory) records the current conversation and once the user is finished, the data entered by the user is printed out. **Figure 88** through **Figure 92** show the flow of the conversation.

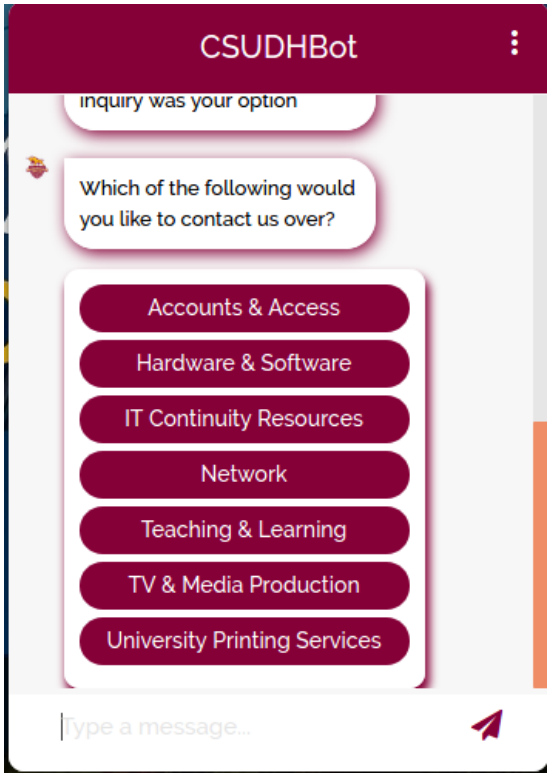


Figure 88: Inquiry-1

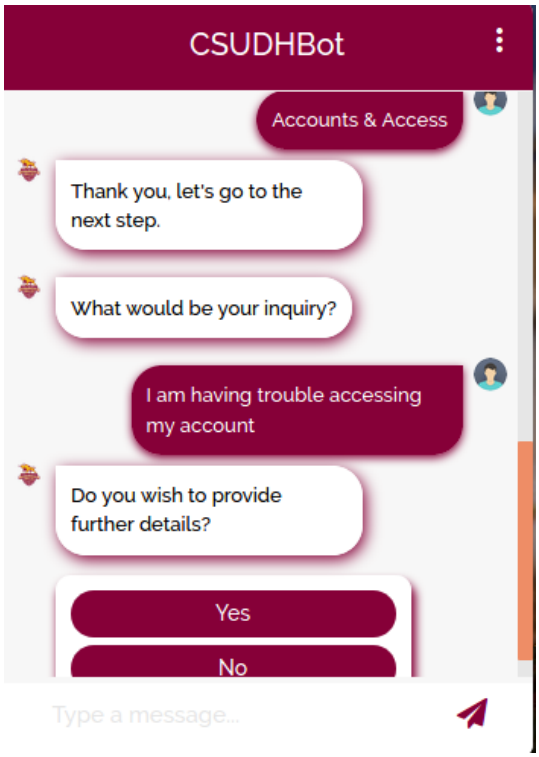


Figure 89: Inquiry-2

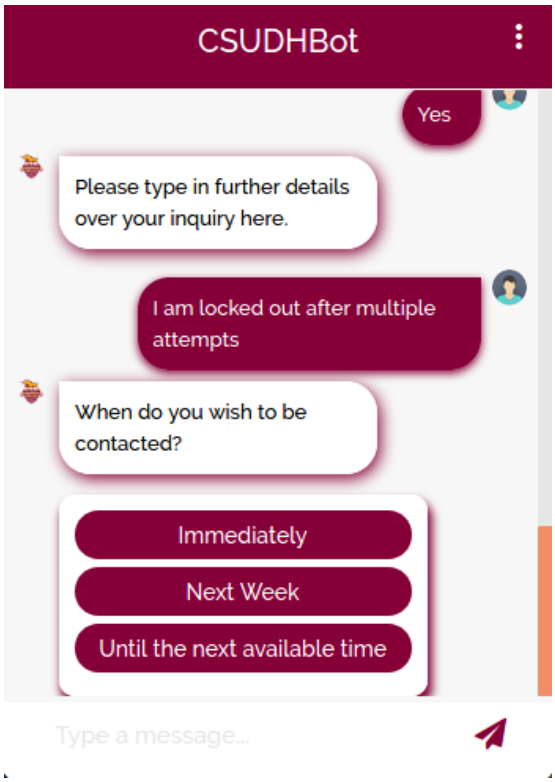


Figure 90: Inquiry-3

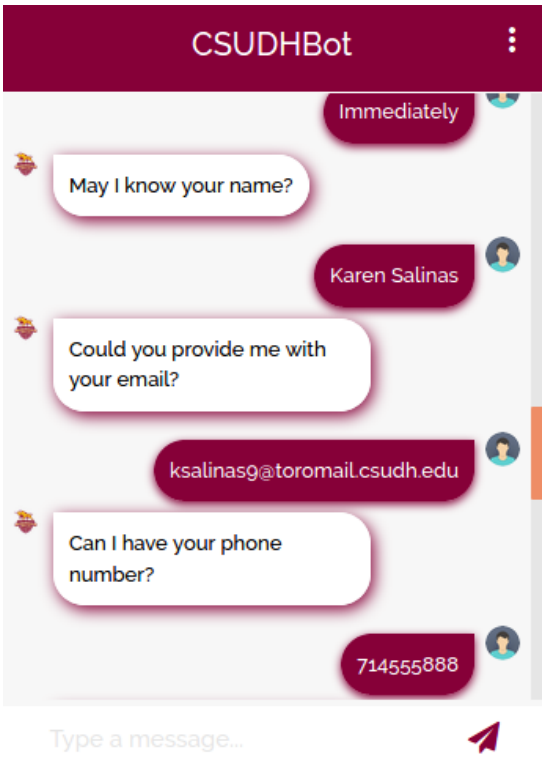


Figure 91:Inquiry-4

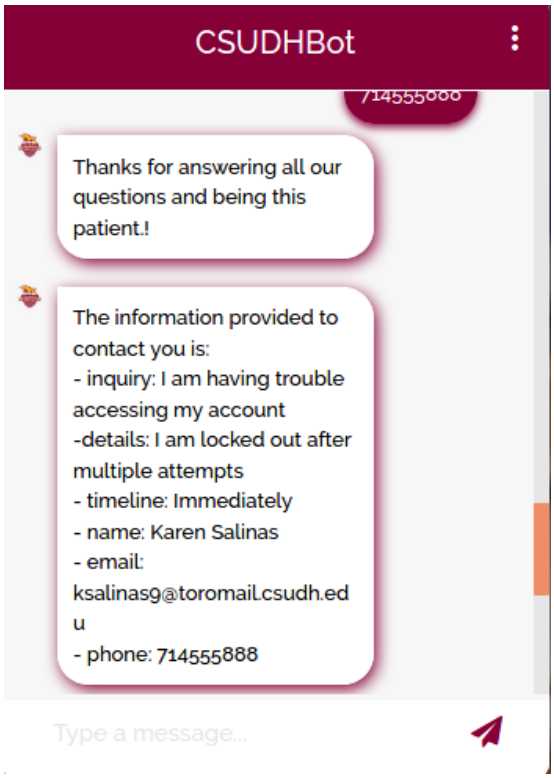


Figure 92: Inquiry-5

Chit-Chat

In the middle of a conversation, the user is able to tell the bot that they are ‘bored’ or the are ‘tired’. The bot responds by small talk response. Figure 93 shows an example of chit-chat.

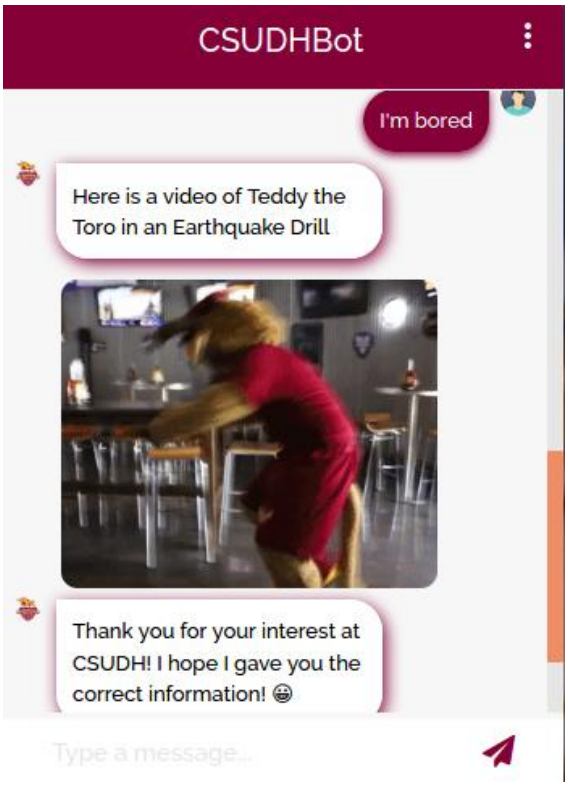


Figure 93: Chit-Chat

Searching Querying Intents-Story 1

The CSUDH guides the user with the conversation at the beginning, but then allows the user to query throughout the conversation. These queries can be intents already in the pre-trained data included in the NLU. This includes keywords such as ‘financial aid’, ‘enrollment’, or ‘computer science’. **Figure 94** and **Figure 99** displays how the user enters their queries and the bot returns a result.

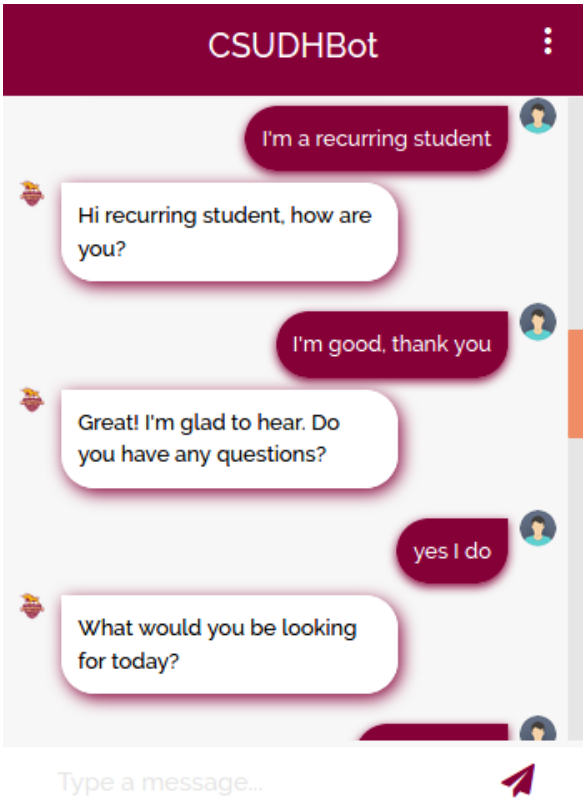


Figure 94: Story1-1

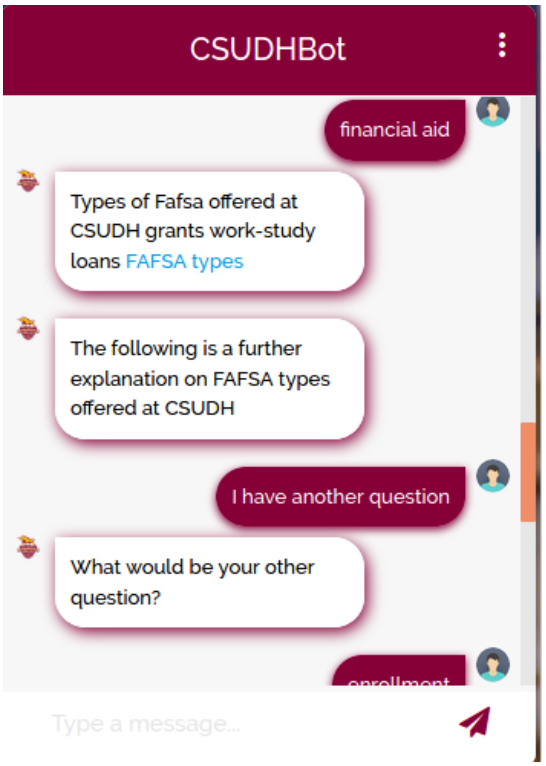


Figure 95: Story1-2

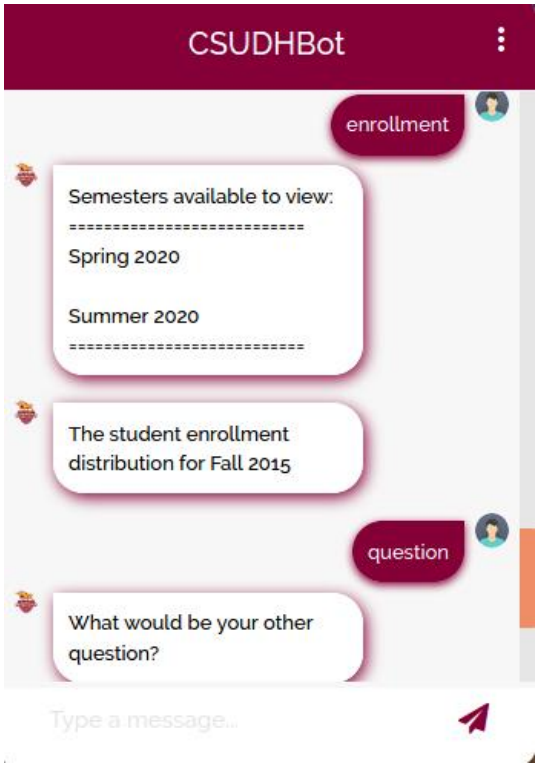


Figure 96: Story1-3

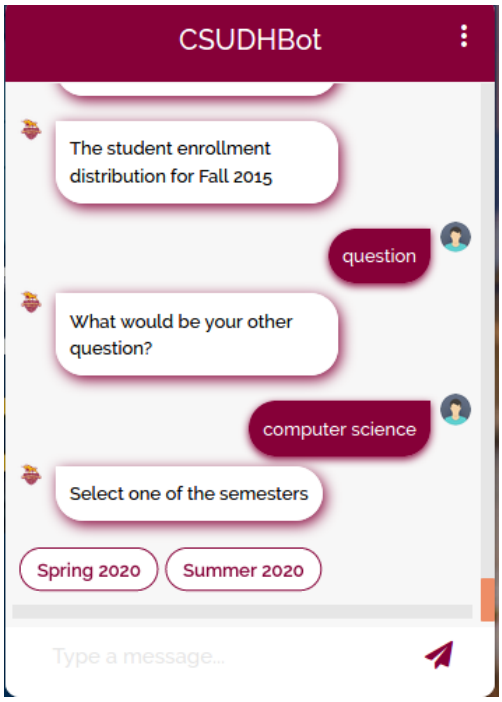


Figure 97: Story1-4

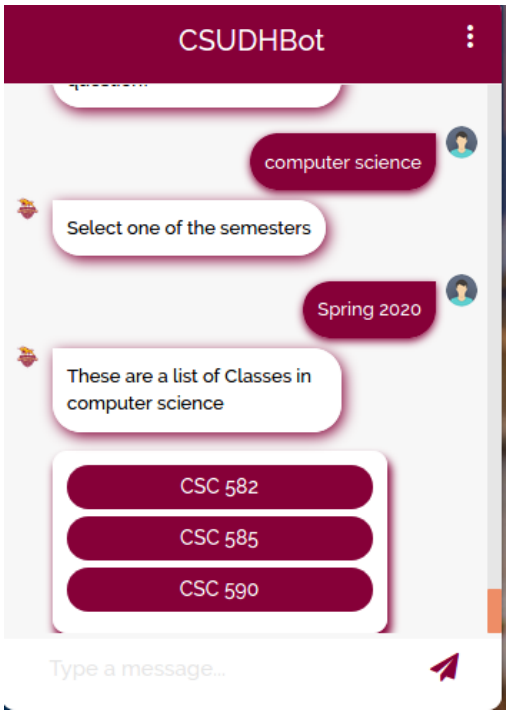


Figure 98: Story1-5

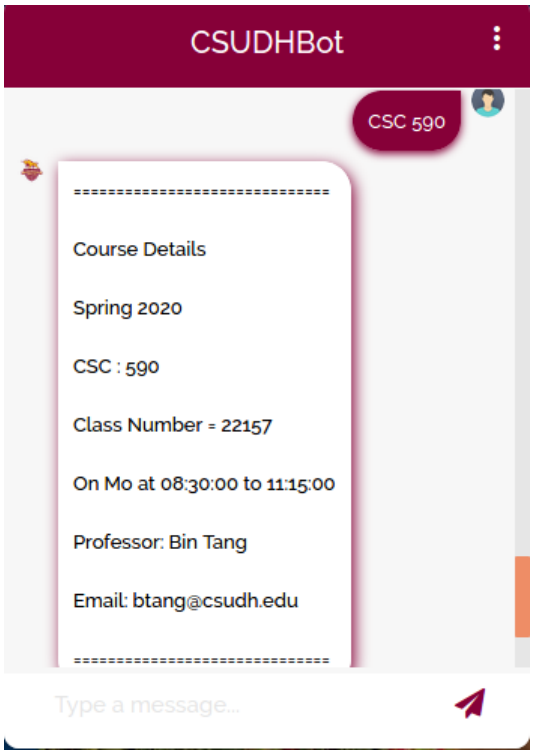


Figure 99: Story1-6

Searching Querying Intents-Story 2

Extending from story one, the CSUDH guides the user with the conversation at the following conversation shows another query intent from the user, which is “campus”, where the user is displayed with a list of options to choose from. Once the user choses the payload, the bot returns a result. **Figure 100** and **Figure 101** show that result.

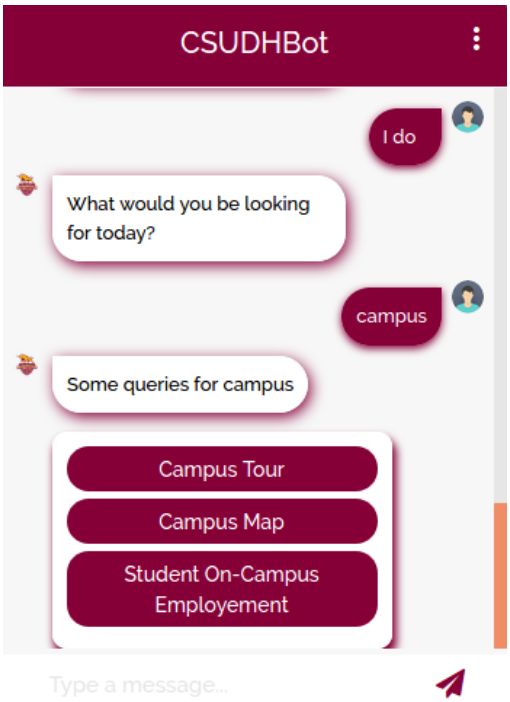


Figure 100: Story2-1

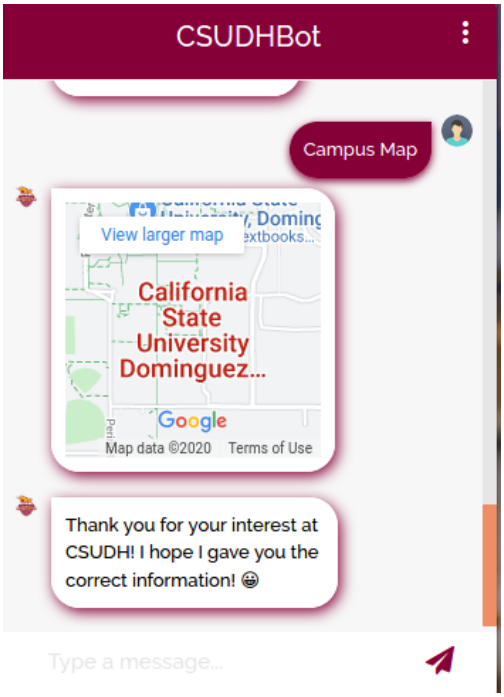


Figure 101: Story2-2

CHAPTER 13

CONCLUSION AND FUTURE WORK

The future goal for the CSUDHBot is to expand the program by adding a wide range of use cases and various features. An appealing feature would be helping students to register to a class whereas another feature that would market the bot well would be it display the daily food menu of the school's Locker Union Food Court. Moreover, the primary goal of CSUDHBot was to assists students in navigating the school's website. While working with the virtual assistant, the possibility of implementing various services is not far-fetched. Since the school is facing unprecedented times, a virtual assistant would be an appealing idea to add in the school's website since human support services won't be available for a year. In conclusion, the NLP machine learning method set to create a school bot has demonstrated the many ways uses of creating stories to provide different services that accommodates any user.

REFERENCES

[1] Dylanavalverde, "Perception, Control, Cognition," 05 December 2018. [Online]. Available: https://pcc.cs.byu.edu/2018/03/26/a-brief-history-of-chatbots/.

[2] G. S. a. G. R. K.Jwala, "Developing a Chatbot using Machine Learning," Vols. Volume-8 Issue-1S3, IJRTE, no. ISSN: 2277-3878, 2019.

[3] "Tracker Stores,". https://rasa.com/docs/rasa/api/tracker-stores/.

[4] Architecture. https://rasa.com/docs/rasa-x/installation-and-setup/architecture/.

[5] "Domains,". rasa.com/docs/rasa/core/domains/.

[6] T. Wochinger, "The Rasa Blog: Machine Learning Powered by Open Source," The Rasa Blog: Machine Learning Powered by Open Source, 04 June 2019. [Online]. Available: https://blog.rasa.com/rasa-nlu-in-depth-part-1-intent-classification/.

[7] "Entity Extraction,". https://rasa.com/docs/rasa/nlu/entity-extraction/.

[8] "Slots," [Online]. Available: https://rasa.com/docs/rasa/core/slots/.

[9] "Stories," [Online]. Available: rasa.com/docs/rasa/core/stories/.

APPENDIX A: SOURCE CODE

File: Domain.yml

##Note: All source code in CSUDBot Project is found in

##https://github.com/blueberrykary/Final_Master_Project/blob/master/Final_Masters_Project/

Git clone source code to run it, but implementation of database will be required as it is

##provided in Appendix B

```
session_config:
  session_expiration_time: 60
  carry_over_slots_to_new_session: true
##============================================================================##
##============================================================================##
##==================================INTENTS=================================##
intents:
- greet
- goodbye
- choice_payloads
- choice_text
- student_type_two
- student_type_one
- choose_incoming_undergrad_program
- choose_incoming_grad_program
- undergrad_programs_info
- accept
- reject
- begin_lead
- help_inq
- select_program_details_no
- select_program_details_yes
- select_requirements_yes
- select_requirements_no
- select_school_info_yes
- select_school_info_no
- select_stepstoapply_yes
- select_stepstoapply_no
- select_grad_program_details_yes
- select_grad_program_details_no
- select_grad_requirements_yes
- select_grad_requirements_no
- select_gradschool_yes
- select_gradschool_no
- select_gradsteps_yes
- select_gradsteps_no
- good_thanks
- yes_ido
- grad_program_info
- financial_aid_application
- university_enrollment
```

```
- on_campus
- on_campus_tour
- campus_map
- campus_employment
- art
- business_admin
- computer_science
- course_description
- Spring2020
- Summer2020
- courses_spring2020
- courses_spring20201
- courses_spring20202
- ask_a_question
- no_more_ques
- courses_summer2020
- courses_summer20201
# - agree
# - disagree
- bot_challenge
- bored
- sad
- lazy
##============================================================================##
##============================================================================##
##============================================================================##
##============================ENTITIES======================================##
entities:
- student_num_one
- student_num_two
- instruct_one
- go
- back
- link
- redirected
- link2
- redirected2
- link3
- redirected3
- link4
- redirected4
- link5
- redirected5
- sched_appointment
- Appt
- uni
- universities
- school
- major
- by_maj
- choice_one
- c_one
- choice
- choice_two
- student_incoming
- student_recurr
- student
```

- iinfo
- rinfo
- uprogram
- uprograms
- gprogram
- gprograms
- program_undergrad
- program_undergrad_info
- program_school_info
- program_under_deadline
- uprog_info
- uprog_iinfo
- uprog_req
- uprog_requirements
- uschool_info
- uschool_iinfo
- uprog_apply
- uprogram_apply
- uprog_wapply
- uprogram_wapply
- uprog_dline
- uprogram_deadline
- inquiry
- details
- timeline
- name
- email
- phone
- program_graduate
- program_details_yes
- program_details_no
- program_yes
- program_no
- prog_yes
- prog_no
- program_requirement_details_yes
- program_requirement_details_no
- program_req_yes
- program_req_no
- prog_req_yes
- prog_req_no
- school_info_yes
- school_info_no
- schoolinfo_no
- schooli_no
- schoolinfo_yes
- schooli_yes
- steps_apply_yes
- steps_apply_no
- steps_app_yes
- steps_app_no
- steps_yes
- steps_no
- graduate_program_details
- graduate_program
- program_masters
- grad_program_info

- grad_program
- program_grad_details_yes
- graduate_prog_yes
- grad_prog_yes
- program_grad_details_no
- grad_prog_no
- graduate_prog_no
- program_gradrequirement_details_yes
- program_gradrequirement_details_no
- graduate_progreq_yes
- grad_progreq_yes
- graduate_progreq_no
- grad_progreq_no
- view_gradschool_yes
- view_gradschool_no
- graduateinfo_yes
- graduateinfo_no
- gradinfo_yes
- gradinfo_no
- view_grad_steps_yes
- view_grad_steps_no
- graduatesteps_app_yes
- gradsteps_yes
- graduatesteps_app_no
- gradsteps_no
- FAFSA_APP
- fafsa_appp
- fafsa_application
- uni_enroll
- university_enroll
- unii_enrollment
- campus_CSUDH
- campus_information
- campus_info
- campus_tour_CSUDH
- on_campus_tour_info
- campus_tour_info
- campus_map_CSUDH
- campusMapCSUDH
- campus_mapCSUDH
- campus_employment_CSUDH
- campus_employment_stu
- campus_employment
- course
- getting_course
- course_start
- course_end
- spring_2020_course1
- spring_c1
- spring_2020c1
- spring_2020_course2
- spring_c2
- spring_2020c2
- spring_2020_course3
- spring_c3
- spring_2020c3
- summer_2020_course1

```
- summer_2020_course2
- summer_c1
- summer_c2
- summer_2020c1
- summer_2020c2

##==========================================================================##
##==========================================================================##
##==========================================================================##
##==================================SLOTS===================================##
slots:
  Appt:
    type: text
  FAFSA_APP:
    type: text
  by_maj:
    type: text
  c_one:
    type: text
  campusMapCSUDH:
    type: text
  campus_CSUDH:
    type: text
  campus_employment:
    type: text
  campus_employment_CSUDH:
    type: text
  campus_employment_stu:
    type: text
  campus_info:
    type: text
  campus_information:
    type: text
  campus_mapCSUDH:
    type: text
  campus_map_CSUDH:
    type: text
  campus_tour_CSUDH:
    type: text
  campus_tour_info:
    type: text
  choice:
    type: text
  choice_one:
    type: text
  choice_two:
    type: text
  details:
    type: unfeaturized
    auto_fill: false
  email:
    type: unfeaturized
    auto_fill: false
  fafsa_application:
    type: text
  fafsa_appp:
    type: text
```

```
go:
  type: text
gprogram:
  type: text
gprograms:
  type: text
grad_prog_no:
  type: text
grad_prog_yes:
  type: text
grad_program:
  type: text
grad_program_info:
  type: text
grad_progreq_no:
  type: text
grad_progreq_yes:
  type: text
gradinfo_no:
  type: text
gradinfo_yes:
  type: text
gradsteps_no:
  type: text
gradsteps_yes:
  type: text
graduate_prog_no:
  type: text
graduate_prog_yes:
  type: text
graduate_program_details:
  type: text
graduate_progreq_no:
  type: text
graduate_progreq_yes:
  type: text
graduateinfo_no:
  type: text
graduateinfo_yes:
  type: text
graduatesteps_app_no:
  type: text
graduatesteps_app_yes:
  type: text
iinfo:
  type: text
inquiry:
  type: unfeaturized
  auto_fill: false
instruct:
  type: text
instruct_one:
  type: text
link:
  type: text
link2:
  type: text
```

```
link3:
  type: text
link4:
  type: text
link5:
  type: text
major:
  type: text
name:
  type: unfeaturized
  auto_fill: false
on_campus_tour_info:
  type: text
phone:
  type: unfeaturized
  auto_fill: false
prog_no:
  type: text
prog_req_no:
  type: text
prog_req_yes:
  type: text
prog_yes:
  type: text
program_details_no:
  type: text
program_details_yes:
  type: text
program_grad_details_no:
  type: text
program_grad_details_yes:
  type: text
program_gradrequirement_details_no:
  type: text
program_gradrequirement_details_yes:
  type: text
program_graduate:
  type: text
program_masters:
  type: text
program_no:
  type: text
program_req_no:
  type: text
program_req_yes:
  type: text
program_requirement_details_no:
  type: text
program_requirement_details_yes:
  type: text
program_school_info:
  type: text
program_under_deadline:
  type: text
program_undergrad:
  type: text
program_undergrad_apply:
```

```
  type: text
program_undergrad_info:
  type: text
program_undergrad_req:
  type: text
program_yes:
  type: text
redirected:
  type: text
redirected2:
  type: text
redirected3:
  type: text
redirected4:
  type: text
redirected5:
  type: text
rinfo:
  type: text
sched_appointment:
  type: text
school_info_no:
  type: text
school_info_yes:
  type: text
schooli_no:
  type: text
schooli_yes:
  type: text
schoolinfo_no:
  type: text
schoolinfo_yes:
  type: text
steps_app_no:
  type: text
steps_app_yes:
  type: text
steps_apply_no:
  type: text
steps_apply_yes:
  type: text
steps_no:
  type: text
steps_yes:
  type: text
student_incoming:
  type: text
student_num_one:
  type: text
student_num_two:
  type: text
student_recurr:
  type: text
timeline:
  type: unfeaturized
  auto_fill: false
uni:
```

```
  type: text
uni_enroll:
  type: text
unii_enrollment:
  type: text
universities:
  type: text
university_enroll:
  type: text
uprog_apply:
  type: text
uprog_dline:
  type: text
uprog_iinfo:
  type: text
uprog_info:
  type: text
uprog_req:
  type: text
uprog_requirements:
  type: text
uprog_wapply:
  type: text
uprogram:
  type: text
uprogram_apply:
  type: text
uprogram_deadline:
  type: text
uprogram_wapply:
  type: text
uprograms:
  type: text
uschool_iinfo:
  type: text
uschool_info:
  type: text
user_type:
  type: categorical
  values:
  - future
  - current
view_grad_steps_no:
  type: text
view_grad_steps_yes:
  type: text
view_gradschool_no:
  type: text
view_gradschool_yes:
  type: text
getting_course:
  type: text
course_start:
  type: text
course_end:
  type: text
spring_2020_course1:
```

```
    type: text
  spring_c1:
    type: text
  spring_2020c1:
    type: text
  spring_2020_course2:
    type: text
  spring_c2:
    type: text
  spring_2020c2:
    type: text
  spring_2020_course3:
    type: text
  spring_c3:
    type: text
  spring_2020c3:
    type: text
  summer_2020_course1:
    type: text
  summer_2020_course2:
    type: text
  summer_c1:
    type: text
  summer_c2:
    type: text
  summer_2020c1:
    type: text
  summer_2020c2:
    type: text

##========================================================================##
##========================================================================##
##==============================RESPONSES=================================##
responses:
##========================================================================##
##========================================================================##
##========================================================================##
##========================================================================##
##=========================Introduction ==================================##
## Introduction Message
  utter_introduction:
  - text: Welcome to CSUDH, I am CSUDHBot.
## Introduction Message 2 to give options to user
  utter_introduction_r2:
  - text: I am here to help you navigate information you may have trouble
finding
      in the school's website. Would you like to ask me questions or are
running into
      problems that you may want to place some inquiries?
    buttons:
    - title: Ask Questions to CSUDHBot
      payload: /choice_text{"choice":"text"}
    - title: Inquiry
      payload: /choice_payloads{"choice":"payloads"}
##========================================================================##
##========================================================================##
##========================================================================##
```

```
##=========================greet future student============================##
## Meant to greet the future student
  # utter_greet_text:
  # - text: Hi! You have selected {choice}. How are you today?
##========================================================================##
##========================================================================##
##========================================================================##
##====================Ask if they are recurring or incoming ==============##
## Meant to ask student whether they are a recurring or incoming student.
Depending on their response, they are redirected to a story.
  utter_ask_type_of_student:
  - text: Are you an recurring or incoming undergraduate or graduate student?
##========================================================================##
##========================================================================##
##========================================================================##
##=====================Incoming student message===========================##
## Meant to ask student if they are intereted in undergraduate or graduate
program. This redirects them to either the undergraduate or graduate
storie(s).
  utter_future_student_r3:
  - text: Are you interested in learning more over our graduate or
undergraduate programs?
##========================================================================##
##========================================================================##
##========================================================================##
##======================== Current student message =======================##
 ## Introduces the current student to the current student story.
  utter_current_student:
  - text: Hi {student_recurr}, how are you?
##========================================================================##
##========================================================================##
##========================================================================##
##=============incoming undergraduate starting point of story ============##
  utter_undergraduate:
  - text: "CSUDH offers a wide selection of outstanding degrees that reflect
today’s\
      \ in-demand fields. \n\n Our degree programs offer a well-rounded
curriculum\
      \ that combines rigorous academics with a practical education. \n\n
Would you\
      \ be interested in learning over the academic requirements, types of
programs,\
      \ or the steps to apply to CSUDH?"
##========================================================================##
##========================================================================##
##========================================================================##
## This redirects incoming undergraduate student with an option, as an
example of what could be expanded in the future if the administrator wishes
for more buttons of options
  utter_undergraduate_program_options:
  - text: Select the following.
    buttons:
    - title: Tell me more about the undergrad program
      payload: /undergrad_programs_info{"program_undergrad_info":"undergrad
program"}
##========================================================================##
##========================================================================##
```

```
##=========================================================================##
## Meant for the incoming student opening
  utter_future_student:
  - text: Thank you {student_incoming} for your interest with CSUDH.

  utter_future_student_r2:
  - text: Applications for Spring 2020 are closed. The international
undergraduate
      deadline is on May 1st, 2020.
    image: https://media0.giphy.com/media/mBw8vRgeK75MA9wICY/giphy.gif
#=========================================================================##
##=========================================================================##
##=========================================================================##
##=========================================================================##
## Utter options for programs for incoming undergraduate student and graduate
student
  utter_option_program:
  - text: Would you like to know more details about the program?
    buttons:
    - title: Yes
      payload: /select_program_details_yes{"program_details_yes":"Of Course"}
    - title: No
      payload: /select_program_details_no{"program_details_no":"Of Course
Not"}

##=========================================================================##
##=========================================================================##
##=========================================================================##
##=========================================================================##
##=======Utter Program Information used in Incoming Undergraduate
Program======##
  utter_program_information:
  - text: These are a few of the programs offered at CSUDH
    custom:
      payload: quickReplies
      data:
        - title: Art
          payload: /art
        - title: Business Administration
          payload: /business_admin
        - title: Computer Science
          payload: /computer_science

  utter_art_loc:
  - text: "Art Department Location \n\n Art Department located at Room A111
of La Corte Hall."
    image:
"file:///home/blueberrykary/Desktop/Final_Masters_Project/static/img/art_depa
rt.png"

  utter_art_program:
  - text: "Art Design Option Program Details \n\n The Department of Art and
Design programs at California State University, Dominguez Hills are
characterized by a unique and close correlation among the Art History, Studio
Art, and Design Options, and by a strong coordination among the different
studio areas. \n\n These Options introduce students to potential careers in
the art world as well as providing personal fulfillment through creative and
```

scholarly endeavors. \n\n Although a minor is not required, those who foresee a career in art are encouraged to complete a minor in the Department of Art and Design. Some courses are offered in the evening to allow flexible schedules."

  utter_business_admin_description:
  - text: "The undergraduate program in Business Administration, which leads to a Bachelor of Science degree, is designed to accomplish two objectives. \n\n The first of these is to prepare students for lifelong professional careers in commerce, finance and industry, as well as for management careers in the public and not-for-profit sectors. A second objective is to provide students with the knowledge and skills needed to obtain professional, entry level positions in one functional area of the business enterprise, or in some particular field of business. \n\n After that, students must select a concentration. The following courses, or their approved transfer equivalents, are required of all candidates for this degree.
Because most Business Administration course work completed over ten years ago is outdated, students must consult with an advisor in the Student Advisement and Service Center to ascertain whether courses taken ten or more years ago will need to be repeated. The Associate Dean in the College of Business Administration and Public Policy may waive the requirement to repeat outdated course work if there is evidence that student knowledge is current. Such evidence may consist of recent successful completion of more advanced course work in the specific discipline or satisfactory completion of designated national examinations. "
  utter_business_admin:
  - text: "Admission to the Business Administration major requires completion of all lower division Business Administration core courses with minimum grades of “C”. All new majors are classified as “pre-business majors” until the requirements are met. Upper division advanced and concentration courses are available only to “Business Administration” majors. Students should consult with an advisor in the Business Advisement Center for more information."

  utter_comp_sci:
  - text: "The curriculum is designed to provide preparation for professional careers in the areas of software design and applied computer science, as well as to give the necessary theoretical background for graduate study in the field and to allow a flexible response to a dynamic and growing profession. \n\n The required courses give students a firm foundation in the basic areas of computer science and related areas of mathematics, and a wide choice of electives allow them to tailor their program to their specific interests. \n\n The baccalaureate program (Bachelor of Science in Computer Science) is accredited by the Computer Science Accreditation Commission of the Accreditation Board for Engineering and Technology, 111 Market Place, Suite 1050, Baltimore, MD 21202-4012, (410) 347-7700. The programs accredited by the Commission of Accreditation Board for Engineering and Technology are accredited as separate and distinct from any other programs or kinds of accreditation."

  utter_comp_sci_req:
  - text: " Students entering the Computer Science program must complete the following. \n\n Earn an overall grade point average of 2.0 or better in courses taken outside of the department.\n\n Earn a grade of C or better in each course taken within the department. \n\n Earn a grade of C or better in all direct and indirect prerequisite courses listed in the catalog before advancing to the next level course in a sequence for English, Mathematics,

```
and Science courses. \n\n Students must take capstone course CSC 492 at
CSUDH."
##=========================================================================##
##=========================================================================##
##=========================================================================##
##=========================================================================##
## Steps for incoming undergraduate in requirements, school information, and
## steps to apply.

  utter_curious_req:
  - text: Would you like to know a bit more about the student requirements?
    buttons:
    - title: Yes
      payload:
/select_requirements_yes{"program_requirement_details_yes":"Indeed"}
    - title: No
      payload:
/select_requirements_no{"program_requirement_details_yes":"Definately
       Not"}

  utter_ask_school:
  - text: Would you like to know more over the school information?
    buttons:
    - title: Yes
      payload: /select_school_info_yes{"school_info_yes":"Yessss"}
    - title: No
      payload: /select_school_info_no{"school_info_no":"Nooo"}

  utter_ask_steps:
  - text: Would you like to see the steps to applying to CSUDH?
    buttons:
    - title: Yes
      payload: /select_stepstoapply_yes{"steps_apply_yes":"Steps yes"}
    - title: No
      payload: /select_stepstoapply_no{"steps_apply_no":"Steps no"}
##=========================================================================##
##=========================================================================##
##=========================================================================##
##=========================================================================##
  utter_thank_you_uprogram:
  - text: Thank you for your interest at CSUDH! I hope I gave you the correct
information!
      😀

  utter_okay:
  - text: Okay great! Would you like to know what the requirements are?


  utter_graduate_program_options:
  - text: Which of the following would you like to explore?
    buttons:
    - title: Tell me more about the graduate program
      payload: /grad_program_info{"program_masters":"graduate program"}

  utter_grad_option_program:
  - text: Would you like to view few of the programs offered at CSUDH?
    buttons:
```

```
    - title: Yes
      payload:
/select_grad_program_details_yes{"program_grad_details_yes":"Yupp"}
    - title: No
      payload:
/select_grad_program_details_no{"program_grad_details_no":"Noope"}

  utter_grad_requirements:
  - text: Would you like to know a bit more about the graduate requirements?
    buttons:
    - title: Yes
      payload:
/select_grad_requirements_yes{"program_gradrequirement_details_yes":"Yas"}
    - title: No
      payload:
/select_grad_requirements_no{"program_gradrequirement_details_no":"Nopee"}

  utter_graduate_ask_school:
  - text: Would you like to know a bit more about the graduate requirements?
    buttons:
    - title: Yes
      payload: /select_gradschool_yes{"view_gradschool_yes":"Si"}
    - title: No
      payload: /select_gradschool_no{"view_gradschool_no":"Nay"}

  utter_grad_ask_steps:
  - text: Would you like to know a bit more about the graduate requirements?
    buttons:
    - title: Yes
      payload: /select_gradsteps_yes{"view_grad_steps_yes":"yasss"}
    - title: No
      payload: /select_gradsteps_no{"view_grad_steps_no":"nooooo"}

  utter_requirements:
  - text: The following is the requirements contains
    image:
file:///home/blueberrykary/Desktop/Final_Masters_Project/static/img/reselibil
ity1.png

  utter_graduate:
  - text: you are looking at graduate

  utter_welcome_basic:
  - text: welcome to future student

  utter_welcome_premium:
  - text: welcome to current student

  # utter_information:
  # - text: School Information
  #   buttons:
  #   - text: Help Desk
  #     payload: /help_inq

  utter_art_img:
  - text: The art department is located at <iframe width="180" height="180"
frameborder="0" style="border:0"
```

```
src="https://www.google.com/maps/embed/v1/place?key=AIzaSyAeAYpLBY6vo8ZSECeMa
dGx5Y9oLHcUw6s&q=CSUDH+LaCorte+Hall+(LCH)" allowfullscreen></iframe>

##============================================================================##
##============================================================================##
##============================================================================##
##============================================================================##

  utter_menu:
  - text: Which of the following would you like to contact us over?
    buttons:
    - title: Accounts & Access
      payload: /begin_lead
    - title: Hardware & Software
      payload: /begin_lead
    - title: IT Continuity Resources
      payload: /begin_lead
    - title: Network
      payload: /begin_lead
    - title: Teaching & Learning
      payload: /begin_lead
    - title: TV & Media Production
      payload: /begin_lead
    - title: University Printing Services

  utter_lead_q1:
  - text: Thank you, let's go to the next step.

  utter_ask_inquiry:
  - text: What would be your inquiry?

  utter_more_details:
  - text: Do you wish to provide further details?
    buttons:
    - title: "Yes"
      payload: "yes"
    - title: "No"
      payload: "no"

  utter_ask_details:
  - text: Please type in further details over your inquiry here.

  utter_ask_timeline:
  - text: When do you wish to be contacted?
    buttons:
    - title: Immediately
      payload: Immediately
    - title: Next Week
      payload: Next Week
    - title: Until the next available time
      payload: Until the next available time


  utter_ask_name:
  - text: May I know your name?
```

```
  utter_ask_email:
  - text: Could you provide me with your email?


  utter_ask_phone:
  - text: Can I have your phone number?


  utter_lead_q2:
  - text: Thanks for answering all our questions and being this patient.!


  utter_lead_q3:
  - text: "The information provided to contact you is: <br/> - inquiry:
{inquiry} <br/> -details: {details} <br/> - timeline: {timeline} <br/> -
name: {name} <br/> - email: {email} <br/> - phone: {phone}"



  utter_lead_q4:
  - text: Our help desk will keep in touch with you over your inquiry soon.!


  utter_lead_q5:
  - text: In the meanwhile for anything, you can contact us at (310) 243-2500
or visit
      our 24/7 Answers at [here](https://csudh.service-
now.com/it?id=kb_view2)

##============================================================================##
##============================================================================##
##============================================================================##
##============================================================================##

  utter_campus_info:
  - text: Some queries for campus
    buttons:
    - title: Campus Tour
      payload: /on_campus_tour{"campus_tour_CSUDH":"campus tour"}
    - title: Campus Map
      payload: /campus_map{"campus_map_CSUDH":"campus map"}
    - title: Student On-Campus Employement
      payload: /campus_employment{"campus_employment_CSUDH":"employment"}


  utter_enrollment:
  - text: "The student enrollment distribution for Fall 2015"
    custom:
      payload: chart
      data:
        title: Fall 2015
        labels:
        - Undergraduate
        - Graduate/PB
        - High School Students
        - Total
        backgroundColor:
```

```
        - "#36a2eb"
        - "#ffcd56"
        - "#ff6384"
        - "#009688"
        - "#c45850"
        chartsData:
        - 12562
        - 2073
        - 0
        - 14635
        chartType: pie
        displayLegend: 'true'


  utter_computerscience_option:
  - text: "Select one of the semesters"
    custom:
      payload: quickReplies
      data:
      - title: Spring 2020
        payload: /Spring2020
      - title: Summer 2020
        payload: /Summer2020


  utter_classes_spring_2020:
  - text: These are a list of Classes in computer science
    buttons:
    - title: CSC 582
      payload: /courses_spring20202{"spring_2020_course3":"csc 582"}
    - title: CSC 585
      payload: /courses_spring20201{"spring_2020_course2":"csc 585"}
    - title: CSC 590
      payload: /courses_spring2020{"spring_2020_course1":"csc 590"}

  utter_classes_summer_20201:
  - text: "These are a list of Classes in computer science"
    custom:
      payload: quickReplies
      data:
      - title: CSC 500
        payload: /courses_summer2020{"summer_2020_course1":"csc 500"}
      - title: CSC 501
        payload: /courses_summer20201{"summer_2020_course2":"csc 501"}

  utter_school_info:
  - text: school information image
    image:
"file:///home/blueberrykary/Desktop/Final_Masters_Project/static/img/art_depa
rt.png"

  utter_great:
  - text: Great! I'm glad to hear. Do you have any questions?

  utter_financial_aid:
  - text: "The following is a further explanation on FAFSA types offered at
CSUDH"
```

```
    custom:
        payload: "collapsible"
        data:
        - title: Grants
          description: CSUDH offers a variety of grants, ranging from the
Federal Pell Grant, TEACH Grants, State University Grants, and more.
        - title: Work-Study
          description: The Federal Work Study (FWS) Program provides funds to
employ students (on-campus or off-campus) who qualify for financial aid.
Students will be considered for an award by having submitted their FAFSA by
the established deadline, having submitted any requested documents by their
established deadline, and demonstrating financial need greater than $1,000.
Students must be enrolled at least half-time to be considered for this
program. Awards range from $1,000 to $4,000.
        - title: Loan Programs
          description: Contains a wide variety of grants ranging from the
California Dream Loan, Parent Loans, and more.

##============================================================================##
##============================================================================##
##============================================================================##
##============================================================================##
##============================================================================##

  utter_ask1:
  - text: What would you be looking for today?
  utter_ask2:
  - text: What would be your other question?
  utter_ask3:
  - text: What would be your other question?

  utter_goodbye:
  - text: Goodbye

  utter_default:
  - text: I'm sorry, I was not able to understand you. Would you please try
again?

##============================================================================##
##============================================================================##
## Chit-Chat

  utter_bored:
  - text: Here is a video of Teddy the Toro in an Earthquake Drill
    image: "https://media.giphy.com/media/UQmufdoqwUbVbgDr5U/giphy.gif"

  utter_sad:
  - text: Here is a funny gif to cheer you up!
    image: "https://media.giphy.com/media/LZElUsjl1Bu6c/giphy.gif"

  utter_lazy:
  - text: Nooo, don't get lazy!! Get back to work! =)

##============================================================================##
##============================================================================##
##============================================================================##
##============================================================================##
```

```
##============================ACTIONS=======================================##
actions:
- utter_introduction
- utter_introduction_r2
- utter_goodbye
- action_search_instruct
- action_goback
- action_goback2
- action_goback3
- action_goback4
- action_goback5
- action_schedule_appt
- action_schedule_university
- action_schedule_major
- action_choice_one
- action_choice_two
- action_hello_world
- utter_ask_type_of_student
- utter_welcome_premium
- utter_welcome_basic
- action_fetch_profile
- action_searchincomingstudent
- action_searchrecurringstudent
- utter_future_student
- utter_future_student_r2
- utter_future_student_r3
- utter_current_student
- utter_undergraduate
- utter_graduate
- action_search_undergraduateprogram
- action_search_graduateprogram
- utter_undergraduate_program_options
- utter_graduate_program_options
- action_display_undergradinfo
- utter_requirements
- action_req
- action_show
- action_requirement_information
- action_display_undergradreq
- action_display_uschoolinfo
- action_whento_apply
- action_progdeadline
- action_test
- utter_menu
- utter_more_details
- utter_lead_q1
- utter_lead_q2
- utter_lead_q3
- utter_lead_q4
- utter_lead_q5
- utter_program_information
- utter_ask_inquiry
- utter_ask_details
- utter_ask_timeline
- utter_ask_name
- utter_ask_email
- utter_ask_phone
```

```
- utter_thank_you_uprogram
- utter_curious_req
- utter_okay
- utter_option_program
- utter_ask_school
- utter_ask_steps
- action_prog_no
- action_prog_yes
- action_pro_req_yes
- action_pro_req_no
- action_school_yes
- action_school_no
- action_steps_apply_yes
- action_steps_apply_no
- action_graduatedd_program
- utter_grad_option_program
- action_grad_prog_yes
- action_grad_prog_no
- utter_grad_requirements
- action_grad_req_yes
- action_grad_req_no
- utter_graduate_ask_school
- action_gradinfo_yes
- action_gradinfo_no
- utter_grad_ask_steps
- action_grad_steps_apply_yes
- action_grad_steps_apply_no
- action_FAFSA
- utter_ask3
- action_Eenrollment
- utter_ask2
- action_campus_info
- utter_campus_info
- action_oncampus_tour
- action_campusmap
- action_employment
- action_course
- utter_great
- utter_ask1
- utter_enrollment
- utter_financial_aid
- utter_art_loc
- utter_art_program
- action_view_cs_course
- utter_computerscience_option
- utter_classes_summer_20201
- utter_classes_spring_2020
- action_view_spring2020
- action_view_spring20202
- action_view_spring20203
- utter_business_admin
- utter_business_admin_description
- utter_comp_sci
- utter_comp_sci_req
- action_view_summer2020
- action_view_summer20202
```

```
##============================================================================##
##============================================================================##
##============================================================================##
##==============================FORMS=========================================##
forms:
- lead_form_p1
- lead_form_p2
- lead_form_p3
```

File: nlu.md

```
## intent:greet
- hey
- hello
- hi
- good morning
- good evening
- hey there
- hi there
- what's up
- how's it going
- howdy
- evenin'
- sup
- heyyy
- hiii
- bonjour
- hola
- buenos dias
- good day
- hi there
- hi, how's it going
- hallo
- good afternoon

## intent: choice_text
- [text](choice)
- Return to the main menu
- I want to return to the main menu
- I want to ask a question
- I need regarding a question

## intent: choice_payloads
- [payloads](choice)
- I have [payloads](choice)
- I am selecting [payloads](choice)

## intent: student_type_one
- I am an [incoming student](student_incoming)
- I'm a [prospective student](student_incoming)
- I am an [applicant](student_incoming)
- I'm a [incoming undergraduate student](student_incoming)
- I'm an [incoming undergrad student](student_incoming)
- I'm an [incoming graduate student](student_incoming)
- I'm an [incoming grad student](student_incoming)
- I'm an [incoming freshman](student_incoming)
```

```
- I'm a [transfer student](student_incoming)
- I'm a [incoming first year student](student_incoming)
- I'm a [incoming master's student](student_incoming)

## intent: student_type_two
- I am an [ongoing student](student_recurr)
- I'm a [freshman](student_recurr)
- I'm a [sophmore](student_recurr)
- I'm a [junior](student_recurr)
- I'm a [senior](student_recurr)
- I'm a [master's student](student_recurr)
- I'm a [first year student](student_recurr)
- I'm a [second year student](student_recurr)
- I'm a [third year student](student_recurr)
- I'm a [fourth year student](student_recurr)
- I'm a [fifth year student](student_recurr)
- I'm a [graduate student](student_recurr)
- I'm a [undergraduate student](student_recurr)
- I am an [international student](student_recurr)
- I'm a [recurring student](student_recurr)
- I'm a [continuing student](student_recurr)
- I'm a [recurring undergraduate student](student_recurr)
- I'm a [recurring graduate student](student_recurr)
- I'm an [ongoing undergraduate student](student_recurr)
- I'm an [ongoing graduate student](student_recurr)

## intent: choose_incoming_undergrad_program
- I am interested in learning more about the [undergraduate
program](program_undergrad)
- Tell me over the  [undergrad program](program_undergrad)
- I want to know more about the [undergraduate program](program_undergrad)
- Could you please tell me more about the [undergraduate
program](program_undergrad)
- Please tell me more about the [undergraduate program](program_undergrad)


## intent: choose_incoming_grad_program
- I am interested in learning more about the [graduate
program](program_graduate)
- Tell me more about the [grad program](program_graduate)
- Can you please tell me more about the [graduate program](program_graduate)
- I am curious about the [graduate program](program_graduate)

## intent: undergrad_programs_info
- Tell me more about the [undergrad program](program_undergrad_info)
- Selecting [undergrad program](program_undergrad_info)
- Choosing [undergrad_program](program_undergrad_info)

<!-- ## intent: undergrad_programs_requirements
- What are the [requirements](program_undergrad_req) to apply?
-  Yes, tell me more about the [program requirements](program_undergrad_req).
- Yes, please tell me more about the [requirements](program_undergrad_req).
- Yes, I'd like to know more about the [requirements](program_undergrad_req).
-->


<!-- ## intent: undergrad_programs_apply
```

```
- What are the steps to [applying](program_undergrad_apply)? -->

<!-- ## intent: undergrad_programs_where_apply
- Where do I [apply](program_under_when_apply)? -->



<!--============================================================-->
## intent: grad_program_info
- Tell me more about the [graduate program](program_masters)
- Need to know more over the [graduate program](program_masters)
- I don't know anything about the [graduate program](program_masters)

<!-- ## intent: help_inq
  - Help Desk
  - Help Desk
  - help_inq
  - Help use
  - Help done -->

<!--================================================-->
<!--================================================-->
<!--================================================-->
<!--================================================-->
<!--================================================-->
## intent: select_program_details_yes
  - [Of Course](program_details_yes)
  - I am gonna choose [Of Course](program_details_yes)
  - Gotta know some more, so [Of Course](program_details_yes)
<!--==================YES OR NO=====================-->
## intent: select_program_details_no
- [Of Course Not](program_details_no)
- Well, [Of Course Not](program_details_no)
- I can't, [Of Course Not](program_details_no)

<!--================================================-->
<!--================================================-->
<!--================================================-->
<!--================================================-->
## intent: select_requirements_yes
  - [Indeed](program_requirement_details_yes)
  - Why, [Indeed](program_requirement_details_yes)
<!--==================YES OR NO=====================-->
## intent: select_requirements_no
  - [Definately Not](program_requirement_details_no)
  - Totally, [Definately Not](program_requirement_details_no)
<!--================================================-->
<!--================================================-->
<!--================================================-->
<!--================================================-->
## intent: select_school_info_yes
  - [Yessss](school_info_yes)
  - Don't mind if I do, [Yessss](school_info_yes)
<!--==================YES OR NO=====================-->
## intent: select_school_info_no
  - [Nooo](school_info_no)
  - Don't even bring about it [Nooo](school_info_no)
```

```
<!--==================================================-->
<!--==================================================-->
<!--==================================================-->
<!--==================================================-->
## intent: select_stepstoapply_yes
  - [Steps yes](steps_apply_yes)
  - I would like to know more on the [Steps yes](steps_apply_yes) to apply
<!--==================YES OR NO=====================-->
## intent: select_stepstoapply_no
  - [Steps no](steps_apply_no)
  - I don't think I want to know more [Steps no](steps_apply_no)
<!--==================================================-->
<!--==================================================-->
<!--==================================================-->
<!--==================================================-->


<!-- ## intent: program_requirements_yes
- Yes, I'd like to know more
- Yes, please tell me more about the requirements
- Yes please, tell me more
- Yes
- Yes I would like to know
- Yes please, let me know more
- Yes please, thank you -->



<!-- ## intent: By_Major
- [Major](school) -->

<!--==================================================-->
<!--==================================================-->
<!--==================================================-->
<!--==================================================-->
<!--==================================================-->
## intent: select_grad_program_details_yes
  - [Yupp](program_grad_details_yes)
  - I am going to select [Yupp](program_grad_details_yes)
<!--==================YES OR NO=====================-->
## intent: select_grad_program_details_no
- [Noope](program_grad_details_no)
- I don't want to select the [Noope](program_grad_details_no)

<!--==================================================-->
<!--==================================================-->
<!--==================================================-->
<!--==================================================-->
<!--==================================================-->
<!--==================================================-->
## intent: select_grad_requirements_yes
  - [Yas](program_gradrequirement_details_yes)
  - I am gonna view the requirements
[Yas](program_gradrequirement_details_yes)
<!--==================YES OR NO=====================-->
## intent: select_grad_requirements_no
  - [Nopee](program_gradrequirement_details_no)
```

```
  - I am not gonna view the grad requirements
[Nopee](program_gradrequirement_details_no)
<!--==================================================-->
<!--==================================================-->
<!--==================================================-->
<!--==================================================-->
<!--==================================================-->
<!--==================================================-->
<!--==================================================-->
## intent: select_gradschool_yes
  - [Si](view_gradschool_yes)
  - I am gonna select to view gradschool [Si](view_gradschool_yes)
<!--==================YES OR NO=====================-->
## intent: select_gradschool_no
  - [Nay](view_gradschool_no)
  - It's not worth my time to view gradschool [Nay](view_gradschool_no)
<!--==================================================-->
<!--==================================================-->
<!--==================================================-->
## intent: select_gradsteps_yes
  - [yasss](view_grad_steps_yes)
  - I am gonna view the steps for grad school [yasss](view_grad_steps_yes)
<!--==================YES OR NO=====================-->
## intent: select_gradsteps_no
  - [nooooo](view_grad_steps_no)
  - I don't want to view the graduate school steps
[nooooo](view_grad_steps_no)
<!--==================================================-->
<!--==================================================-->
<!--==================================================-->

## intent: good_thanks
- good, thank you
- good
- good, thanks
- good, I'm doing great
- great
- ok
- good, uh thanks
- yeah, I'm good could be better
- good, how about you!
- I'm good, thank you
- I'm doing great
- very good, thanks
- doing great
- I am alright
- yeah, I'm good
- yup, I'm totally good
- I'm good, thanks
- uh, thanks


## intent: yes_ido
- yes, i do
- yes, I have a question
- yes I do have some questions
- yup
```

```
- indeed
- yes, I do
- confirm
- I do
- yes, I have other questions
- yes, please
- yes, I want to know more
- yes, of course

## intent: ask_a_question
- I have a question
- I have another question
- question
- I have another question to ask
- I need to ask you something else
- I want to know more over another subject
- Can I know more about something else?
- More information please
- More information

## intent: no_more_ques
- I have no more questions
- I don't have anymore questions
- I don't have any questions
- I'm done
- thank you
- thanks
- thanks, I'm good

<!-- ## intent: mention_other_ques
- I have a question
- I have a question
- I have another question
- question
- I have another question to ask
- I need to ask you something else
- I want to know more over another subject
- I want to ask a question
- Can I know more about something else?
- More information please
- More information
- I have another question -->

## intent: on_campus
- I want to know what the [campus](campus_CSUDH) has?
- I want to know the things the campus has [campus](campus_CSUDH)
- I want to see what things the campus has [campus](campus_CSUDH)
- [campus](campus_CSUDH)

## intent: on_campus_tour
-  I want to see [campus tour](campus_tour_CSUDH)
- I want a [campus tour](campus_tour_CSUDH)
- I want to go to a [campus tour](campus_tour_CSUDH)

## intent: campus_map
- I want to see the [campus map](campus_map_CSUDH)
- I want to view the [campus map](campus_map_CSUDH)
```

```
## intent: campus_employment
- I want to see some [employment](campus_employment_CSUDH)
- I want to view the options in [employment](campus_employment_CSUDH)

<!-- ## intent: no_idont
- nope
- no
- no i don't
- not even close
- nahhh
- i don't
- nah
- nah, I'm good
- that's all, thank you
- thank you
- I'm done, thank you
- I'm done
- I'm all good
- I'm done, thanks -->

## intent: financial_aid_application
- I have a question regarding [FAFSA](FAFSA_APP)
- I have a question in regards to [financial aid](FAFSA_APP)
- I have am looking into [financial aid](FAFSA_APP)
- [FAFSA](FAFSA_APP)
- [financial aid](FAFSA_APP)

## intent: university_enrollment
- I have a question about [enrollment](uni_enroll)
- I have a question over [enrollment](uni_enroll)
- When is [enrollment](uni_enroll)
- I want to know when is enrollment [enrollment](uni_enroll)
- I got a question over [enrollment](uni_enroll)
- I'm looking for [enrollment](uni_enroll)
- I don't know when is [enrollment](uni_enroll)
- [enrollment](uni_enroll)

## intent:accept
- yes
- right
- okay
- sure
- fine
- it's ok
- it is okay
- of cause
- ofcause

## intent:reject
- no
- don't
- dont
- do not
- please no
- no please
- never
```

```
- don't do

## intent:begin_lead
- begin lead
- Accounts & Access
- Hardware & Software
- IT Continuity Resources
- Network
- Teaching & Learning
- TV & Media Production
- University Printing Services


## intent: art
- art
- art class
- art and humanities
- studio art

## intent: business_admin
- business administration
- business administration: enterprise concentration
- business administration: accounting concentration
- business administration: information systems concentration

## intent: computer_science
- computer science and information systems
- computer science and technology

## intent: course_description
- [computer science](getting_course)
- I am looking for [computer science](getting_course)
- [computer technology](getting_course)
- I am looking into [computer science](getting_course)
- looking for [comp science](getting_course)
- looking into [comp sci](getting_course)


<!-- ## intent: corona_state
- [california](state)
- [washington](state)
- [oregon](state)
- [united states](state) -->

## intent:bot_challenge
- are you a bot?
- are you a human?
- am I talking to a bot?
- am I talking to a human?


## intent: Spring2020
- Spring 2020
- I want to view the classes for Spring 2020
- I want to look at the Spring 2020 semester

## intent: Summer2020
```

```
- Summer 2020
- I want to look at the Summer 2020 semester
- I want to view the classes in Summer 2020

## intent: courses_spring2020
- [csc 590](spring_2020_course1)
- I want to see the [csc 590](spring_2020_course1)

## intent: courses_spring20201
- [csc 585](spring_2020_course2)
- I want to view the class [csc 585](spring_2020_course2)

## intent: courses_spring20202
- [csc 582](spring_2020_course3)
- I want to view the master's course [csc 582](spring_2020_course3)

## intent: courses_summer2020
- [csc 500](summer_2020_course1)
- I want to view the summer classes for [csc 500](summer_2020_course1)

## intent: courses_summer20201
- [csc 501](summer_2020_course2)
- I want to see the summer courses for [csc 501](summer_2020_course2)



## intent:goodbye
- bye
- goodbye
- see you around
- see you later


## intent: bored
- I am so bored
- I am bored
- This is so boring, do you have anything else?
- I am so boreddddd
- bored as hell
- bored, please entertain me

## intent: sad
- I am sad
- I want to cry
- I am so sad

## intent: lazy
- I am feeling super lazy
- I am lazy right now
- Don't feel like doing anything right know
- SOS, I am lazy
<!--
## intent: happy
- I am so happy
- happy all over
- I am overjoyed
- Omg, I am so happy
```

```
- I am happy

## intent: graduated
- I graduated
- yay, I graduated
- I am so happy I graduated
- graduated and passed all my exams
- just graduated
- I just graduated

## intent: tired
- I am so tired
- I'm tired
- I am tired
- omg, I am very tired

## intent: stressed
- I am feeling very stressed
- I am stressed out right now
- I am stressed as hell
- I am stressed -->
```

File: dbConnect.py

```
import psycopg2
import traceback

def getData(query:str):
    try:
        mydb = psycopg2.connect(
        host = "localhost",
        database = "masterdb",
        user = "postgres",
        password = "Ks126352"
        )

        cursor = mydb.cursor()
        cursor.execute(query)

        results = cursor.fetchall()
        return results
    except:
        print("Error occured while connecting to database or fetching data
from database. Error Trace: {}".format(traceback.format_exc()))
        return []
```

File: HelpDeskForm.py

```
## Inquiry form for story in "Inquiries.md"

from typing import Dict, Text, Any, List, Union, Optional
from rasa_sdk import Tracker
from rasa_sdk.executor import CollectingDispatcher
```

```
from rasa_sdk.forms import FormAction

##=========================================================================##
##=========================================================================##
##=========================================================================##
##=========================================================================##
##======================Lead Form First-===================================##
class LeadFormFirstPart(FormAction):

    def name(self) -> Text:
        # Unique identifier of the form
        return "lead_form_p1"

    @staticmethod
    def required_slots(tracker: Tracker) -> List[Text]:
        # A list of required slots that the form has to fill
        return ["inquiry"]

    def slot_mappings(self) -> Dict[Text, Union[Dict, List[Dict]]]:
        return {
            "inquiry": [
                self.from_text(),
            ],
        }

    def submit(
        self,
        dispatcher: CollectingDispatcher,
        tracker: Tracker,
        domain: Dict[Text, Any],
    ) -> List[Dict]:

        # utter submit template
        dispatcher.utter_message(template="utter_more_details")
        return []
##=========================================================================##
##=========================================================================##
##=========================================================================##
##=========================================================================##
##======================Lead Form Second===================================##
class LeadFormSecondPart(FormAction):
    def name(self) -> Text:
        return "lead_form_p2"

    @staticmethod
    def required_slots(tracker: Tracker) -> List[Text]:
        return ["details"]

    def slot_mappings(self) -> Dict[Text, Union[Dict, List[Dict]]]:

        return {
            "details": [
                self.from_text(),
            ],
        }

    def submit(
```

```
        self,
        dispatcher: CollectingDispatcher,
        tracker: Tracker,
        domain: Dict[Text, Any],
    ) -> List[Dict]:

        return []
##=========================================================================##
##=========================================================================##
##=========================================================================##
##=========================================================================##
##=====================Lead Form Three=====================================##

class LeadFormThirdPart(FormAction):
    def name(self) -> Text:

        return "lead_form_p3"

    @staticmethod
    def required_slots(tracker: Tracker) -> List[Text]:
        return ["timeline", "name", "email", "phone"]

    def slot_mappings(self) -> Dict[Text, Union[Dict, List[Dict]]]:

        return {
            "timeline": [
                self.from_text(),
            ],
            "name": [
                self.from_text(),
            ],
            "email": [
                self.from_text(),
            ],
            "phone": [
                self.from_text(),
            ],
        }

    def submit(
        self,
        dispatcher: CollectingDispatcher,
        tracker: Tracker,
        domain: Dict[Text, Any],
    ) -> List[Dict]:

        # utter submit response
        dispatcher.utter_message(template="utter_lead_q2")
        return []
```

File: IncomingGraduateStudent.py

```
## For incoming graduate student
## Using data from the database
```

```
from typing import Dict, Text, Any, List, Union, Optional

from rasa_sdk import Action, Tracker
from rasa_sdk.executor import CollectingDispatcher
from rasa_sdk.events import SlotSet
import requests
import json
import re
import sqlalchemy
from dbConnect import getData

class Action_GraduatePrograming(Action):

    def name(self) -> Text:
        return "action_graduatedd_program"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #school information slot
        grad_program = tracker.get_slot("program_masters")
        #the response from uschool_iinfo
        message = "CSUDH offers a wide selection of outstanding degrees that
reflect today’s in-demand fields. Our degree programs offer a well-rounded
curriculum that combines rigorous academics with a practical education."
        message2 = "CSUDH offers 44 undergraduate majors with high-demand
options or concentrations recognized for their quality, well-rounded
curricula, and balance of classroom learning and practical experience."
        #select the school's department information
        q = 'SELECT dept.department_name FROM school as s INNER JOIN
belongs_to_school_depart as b_dept ON s.school_id = b_dept.school_id INNER
JOIN department as dept ON b_dept.department_id = dept.department_id'
        #pass the sql query to the getData method and store the results in
`data` variable.
        data = getData(q)
        #list out the departments
        set_one = str(data[0][0])
        set_two = str(data[1][0])
        set_three = str(data[2][0])
        set_four = str(data[3][0])
        set_five = str(data[4][0])
        set_six = str(data[5][0])
        #combine the departments with print out message
        set_combined  = set_one + "\n\n" + set_two + "\n\n" + set_three +
"\n\n" + set_four + "\n\n" + set_five + "\n\n" + set_six
        #combine all the messages to one
        ult_message = message + "\n\n" + message2 + "\n\n" + set_combined +
"\n\n"
        graduate_program_details = ult_message
        #uttering message containing the school response from slot
        dispatcher.utter_message(text="{}".format(graduate_program_details))
        #returning the slot for school information meant for incoming
graduate students
        return [SlotSet("graduate_program_details", grad_program)]
#
# ##================================DO NOT go to program
details?================#
```

```
# class Action_UndergradProgramNo(Action):
#
#     def name(self) -> Text:
#         return "action_prog_no"
#
#     def run(self, dispatcher: CollectingDispatcher,
#             tracker: Tracker,
#             domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
#         #school information slot
#         prog_no = tracker.get_slot("program_details_no")
#         #the response from uschool_iinfo
#         program_no = "no"
#         #uttering message containing the school response from slot
#         dispatcher.utter_message(text="You said {}".format(program_no))
#
#         #returning the slot for school information meant for incoming
graduate students
#         return [SlotSet("program_no", prog_no)]
##=========================================================================#
##=========================================================================#
##=========================================================================#
##=========================================================================#
##=========================Go to the program requirements? ================#
##======FILL THIS ONE IN!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!==========#
class Action_GradProg_Yes(Action):

    def name(self) -> Text:
        return "action_grad_prog_yes"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #school information slot
        grad_prog_yes = tracker.get_slot("program_grad_details_yes")

        message = "Pick up to 21 programs that match your needs, goals and
personality and the research you have done on different schools. You will
then need to make a time line for each program."
        view_more_programs = '<a href ="https://www.csudh.edu/gsr/graduate-
studies/degree-programs/">View Programs</a>'

        overall = message + "<br/><br/>" + view_more_programs
        graduate_prog_yes = overall
        #uttering message containing the school response from slot
        dispatcher.utter_message(text="{}".format(graduate_prog_yes))

        #returning the slot for school information meant for incoming
graduate students
        return [SlotSet("graduate_prog_yes", grad_prog_yes)]

##=====================DO NOT go to program requirements?================#
class Action_GradProg_No(Action):

    def name(self) -> Text:
        return "action_grad_prog_no"

    def run(self, dispatcher: CollectingDispatcher,
```

```
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #school information slot
        grad_prog_no = tracker.get_slot("program_grad_details_no")
        #the response from uschool_iinfo
        graduate_prog_no = ""
        #uttering message containing the school response from slot
        dispatcher.utter_message(text="".format(graduate_prog_no))

        #returning the slot for school information meant for incoming
graduate students
        return [SlotSet("graduate_prog_no", grad_prog_no)]
##=========================================================================#
##=========================================================================#
##=========================================================================#
class Action_GradReqYes(Action):

    def name(self) -> Text:
        return "action_grad_req_yes"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #school information slot
        grad_progreq_yes =
tracker.get_slot("program_gradrequirement_details_yes")
        #the response from uschool_iinfo
        req = "Completion of a four-year college course of study and an
acceptable baccalaureate degree from an institution accredited by a regional
accrediting association or completion of equivalent academic preparation as
determined by appropriate campus authorities. <br> Good academic standing at
the last college or university attended. <br> A grade point average of at
least 2.5 (A = 4.0) in the last 60 semester (90 quarter) units attempted.
<br> Satisfactory adherence to the professional, personal, scholastic, and
other standards for graduate study, including qualifying examinations, as
appropriate campus authorities may prescribe."

        req_info = '<a href ="https://www.csudh.edu/gsr/graduate-
studies/graduate-admission/">Graduate Requirements</a>'

        overall = req + "<br/>" + req_info
        graduate_progreq_yes = overall
        #uttering message containing the school response from slot
        dispatcher.utter_message(text="{}".format(graduate_progreq_yes))

        #returning the slot for school information meant for incoming
graduate students
        return [SlotSet("graduate_progreq_yes", grad_progreq_yes)]

##=====================DO NOT go to program requirements?================#
class Action_GradReqNo(Action):

    def name(self) -> Text:
        return "action_grad_req_no"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
```

```
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #school information slot
        grad_progreq_no =
tracker.get_slot("program_gradrequirement_details_no")
        #the response from uschool_iinfo
        graduate_progreq_no = ""
        #uttering message containing the school response from slot
        dispatcher.utter_message(text="".format(graduate_progreq_no))

        #returning the slot for school information meant for incoming
graduate students
        return [SlotSet("graduate_progreq_no", grad_progreq_no)]

##=========================================================================#
##=========================================================================#
##=========================================================================##
##=========================================================================##
##=========================================================================##

class Action_GradSchoolInfo_Yes(Action):

    def name(self) -> Text:
        return "action_gradinfo_yes"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #school information slot
        gradinfo_yes = tracker.get_slot("view_gradschool_yes")

        why_grad = " REASONS TO GO TO GRAD SCHOOL: <br/> <br/>A requirement
for the career; for example law, medicine, social work, occupational therapy,
counseling. <br/><br/> To move up in a field of work or earn more income;
e.g., nursing, business management, teaching, public administration.
<br/><br/> To prepare for a research career or to teach in higher education.
<br/><br/>  To change careers <br/><br/> For personal growth"
        how_to_begin = "\n\n HOW TO BEGIN CONSIDERATION TO GRAD SCHOOL:
<br/><br/> Consider the reasons you want to attend graduate school. Evaluate
your interests, skills, values and goals in determining the program that
suits you best. Think about how this decision will affect you and other
people in your life. You want to spend a lot of time making an informed
decision. <br/> The Career Center can help:<br/><br/> Attend a Graduate
School Search Workshop offered by our office <br/> Attend the annual Graduate
School Fair sponsored by the Career Center <br/> Talk with a Career Coach
<br/> Visit the Career Center's Career Resource Library\n\n"
        admin_req = "ADMISSION REQUIREMENTS <br/><br/> May include the
following:<br/><br/> Baccalaureate degree <br/>Minimum grade point average
(cumulative or last 60 units) <br/>Minimum score on a standardized entrance
examination (e.g., GRE, GMAT, MSAT) <br/> Letters of recommendation (usually
3) <br/> Experience in the field\n\n"
        app_process = "APPLICATION PROCESS<br/><br/>Complete Application -
May be required for university and individual program.<br/><br/> Application
Fee (usually $50 to $75)<br/> Transcripts - Official transcripts to be sent
from each from college or university attended. <br/>  Statement of Purpose
<br/> Letters of Recommendation (usually 3, typically professors) <br/>
Required Entrance Examinations - Prepare for and take any assessment
required. There are many preparation courses you can take with a variety of
```

```
costs. Following are the websites for some of the most common exams:
<br/><br/>"

        gre = '<a href ="http://www.ets.org/gre">GRE</a>'
        lsat = '<a href ="https://www.lsac.org/lsat">LSAT</a>'
        pharm = '<a href ="http://www.pcat.info/">PCAT</a>'
        gmat = '<a href ="https://www.mba.com/">GMAT</a>'
        for_more1 = "For more information over graduate school, please go to
the link: "
        for_more = '<a href ="https://www.csudh.edu/career-
center/students/graduate-school/">Grad School Information</a>'

        overall = why_grad + how_to_begin + admin_req + app_process + gre +
"<br/>" + lsat + "<br/>" + pharm + "<br/>" + gmat + "<br/>" + for_more1 +
for_more
        graduateinfo_yes = overall

        dispatcher.utter_message(text="{}".format(graduateinfo_yes))

        #returning the slot for school information meant for incoming
graduate students
        return [SlotSet("graduateinfo_yes", gradinfo_yes)]

##================================DO NOT go to program
requirements?================#
class Action_GradSchoolInfo_No(Action):

    def name(self) -> Text:
        return "action_gradinfo_no"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #school information slot
        grad_progreq_no = tracker.get_slot("view_gradschool_no")
        #the response from uschool_iinfo
        graduate_progreq_no = ""
        #uttering message containing the school response from slot
        dispatcher.utter_message(text="".format(graduate_progreq_no))

        #returning the slot for school information meant for incoming
graduate students
        return [SlotSet("graduate_progreq_no", grad_progreq_no)]
##=========================================================================##
##=========================================================================##
#==========================================================================##
#==========================================================================##
##============================Go to the program details?================#
class Action_GradStepsYes(Action):

    def name(self) -> Text:
        return "action_grad_steps_apply_yes"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #school information slot
```

```
        gradsteps_yes = tracker.get_slot("view_grad_steps_yes")
        #the response from uschool_iinfo
        how_to = '<a href ="https://www.csudh.edu/gsr/graduate-
studies/graduation/how-to-apply/">How To Apply</a>'
        when_to = '<a href ="https://www.csudh.edu/gsr/graduate-
studies/graduation/when-to-apply/">When To Apply</a>'
        candinacy_info = "The degree is awarded upon the satisfactory
completion of all state and university requirements, the specific
requirements for the particular program, the recommendation of the
appropriate graduate adviser and program coordinator (advancement to
candidacy), and the approval of the faculty and Dean of Graduate
Studies.<br/>"
        how_to_know = '<a href ="https://www.csudh.edu/gsr/graduate-
studies/graduation/requirements/advancement-to-candidacy/">Advancement To
Candidancy</a>'
        gwar = '<a href ="https://www.csudh.edu/gsr/graduate-
studies/graduation/requirements/gwar/">GWAR</a>'
        apply_here = '<a href
="https://www2.calstate.edu/apply/graduate">Apply Here</a>'
        time_lim = "Upon acceptance, all requirements for the master's
degree, including all course work on the student's approved program of study
must be completed within five years (some programs permit seven years). This
time limit commences with the semester of the earliest course used on the
student's program of study."

        message = candinacy_info + "<br/>" + time_lim + "<br/><br/>" + how_to
+ "<br/>" + apply_here + "<br/>" + when_to + "<br/>" + how_to_know + "<br/>"
+ gwar

        graduatesteps_app_yes = message
        #uttering message containing the school response from slot
        dispatcher.utter_message(text="{}".format(graduatesteps_app_yes))

        #returning the slot for school information meant for incoming
graduate students
        return [SlotSet("graduatesteps_app_yes", gradsteps_yes)]

##============================DO NOT go to program details?================#
class Action_GradStepsNo(Action):

    def name(self) -> Text:
        return "action_grad_steps_apply_no"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #school information slot
        gradsteps_no = tracker.get_slot("view_grad_steps_no")
        #the response from uschool_iinfo
        graduatesteps_app_no = ""
        #uttering message containing the school response from slot
        dispatcher.utter_message(text="".format(graduatesteps_app_no))

        #returning the slot for school information meant for incoming
graduate students
        return [SlotSet("graduatesteps_app_no", gradsteps_no)]
##=============================================================================#
```

File: IncomingUndergraduateStudent.py

```
## For incoming undergraduate student
## Using data from the database

from typing import Dict, Text, Any, List, Union, Optional

from rasa_sdk import Action, Tracker
from rasa_sdk.executor import CollectingDispatcher
from rasa_sdk.events import SlotSet
import requests
import json
import re
import sqlalchemy
from dbConnect import getData
##========================================================================#
##========================================================================#
##========================================================================#
##=================Display Undergraduate Program Information ============##
class Action_UnderProgramSearchInfo(Action):

    def name(self) -> Text:
        return "action_display_undergradinfo"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #undergraduate information program slot
        uprog_info = tracker.get_slot("program_undergrad_info")
        ## message and message 2 will be combined with query 'q'
        message = "CSUDH offers a wide selection of outstanding degrees that
reflect today’s in-demand fields. Our degree programs offer a well-rounded
curriculum that combines rigorous academics with a practical education."
        message2 = "CSUDH offers 44 undergraduate majors with high-demand
options or concentrations recognized for their quality, well-rounded
curricula, and balance of classroom learning and practical experience."
        #select the school's department and show its name.
        q = 'SELECT dept.department_name FROM school as s INNER JOIN
belongs_to_school_depart as b_dept ON s.school_id = b_dept.school_id INNER
JOIN department as dept ON b_dept.department_id = dept.department_id'
        #pass the sql query to the getData method and store the results in
`data` variable.
        data = getData(q)
        #list out the departments
        set_one = str(data[0][0])
        set_two = str(data[1][0])
        set_three = str(data[2][0])
        set_four = str(data[3][0])
        set_five = str(data[4][0])
        set_six = str(data[5][0])
        #combine the departments with print out message
        set_combined  = set_one + "\n\n" + set_two + "\n\n" + set_three +
"\n\n" + set_four + "\n\n" + set_five + "\n\n" + set_six
        #combine all the messages to one
```

```
        ult_message = message + "\n\n" + message2 + "\n\n" + set_combined +
"\n\n"
        #the response from uprog_iinfo
        uprog_iinfo = ult_message
        # utters message as {} along with link
        dispatcher.utter_message(text="{}".format(uprog_iinfo))
        #store the message into slot
        return [SlotSet("uprog_info", uprog_iinfo)]
##=========================================================================#
##=========================================================================#
##=========================================================================#
##============================Go to the program details====================#

class Action_UndergradProgramYes(Action):

    def name(self) -> Text:
        return "action_prog_yes"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        prog_yes = tracker.get_slot("program_details_yes")
        ## prints empty message
        message =  ""
        ## programs are listed in the utterance since it provides payloads
        #the response from program_yes
        program_yes = message
        #uttering message containing the program response from slot
        dispatcher.utter_message(text="".format(program_yes))
        #returning the slot for program meant for incoming undergraduate
students
        return [SlotSet("program_yes", prog_yes)]

##============================DO NOT go to program details=================#
class Action_UndergradProgramNo(Action):

    def name(self) -> Text:
        return "action_prog_no"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        ##prints an empty message or program_no
        prog_no = tracker.get_slot("program_details_no")
        #the response from program_no, but is empty
        program_no = ""
        #uttering message containing the program no response from slot
        dispatcher.utter_message(text="".format(program_no))
        #returning the slot for program_no meant for incoming undergraduate
students
        return [SlotSet("program_no", prog_no)]
##=========================================================================#
##=========================================================================#
##=========================================================================#
##============================Go to the program requirement===============#
class Action_UndergradReqYes(Action):
```

```
    def name(self) -> Text:
        return "action_pro_req_yes"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #program requirement slot
        prog_req_yes = tracker.get_slot("program_requirement_details_yes")
        ## messages combined to be uttered
        message = "CSUDH is not impacted for First-Time Freshmen Applicants.
"
        message2 = "The university's criteria for admission as a first-time
freshman are met if a student \n"
        message3 = "1. Is a high school graduate; "
        message4 = "2. Has a qualifiable eligibility index; and"
        message5 = "3. Has completed with grades of C or better each of the
courses in the comprehensive pattern of college preparatory subject
requirements. Courses must be completed prior to the first enrollment in the
California State University."
        out = message + "\n\n" + message2 + "\n\n" + message3 + "\n\n" +
message4 + "\n\n" + message5 + "\n\n"
        program_req_yes = out
        #uttering message containing program requirement yes response from
slot
        dispatcher.utter_message(text="{}".format(program_req_yes))
        #returning the slot for program requirement yes for incoming
undergraduate students
        return [SlotSet("program_req_yes", prog_req_yes)]

##======================DO NOT go to program requirements================#
class Action_UndergradReqNo(Action):

    def name(self) -> Text:
        return "action_pro_req_no"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #school information slot
        prog_req_no = tracker.get_slot("program_requirement_details_no")
        #the response from uschool_iinfo
        program_req_no = ""
        #uttering message containing the school response from slot
        dispatcher.utter_message(text="".format(program_req_no))
        #returning the slot for school information meant for incoming
graduate students
        return [SlotSet("program_req_no", prog_req_no)]
##=========================================================================#
##=========================================================================#
##=========================================================================#
##===================Go to the program School Information==================#
class Action_SchoolInfoYes(Action):

    def name(self) -> Text:
        return "action_school_yes"

    def run(self, dispatcher: CollectingDispatcher,
```

```
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #school information slot
        schooli_yes = tracker.get_slot("school_info_yes")
        ## query to get school information from information table
        q = 'select  i.description from school as sc INNER JOIN contains_info
as ci ON ci.school_id = sc.school_id INNER JOIN information as i ON
ci.info_id = i.info_id'
        #pass the sql query to the getData method and store the results in
`data` variable.
        data = getData(q)
        #Print out school information
        details = "School Information"
        seperator = "========================================="
        set_one = str(data[0][0]) + "\n\n" +  str(data[1][0])
        set_combined = details + "\n" + seperator + "\n" +  set_one + "\n"
        #the response from school information yes
        schoolinfo_yes = set_combined
        #uttering message containing the school information response from
slot
        dispatcher.utter_message(text="{}".format(schoolinfo_yes))
        #returning the slot for school information meant for incoming
graduate students
        return [SlotSet("schoolinfo_yes", schooli_yes)]

##================================DO NOT go to program
requirements?================#
class Action_SchoolInfoNo(Action):

    def name(self) -> Text:
        return "action_school_no"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #school information slot
        schooli_no = tracker.get_slot("school_info_no")
        #the response from school information no, but is empty
        schoolinfo_no = ""
        #uttering message containing the school information no response from
slot
        dispatcher.utter_message(text="".format(schoolinfo_no))
        #returning the slot for school information no  meant for incoming
graduate students
        return [SlotSet("schoolinfo_no", schooli_no)]
##============================================================================#
##============================================================================#
##============================================================================#
##============================Go to the program details?==================#
class Action_Steps_to_apply_yes(Action):

    def name(self) -> Text:
        return "action_steps_apply_yes"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
```

```
        #school information slot
        steps_yes = tracker.get_slot("steps_apply_yes")
        #the response from steps to apply
        q = 'select si.step_name, si.step_description from school as sc INNER
JOIN views_steps as v ON v.school_id = v.school_id INNER JOIN steps_info as
si ON v.steps_id = si.steps_id where si.steps_id = 1 OR si.steps_id = 2 OR
si.steps_id = 3 OR si.steps_id = 4'
        #pass the sql query to the getData method and store the results in
`data` variable.
        data = getData(q)
        #Print out the steps to apply for undergraduate students
        details = "The following are steps to apply to CSUDH for Incoming
Undergraduate Students"
        set_one = str(data[0][0]) + " : " +  str(data[0][1])
        set_two = str(data[1][0]) + " : " + str(data[1][1])
        set_three = str(data[2][0]) + " : " + str(data[2][1])
        set_four = str(data[3][0]) + " : " + str(data[3][1])
        ## combining all of the queries together to create one overall
utterance
        set_combined = details + "\n" + set_one + "\n\n" + set_two + "\n\n" +
set_three + "\n\n" + set_four
        ## combined for steps to apply yes
        steps_app_yes = set_combined
        #uttering message containing the steps to apply yes response from
slot
        dispatcher.utter_message(text="{}".format(steps_app_yes))
        #returning the slot for steps to apply yes meant for incoming
graduate students
        return [SlotSet("steps_app_yes", steps_yes)]

##============================DO NOT go to program details?================#
class Action_Steps_to_apply_no(Action):

    def name(self) -> Text:
        return "action_steps_apply_no"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        steps_no = tracker.get_slot("steps_apply_no")
        #the response for steps app no, with empty message
        steps_app_no = ""
        #uttering message containing the steps app no response from slot
        dispatcher.utter_message(text="".format(steps_app_no))
        #returning the slot for school app no meant for incoming graduate
students
        return [SlotSet("steps_app_no", steps_no)]
##============================================================================#
##============================================================================#
##============================================================================#
##============================================================================#
##============================================================================#
```

File: Introduction_CSUDHBot.py

```
## utter_introduction_r2

from typing import Any, Text, Dict, List

from rasa_sdk import Action, Tracker
from rasa_sdk.executor import CollectingDispatcher
from rasa_sdk.events import SlotSet
import requests
import json
import re
##========================================================================#
##========================================================================#
##========================================================================#
##========================================================================#
#This is meant for the introduction to select the "Inquiry" option
class ActionChoice_One(Action):

    def name(self) -> Text:
        return "action_choice_one"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

            choice_one = tracker.get_slot("choice")
            c_one = "input an inquiry"
            dispatcher.utter_message("You selected to {} was your
option".format(c_one))

            return[SlotSet("choice_one", c_one)]
##========================================================================#
##========================================================================#
##========================================================================#
##========================================================================#
#This is meant for the introduction to select the "Ask Questions to CSUDHBot"
option
class ActionChoice_Two(Action):

    def name(self) -> Text:
        return "action_choice_two"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

            choice_two = tracker.get_slot("choice")
            c_two = "ask questions to CSUDHBot"
            dispatcher.utter_message(" You selected to {} was your option.
".format(c_two))

            return[SlotSet("choice_two", c_two)]
##========================================================================#
##========================================================================#
##========================================================================#
##========================================================================#
##========================================================================#
```

File: Questions.py

```
# Questions that student may ask
api_key = "KEY"
from typing import Dict, Text, Any, List, Union, Optional

from rasa_sdk import Action, Tracker
from rasa_sdk.executor import CollectingDispatcher
from rasa_sdk.events import SlotSet
import requests
import json
import re
import sqlalchemy
import webbrowser
import pandas as pd
import googlemaps
from dbConnect import getData

##============================================================================#
##============================================================================#
##============================================================================#
class Action_FAFSA_info(Action):

    def name(self) -> Text:
        return "action_FAFSA"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #school information slot
        fafsa_application = tracker.get_slot("FAFSA_APP")

        q = 'select f.fafsa_type from school as s INNER JOIN offers_fafsa as
oof ON s.school_id = oof.school_id INNER JOIN fafsa as f ON oof.fafsa_id =
f.fafsa_id'
        #pass the sql query to the getData method and store the results in
`data` variable.
        data = getData(q)

        #Print out the fafsa types
        details = "Types of Fafsa offered at CSUDH"
        set_one = str(data[0][0]) + "                                    " +
str(data[1][0]) + "                                " + str(data[2][0])
        #combine the message
        link = '<a href="https://www.csudh.edu/financial-aid/types-
aid/">FAFSA types</a>'
        set_combined = details + "\n" +  set_one + "\n" + link
        # output message
        fafsa_appp = set_combined
        #uttering message containing the school response from slot
        dispatcher.utter_message(text="{}".format(fafsa_appp))
        #returning the slot for fafsa
        return [SlotSet("fafsa_appp", fafsa_application)]

##============================================================================#
##============================================================================#
```

```
##=========================================================================#
class Action_Enrollment(Action):

    def name(self) -> Text:
        return "action_Eenrollment"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #school information slot
        unii_enrollment = tracker.get_slot("uni_enroll")
        q = 'select e.semester from school as sc FULL OUTER JOIN attends as a
ON a.school_id = sc.school_id FULL OUTER JOIN user_type as ut ON a.id = ut.id
FULL OUTER JOIN views_enrollment as ve ON ut.id = ve.id FULL OUTER JOIN
enrollment as e ON ve.enrollment_id = e.enrollment_id WHERE ut.id = 1 AND
e.enrollment_id = 5 OR ut.id = 2 AND e.enrollment_id = 6 '
        #pass the sql query to the getData method and store the results in
`data` variable.
        data = getData(q)

        # list out the semester
        message = "Semesters available to view:"

        opt = "============================"
        set_one = str(data[0][0]) + "<br/><br/>" + str(data[1][0])
        opt2 = "============================"

        set_combined  = message + "\n" + opt + "\n" + set_one + "\n" + opt2

        university_enroll = set_combined
        #uttering message containing the school response from slot
        dispatcher.utter_message(text="{}".format(university_enroll))

        #returning the slot for school information meant for incoming
graduate students
        return [SlotSet("university_enroll", unii_enrollment)]
##=========================================================================#
##=========================================================================#
##=========================================================================#
##======================CAMPUS ===========================================##
class Action_Campus(Action):

    def name(self) -> Text:
        return "action_campus_info"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #school information slot
        campus_info = tracker.get_slot("campus_CSUDH")

        campus_information = ""
        #uttering message containing the school response from slot
        dispatcher.utter_message(text="".format(campus_information))

        #returning the slot for school information meant for incoming
graduate students
```

```
        return [SlotSet("campus_information", campus_info)]

class Action_Campus_Tour(Action):

    def name(self) -> Text:
        return "action_oncampus_tour"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #school information slot
        campus_tour_info = tracker.get_slot("campus_tour_CSUDH")
        q = 'select ct.campus_desc from school as s FULL OUTER JOIN
shows_campus as sc ON sc.school_id = s.school_id FULL OUTER JOIN campus as c
ON sc.campus_id = c.campus_id FULL OUTER JOIN provides_tours as pt ON
pt.campus_id = c.campus_id FULL OUTER JOIN campus_tours as ct ON
ct.campus_tour_id = pt.campus_tour_id'
        #pass the sql query to the getData method and store the results in
`data` variable.
        data = getData(q)
        message = "Prospective Student Tours"

        details = str(data[0][0])


        set_one = str(data[1][0])  + " <br/><br/> " +  str(data[2][0])  + "
<br/><br/> " +  str(data[3][0])  + " <br/><br/> " +  str(data[4][0])
        set_combined = message + "<br/><br/>" + details + "<br/><br/>" +
set_one + "\n\n"


        message2 = "Group Tours" + "<br/><br/>"

        set_two = str(data[5][0]) + "<br/><br/>" + str(data[6][0])

        link = '<a href = "https://www.csudh.edu/future-students/more-
info/visit-csudh/schedule-a-tour/">More Tour Information</a>'

        set_combined2 = message2 + set_two

        combined = set_combined + set_combined2 + "\n\n" + link
        on_campus_tour_info = combined
        #uttering message containing the school response from slot
        dispatcher.utter_message(text="{}".format(on_campus_tour_info))

        #returning the slot for school information meant for incoming
graduate students
        return [SlotSet("on_campus_tour_info", campus_tour_info)]

class Action_Campus_Map(Action):

    def name(self) -> Text:
        return "action_campusmap"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
```

```
        #school information slot
        campus_mapCSUDH = tracker.get_slot("campus_map_CSUDH")
        # api_key = "KEY"
        # url =  "https://maps.googleapis.com/maps/api/staticmap?"
        # location = input("Enter location: ")
        # center = location
        # zoom = 15
        # r = requests.get(url + "center" + center + "&zoom= " + str(zoom) +
"&size =400x400&key=" +  api_key)
        # print(url + "center=" + center + "&zoom=" + str(zoom) +
"&size=400x400&key=" + api_key)
        # f = open('/home/blueberrykary/Desktop/b.png', 'wb')
        # f.write(r.content)
        # f.close()

        #map that was received from google static maps
        ##Geocoding map coordinates
        # df = pd.DataFrame({
        #     'address':['CSUDH']
        #     })
        #
        # gmaps_key = googlemaps.Client(key = "KEY")
        #
        # df['Lat'] = None
        # df['Lon'] = None
        #
        # for i in range(len(df)):
        #     geocode_result = gmaps_key.geocode(df.loc[i, 'address'])
        #     try:
        #         lat = geocode_result[0]["geometry"]["location"]["lat"]
        #         lng = geocode_result[0]["geometry"]["location"]["lng"]
        #         df.loc[i, 'Lat'] = lat
        #         df.loc[i, 'Lon'] = lng
        #     except:
        #         lat = None
        #         lng = None


        link = '<iframe width="180" height="180" frameborder="0"
style="border:0"
src="https://www.google.com/maps/embed/v1/place?key=KEY&q=California+State+Un
iversity+Dominguez+Hills,Carson+CA" allowfullscreen></iframe>'

        campusMapCSUDH = link
        #uttering message containing the response from slot
        dispatcher.utter_message(text="{}".format(campusMapCSUDH))

        #returning the slot for meant for incoming graduate students
        return [SlotSet("campusMapCSUDH", campus_mapCSUDH)]


class Action_Campus_Employment(Action):

    def name(self) -> Text:
        return "action_employment"

    def run(self, dispatcher: CollectingDispatcher,
```

```
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #school information slot
        campus_employment = tracker.get_slot("campus_employment_CSUDH")
        q = 'select campus_e.employ_description from school as s FULL OUTER
JOIN shows_campus as sc ON sc.school_id = s.school_id FULL OUTER JOIN campus
as c ON sc.campus_id = c.campus_id FULL OUTER JOIN contains_employment as ce
ON c.campus_id = ce.campus_id FULL OUTER JOIN campus_employment as campus_e
ON ce.campus_employ_id = campus_e.campus_employ_id ORDER BY
campus_e.campus_employ_id LIMIT 5'
        #pass the sql query to the getData method and store the results in
`data` variable.
        data = getData(q)
        details = str(data[0][0])
        seperator = "================================================"
        set_one = str(data[1][0])  + " <br/> " +  str(data[2][0])  + " <br/>
" +  str(data[3][0])  + " <br/> " +  str(data[4][0])
        set_combined = details + "\n" + seperator + "\n" +  set_one + "\n\n"
        #-------------------------------------------------------------------#
        #query2
        q2 = 'select campus_e.employ_description from school as s FULL OUTER
JOIN shows_campus as sc ON sc.school_id = s.school_id FULL OUTER JOIN campus
as c ON sc.campus_id = c.campus_id FULL OUTER JOIN contains_employment as ce
ON c.campus_id = ce.campus_id FULL OUTER JOIN campus_employment as campus_e
ON ce.campus_employ_id = campus_e.campus_employ_id ORDER BY
campus_e.campus_employ_id LIMIT 5 OFFSET 5'
        #pass the sql query to the getData method and store the results in
`data` variable.
        data2 = getData(q2)
        details2 = str(data2[0][0])
        seperator2 = "================================================"
        set_one2 = str(data2[1][0])  + " <br/> " +  str(data2[2][0])  + "
<br/> " +  str(data2[3][0])  + " <br/> " +  str(data2[4][0])
        set_combined2 = details2 + "\n" + seperator2 + "\n" +  set_one2 +
"\n\n"
        #-------------------------------------------------------------------#
        # query3
        q3 = 'select campus_e.employ_description from school as s FULL OUTER
JOIN shows_campus as sc ON sc.school_id = s.school_id FULL OUTER JOIN campus
as c ON sc.campus_id = c.campus_id FULL OUTER JOIN contains_employment as ce
ON c.campus_id = ce.campus_id FULL OUTER JOIN campus_employment as campus_e
ON ce.campus_employ_id = campus_e.campus_employ_id ORDER BY
campus_e.campus_employ_id LIMIT 1 OFFSET 10'
        #pass the sql query to the getData method and store the results in
`data` variable.
        data3 = getData(q3)
        details3 = str(data3[0][0])
        set_combined3 = details3 + "\n\n"
        #-------------------------------------------------------------------#
        # query4
        q4 = 'select campus_e.employ_description from school as s FULL OUTER
JOIN shows_campus as sc ON sc.school_id = s.school_id FULL OUTER JOIN campus
as c ON sc.campus_id = c.campus_id FULL OUTER JOIN contains_employment as ce
ON c.campus_id = ce.campus_id FULL OUTER JOIN campus_employment as campus_e
ON ce.campus_employ_id = campus_e.campus_employ_id ORDER BY
campus_e.campus_employ_id LIMIT 5 OFFSET 11'
```

```
        #pass the sql query to the getData method and store the results in
`data` variable.
        data4 = getData(q4)
        #Print out the course details for 595
        details4 = str(data4[0][0])
        seperator4 = "================================================="
        set_one4 = str(data4[1][0])  + " <br/> " +  str(data4[2][0])  + "
<br/> " +  str(data4[3][0])  + " <br/> " +  str(data4[4][0])
        set_combined4 = details4 + "\n" + seperator4 + "\n" +  set_one4 +
"\n\n"
        #-------------------------------------------------------------------#
        #query5
        q5 = 'select campus_e.employ_description from school as s FULL OUTER
JOIN shows_campus as sc ON sc.school_id = s.school_id FULL OUTER JOIN campus
as c ON sc.campus_id = c.campus_id FULL OUTER JOIN contains_employment as ce
ON c.campus_id = ce.campus_id FULL OUTER JOIN campus_employment as campus_e
ON ce.campus_employ_id = campus_e.campus_employ_id ORDER BY
campus_e.campus_employ_id LIMIT 6 OFFSET 16'
        #pass the sql query to the getData method and store the results in
`data` variable.
        data5 = getData(q5)
        details5 = str(data5[0][0])
        seperator5 = "================================================="
        set_one5 = str(data5[1][0])  + " <br/> " +  str(data5[2][0])  + "
<br/> " +  str(data5[3][0])  + " <br/> " +  str(data5[4][0]) + " <br/> " +
str(data5[5][0])
        set_combined5 = details5 + "\n" + seperator5 + "\n" +  set_one5 +
"\n\n"
        #-------------------------------------------------------------------#
        message4 = '<a href="https://www.lsucsudh.org/student-
employment/">Employment Opportunities</a>'
        output = set_combined + "\n\n" + message4
        campus_employment_stu = output

        #uttering message containing the school response from slot
        dispatcher.utter_message(text="{}".format(campus_employment_stu))

        #returning the slot for school information meant for incoming
graduate students
        return [SlotSet("campus_employment_stu", campus_employment)]

##=====================================================================#
##====================================================================#
##=====================================================================#
class Action_view_course(Action):

    def name(self) -> Text:
        return "action_view_cs_course"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #school information slot
        course_start = tracker.get_slot("getting_course")

        course_end = ""
        #uttering message containing the school response from slot
```

```
        dispatcher.utter_message(text="".format(course_end))

        #returning the slot for school information meant for incoming
graduate students
        return [SlotSet("course_end", course_start)]

##=========================================================================#
##=========================================================================#
##=========================================================================#
class Action_view_spring2020_course(Action):

    def name(self) -> Text:
        return "action_view_spring2020"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #school information slot
        spring_c1 = tracker.get_slot("spring_2020_course1")

        #select the semester, the subject type, the course number, class
number, and class time for CSC 590
        q = 'select  e.semester, s.subject_type, c.course_number,
cl.class_number, t.day_time, t.start_time, t.end_time, p.fname, p.lname,
p.email from school as sc FULL OUTER JOIN attends as a ON a.school_id =
sc.school_id FULL OUTER JOIN user_type as ut ON a.id = ut.id FULL OUTER JOIN
views_enrollment as ve ON ut.id = ve.id FULL OUTER JOIN enrollment as e ON
ve.enrollment_id = e.enrollment_id FULL OUTER JOIN has_course as hc ON
e.enrollment_id = hc.enrollment_id FULL OUTER JOIN course as c ON
hc.course_id = c.course_id FULL OUTER JOIN contains_subject as csub ON
c.course_id = csub.course_id FULL OUTER JOIN subject as s ON csub.subject_id
= s.subject_id FULL OUTER JOIN offers_class as oc ON s.subject_id =
oc.subject_id FULL OUTER JOIN class as cl ON oc.class_id = cl.class_id FULL
OUTER JOIN meets as m ON cl.class_id = m.class_id FULL OUTER JOIN class_time
as t ON m.class_time_id = t.class_time_id FULL OUTER JOIN teaches as tt ON
cl.class_id = tt.class_id FULL OUTER JOIN professor as p ON tt.professor_id =
p.professor_id WHERE ut.id = 1 AND e.enrollment_id = 5 AND t.class_time_id =
16 AND c.course_id = 21'
        #pass the sql query to the getData method and store the results in
`data` variable.
        data = getData(q)


        #Print out the course details for 595
        opt = "============================="
        details = "Course Details"
        set_one = str(data[0][0])
        set_two = str(data[0][1])
        set_three = str(data[0][2])
        set_four = "    Class Number = " + str(data[0][3])
        set_five = "On " + str(data[0][4]) + " at " + str(data[0][5]) + " to
" + str(data[0][6])
        set_six = "    Professor: " + str(data[0][7]) + " " + str(data[0][8])
+ "\n" + "   Email: " + str(data[0][9])
        opt2 = "============================="
```

```
        set_combined  = opt + "<br/><br/>" + details + "<br/><br/>"+ set_one
+ "<br/><br/>" + set_two + "<br/><br/>" + set_three
        set_combined2 = " " + set_four + "<br/><br/>" + set_five +
"<br/><br/>" + set_six + "<br/><br/>" + opt2

        set_com = set_combined + set_combined2

        spring_2020c1 = set_com

        #uttering message containing the school response from slot
        dispatcher.utter_message(text="{}".format(spring_2020c1))

        #returning the slot for school information meant for incoming
graduate students
        return [SlotSet("spring_2020c1", spring_c1)]

##=========================================================================#
##=========================================================================#
##=========================================================================#
class Action_view_spring2020_course2(Action):

    def name(self) -> Text:
        return "action_view_spring20202"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #school information slot
        spring_c2 = tracker.get_slot("spring_2020_course2")

        #select the semester, the subject type, the course number, class
number, and class time for CSC 590
        q = 'select  e.semester, s.subject_type, c.course_number,
cl.class_number, t.day_time, t.start_time, t.end_time, p.fname, p.lname,
p.email from school as sc FULL OUTER JOIN attends as a ON a.school_id =
sc.school_id FULL OUTER JOIN user_type as ut ON a.id = ut.id FULL OUTER JOIN
views_enrollment as ve ON ut.id = ve.id FULL OUTER JOIN enrollment as e ON
ve.enrollment_id = e.enrollment_id FULL OUTER JOIN has_course as hc ON
e.enrollment_id = hc.enrollment_id FULL OUTER JOIN course as c ON
hc.course_id = c.course_id FULL OUTER JOIN contains_subject as csub ON
c.course_id = csub.course_id FULL OUTER JOIN subject as s ON csub.subject_id
= s.subject_id FULL OUTER JOIN offers_class as oc ON s.subject_id =
oc.subject_id FULL OUTER JOIN class as cl ON oc.class_id = cl.class_id FULL
OUTER JOIN meets as m ON cl.class_id = m.class_id FULL OUTER JOIN class_time
as t ON m.class_time_id = t.class_time_id FULL OUTER JOIN teaches as tt ON
cl.class_id = tt.class_id FULL OUTER JOIN professor as p ON tt.professor_id =
p.professor_id WHERE ut.id = 1 AND e.enrollment_id = 5 AND t.class_time_id =
11 AND c.course_id = 20 AND p.professor_id = 12'
        #pass the sql query to the getData method and store the results in
`data` variable.
        data = getData(q)


        #Print out the course details for 595
        opt = "============================="
        details = "     Course Details       "
        set_one = "      " + str(data[0][0]) + "      "
```

```
        set_two = "      " + str(data[0][1]) + "      "
        set_three = "      " + str(data[0][2]) + "      "
        set_four = "      " + "Class Number = " + str(data[0][3]) + "      "
        set_five = "      " + "On " + str(data[0][4]) + " at " +
str(data[0][5]) + " to " + str(data[0][6]) + "      "
        set_six = "      " + "    Professor: " + str(data[0][7]) + " " +
str(data[0][8]) + "\n" + "   Email: " + str(data[0][9]) + "      "
        opt2 = "=============================="

        set_combined  = opt + "<br/><br/>" + details + "<br/><br/>" +
set_one + "\n" + "<br/><br/>" + set_two + "<br/><br/>" + set_three
        set_combined2 = " " + set_four + "<br/><br/>" + set_five +
"<br/><br/>" + set_six + "<br/><br/>" + opt2

        set_com = set_combined + set_combined2

        spring_2020c2 = set_com

        #uttering message containing the school response from slot
        dispatcher.utter_message(text="{}".format(spring_2020c2))

        #returning the slot for school information meant for incoming
graduate students
        return [SlotSet("spring_2020c2", spring_c2)]

##=========================================================================#
##=========================================================================#
##=========================================================================#
class Action_view_spring2020_course3(Action):

    def name(self) -> Text:
        return "action_view_spring20203"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #school information slot
        spring_c3 = tracker.get_slot("spring_2020_course3")

        #select the semester, the subject type, the course number, class
number, and class time for CSC 590
        q = 'select  e.semester, s.subject_type, c.course_number,
cl.class_number, t.day_time, t.start_time, t.end_time, p.fname, p.lname,
p.email from school as sc FULL OUTER JOIN attends as a ON a.school_id =
sc.school_id FULL OUTER JOIN user_type as ut ON a.id = ut.id FULL OUTER JOIN
views_enrollment as ve ON ut.id = ve.id FULL OUTER JOIN enrollment as e ON
ve.enrollment_id = e.enrollment_id FULL OUTER JOIN has_course as hc ON
e.enrollment_id = hc.enrollment_id FULL OUTER JOIN course as c ON
hc.course_id = c.course_id FULL OUTER JOIN contains_subject as csub ON
c.course_id = csub.course_id FULL OUTER JOIN subject as s ON csub.subject_id
= s.subject_id FULL OUTER JOIN offers_class as oc ON s.subject_id =
oc.subject_id FULL OUTER JOIN class as cl ON oc.class_id = cl.class_id FULL
OUTER JOIN meets as m ON cl.class_id = m.class_id FULL OUTER JOIN class_time
as t ON m.class_time_id = t.class_time_id FULL OUTER JOIN teaches as tt ON
cl.class_id = tt.class_id FULL OUTER JOIN professor as p ON tt.professor_id =
p.professor_id WHERE ut.id = 1 AND e.enrollment_id = 5 AND c.course_id = 18
AND p.professor_id = 16'
```

```
        #pass the sql query to the getData method and store the results in
`data` variable.
        data = getData(q)


        #Print out the course details for 595
        opt = "============================="
        details = "Course Details"
        set_one = str(data[0][0])
        set_two = str(data[0][1])
        set_three = str(data[0][2])
        set_four = "    Class Number = " + str(data[0][3])
        set_five = "On " + str(data[0][4]) + " at " + str(data[0][5]) + " to
" + str(data[0][6])
        set_six = "Professor: " + str(data[0][7]) + " " + str(data[0][8]) +
"\n" + " Email: " + str(data[0][9])
        opt2 = "============================="

        set_combined  = opt + "<br/><br/>" + details + "<br/><br/>" + set_one
+ "<br/><br/>" + set_two + "<br/><br/>" + set_three
        set_combined2 = " " + set_four + "<br/>" + set_five + "<br/>" +
set_six + "<br/><br/>" + opt2

        set_com = set_combined + set_combined2

        spring_2020c3 = set_com

        #uttering message containing the school response from slot
        dispatcher.utter_message(text="{}".format(spring_2020c3))

        #returning the slot for school information meant for incoming
graduate students
        return [SlotSet("spring_2020c3", spring_c3)]

##=========================================================================#
##=========================================================================#
##=========================================================================#
class Action_view_summer2020_course1(Action):

    def name(self) -> Text:
        return "action_view_summer2020"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #school information slot
        summer_c1 = tracker.get_slot("summer_2020_course1")

        q = 'select  e.semester, s.subject_type, c.course_number,
cl.class_number, t.day_time, t.start_time, t.end_time, p.fname, p.lname,
p.email from school as sc FULL OUTER JOIN attends as a ON a.school_id =
sc.school_id FULL OUTER JOIN user_type as ut ON a.id = ut.id FULL OUTER JOIN
views_enrollment as ve ON ut.id = ve.id FULL OUTER JOIN enrollment as e ON
ve.enrollment_id = e.enrollment_id FULL OUTER JOIN has_course as hc ON
e.enrollment_id = hc.enrollment_id FULL OUTER JOIN course as c ON
hc.course_id = c.course_id FULL OUTER JOIN contains_subject as csub ON
c.course_id = csub.course_id FULL OUTER JOIN subject as s ON csub.subject_id
```

```
= s.subject_id FULL OUTER JOIN offers_class as oc ON s.subject_id =
oc.subject_id FULL OUTER JOIN class as cl ON oc.class_id = cl.class_id FULL
OUTER JOIN meets as m ON cl.class_id = m.class_id FULL OUTER JOIN class_time
as t ON m.class_time_id = t.class_time_id FULL OUTER JOIN teaches as tt ON
cl.class_id = tt.class_id FULL OUTER JOIN professor as p ON tt.professor_id =
p.professor_id WHERE ut.id = 1 AND e.enrollment_id = 6 AND t.class_time_id =
17 AND c.course_id = 24 AND cl.class_number = 31146'
        #pass the sql query to the getData method and store the results in
`data` variable.
        data = getData(q)


        #Print out the course details for 595
        opt = "============================="
        details = "Course Details"
        set_one = str(data[0][0])
        set_two = str(data[0][1])
        set_three = str(data[0][2])
        set_four = "    Class Number = " + str(data[0][3])
        set_five = "On " + str(data[0][4]) + " at " + str(data[0][5]) + " to
" + str(data[0][6])
        set_six = "    Professor: " + str(data[0][7]) + " " + str(data[0][8])
+ "\n" + "   Email: " + str(data[0][9])
        opt2 = "============================="

        set_combined  = opt + "<br/><br/>" + details + "<br/><br/>" +
set_one + "<br/><br/>" + set_two + "<br/><br/>" + set_three
        set_combined2 = " " + set_four + "<br/><br/>" + set_five +
"<br/><br/>" + set_six + "<br/><br/>" + opt2

        set_com = set_combined + set_combined2

        summer_2020c1 = set_com

        #uttering message containing the school response from slot
        dispatcher.utter_message(text="{}".format(summer_2020c1))

        #returning the slot for school information meant for incoming
graduate students
        return [SlotSet("summer_2020c1", summer_c1)]

##=========================================================================#
##=========================================================================#
##=========================================================================#
class Action_view_summer2020_course2(Action):

    def name(self) -> Text:
        return "action_view_summer20202"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #school information slot
        summer_c2 = tracker.get_slot("summer_2020_course2")

        q = 'select  e.semester, s.subject_type, c.course_number,
cl.class_number, t.day_time, t.start_time, t.end_time, p.fname, p.lname,
```

```
p.email from school as sc FULL OUTER JOIN attends as a ON a.school_id =
sc.school_id FULL OUTER JOIN user_type as ut ON a.id = ut.id FULL OUTER JOIN
views_enrollment as ve ON ut.id = ve.id FULL OUTER JOIN enrollment as e ON
ve.enrollment_id = e.enrollment_id FULL OUTER JOIN has_course as hc ON
e.enrollment_id = hc.enrollment_id FULL OUTER JOIN course as c ON
hc.course_id = c.course_id FULL OUTER JOIN contains_subject as csub ON
c.course_id = csub.course_id FULL OUTER JOIN subject as s ON csub.subject_id
= s.subject_id FULL OUTER JOIN offers_class as oc ON s.subject_id =
oc.subject_id FULL OUTER JOIN class as cl ON oc.class_id = cl.class_id FULL
OUTER JOIN meets as m ON cl.class_id = m.class_id FULL OUTER JOIN class_time
as t ON m.class_time_id = t.class_time_id FULL OUTER JOIN teaches as tt ON
cl.class_id = tt.class_id FULL OUTER JOIN professor as p ON tt.professor_id =
p.professor_id WHERE ut.id = 1 AND e.enrollment_id = 6 AND t.class_time_id =
17 AND c.course_id = 25 AND cl.class_number = 31147'

        #pass the sql query to the getData method and store the results in
`data` variable.
        data = getData(q)

        #Print out the course details for 595
        opt = "============================="
        details = "Course Details"
        set_one = str(data[0][0])
        set_two = str(data[0][1])
        set_three = str(data[0][2])
        set_four = "    Class Number = " + str(data[0][3])
        set_five = "On " + str(data[0][4]) + " at " + str(data[0][5]) + " to
" + str(data[0][6])
        set_six = "    Professor: " + str(data[0][7]) + " " + str(data[0][8])
+ "\n" + "   Email: " + str(data[0][9])
        opt2 = "============================="

        set_combined  = opt + "<br/><br/>" + details + "<br/><br/>" + set_one
+ "<br/><br/>" + set_two + "<br/><br/>" + set_three
        set_combined2 = " " + set_four + "<br/><br/>" + set_five +
"<br/><br/>" + set_six + "<br/><br/>" + opt2

        set_com = set_combined + set_combined2

        summer_2020c2 = set_com

        #uttering message containing the school response from slot
        dispatcher.utter_message(text="{}".format(summer_2020c2))

        #returning the slot for school information meant for incoming
graduate students
        return [SlotSet("summer_2020c2", summer_c2)]
```

File: SearchStudentType.py

```
## Program to Search for Student, and redirects user to different branches
## within the stories.

from typing import Any, Text, Dict, List
```

```
from rasa_sdk import Action, Tracker
from rasa_sdk.executor import CollectingDispatcher
from rasa_sdk.events import SlotSet
import requests
import json
import re

##=========================================================================#
##=========================================================================#
##=========================================================================#
##=========================================================================#
##====================Selecting from incoming student====================##
class Action_Search_IncomingStudent(Action):

    def name(self) -> Text:
        return "action_searchincomingstudent"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #istudent: incoming student slot
        istudent = tracker.get_slot("student_incoming")
        #iinfo: incoming student information
        iinfo = "incoming student"
        #uttering message when user inputs the type of student they are
        #in this case they are an incoming student
        dispatcher.utter_message("You are an {}".format(iinfo))
        #returning the slot for incoming student information
        return [SlotSet("iinfo", istudent)]
##=========================================================================#
##=========================================================================#
##=========================================================================#
##=========================================================================#
##==================Select the incoming program:undergraduate============##
class Action_ProgramSearchUndergraduate(Action):

    def name(self) -> Text:
        return "action_search_undergraduateprogram"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #undergraduate program slot
        uprog = tracker.get_slot("program_undergrad")
        #the response from uprograms
        uprograms = "undergraduate program"
        #uttering message containing uprograms response from slot
        dispatcher.utter_message(text = "The following contains information
over the {} program.".format(uprograms))
        #returning the slot for programs meant for incoming undergraduate
        return [SlotSet("uprograms", uprog)]
##=========================================================================#
##=========================================================================#
##=========================================================================#
##=========================================================================#
##==============Select the incoming program: graduate ====================##
class Action_ProgramSearch(Action):
```

```
    def name(self) -> Text:
        return "action_search_graduateprogram"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #graduate program slot
        gprog = tracker.get_slot("program_graduate")
        #the response from gprograms
        gprograms = "undergraduate program"
        #uttering message containing gprograms response from slot
        dispatcher.utter_message("The following contains information over the
{} program.".format(gprograms))
        #returning the slot for programs meant for incoming graduate students
        return [SlotSet("gprograms", gprog)]
##=========================================================================#
##=========================================================================#
##=========================================================================#
##=========================================================================#
##====================Selecting from recurring student====================##
class Action_Search_RecurrStudent(Action):

    def name(self) -> Text:
        return "action_searchrecurringstudent"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        #rstudent: recurring student slot
        rstudent = tracker.get_slot("student_recurr")
        # rinfo: recurring student information
        rinfo = ""
        #uttering message when user inputs the type of student they are
        #in this case they are a recurring student
        dispatcher.utter_message("".format(rinfo))
        #returning the slot for recurring information
        return [SlotSet("rinfo", rstudent)]
##=========================================================================#
##=========================================================================#
##=========================================================================#
##=========================================================================#
##=========================================================================#
```

File: Conversation_Small_Talk.md

```
# story bored
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* bored
```

```
  - utter_bored
  - utter_thank_you_uprogram
  - utter_goodbye
  - action_restart

## story sad
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* sad
  - utter_sad
  - utter_thank_you_uprogram
  - utter_goodbye
  - action_restart

## story lazy
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* lazy
  - utter_lazy
  - utter_thank_you_uprogram
  - utter_goodbye
  - action_restart
```

File: CurrentStudent_V1_1.md

```
## Story to view financial aid, enrollment, and computer science 585 then
ending the program
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_two{"student_recurr":"ongoing student", "student_recurr":
"freshman", "student_recurr":"sophmore", "student_recurr":"junior",
"student_recurr":"senior", "student_recurr": "master's student",
"student_recurr": "first year  student", "student_recurr":"second year
student", "student_recurr" : "third year student", "student_recurr": "fourth
year student", "student_recurr": "fifth year student",
"student_recurr":"graduate student", "student_recurr": "undergraduate
student", "student_recurr": "international student",
"student_recurr":"recurring student", "student_recurr":"continuing student",
"student_recurr": "recurring undergraduate student", "student_recurr":
```

```
"recurring graduate student", "student_recurr": "ongoing undergraduate
student", "student_recurr":"ongoing graduate student"}
  - action_searchrecurringstudent
  - slot{"rinfo":"rstudent"}
  - utter_current_student
* good_thanks
  - utter_great
* yes_ido
  - utter_ask1
* financial_aid_application{"FAFSA_APP":"FAFSA", "FAFSA_APP":"financial aid"}
  - action_FAFSA
  - slot{"fafsa_appp":"fafsa_application"}
  - utter_financial_aid
* no_more_ques
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart


<!--====================================================-->
<!--====================================================-->

## Story to view financial aid, enrollment, and computer science 590 then
ending the program
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_two{"student_recurr":"ongoing student", "student_recurr":
"freshman", "student_recurr":"sophmore", "student_recurr":"junior",
"student_recurr":"senior", "student_recurr": "master's student",
"student_recurr": "first year  student", "student_recurr":"second year
student", "student_recurr" : "third year student", "student_recurr": "fourth
year student", "student_recurr": "fifth year student",
"student_recurr":"graduate student", "student_recurr": "undergraduate
student", "student_recurr": "international student",
"student_recurr":"recurring student", "student_recurr":"continuing student",
"student_recurr": "recurring undergraduate student", "student_recurr":
"recurring graduate student", "student_recurr": "ongoing undergraduate
student", "student_recurr":"ongoing graduate student"}
  - action_searchrecurringstudent
  - slot{"rinfo":"rstudent"}
  - utter_current_student
* good_thanks
  - utter_great
* yes_ido
  - utter_ask1
* financial_aid_application{"FAFSA_APP":"FAFSA", "FAFSA_APP":"financial aid"}
  - action_FAFSA
  - slot{"fafsa_appp":"fafsa_application"}
  - utter_financial_aid
* ask_a_question
  - utter_ask2
```

```
* university_enrollment{"uni_enroll":"enrollment"}
  - action_Eenrollment
  - slot{"university_enroll":"unii_enrollment"}
  - utter_enrollment
* no_more_ques
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart
```

File: CurrentStudent_V1_Spring.md

```
## Story to view financial aid, enrollment, and computer science 585 then
ending the program
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_two{"student_recurr":"ongoing student", "student_recurr":
"freshman", "student_recurr":"sophmore", "student_recurr":"junior",
"student_recurr":"senior", "student_recurr": "master's student",
"student_recurr": "first year  student", "student_recurr":"second year
student", "student_recurr" : "third year student", "student_recurr": "fourth
year student", "student_recurr": "fifth year student",
"student_recurr":"graduate student", "student_recurr": "undergraduate
student", "student_recurr": "international student",
"student_recurr":"recurring student", "student_recurr":"continuing student",
"student_recurr": "recurring undergraduate student", "student_recurr":
"recurring graduate student", "student_recurr": "ongoing undergraduate
student", "student_recurr":"ongoing graduate student"}
  - action_searchrecurringstudent
  - slot{"rinfo":"rstudent"}
  - utter_current_student
* good_thanks
  - utter_great
* yes_ido
  - utter_ask1
* financial_aid_application{"FAFSA_APP":"FAFSA", "FAFSA_APP":"financial aid"}
  - action_FAFSA
  - slot{"fafsa_appp":"fafsa_application"}
  - utter_financial_aid
* ask_a_question
  - utter_ask2
* university_enrollment{"uni_enroll":"enrollment"}
  - action_Eenrollment
  - slot{"university_enroll":"unii_enrollment"}
  - utter_enrollment
* ask_a_question
  - utter_ask3
* course_description{"getting_course": "computer science",
"getting_course":"comp sci", "getting_course": "comp science"}
  - action_view_cs_course
  - slot{"course_end":"course_start"}
```

```
  - utter_computerscience_option
* Spring2020
  - utter_classes_spring_2020
* courses_spring20201{"spring_2020_course2":"csc 585"}
  - action_view_spring20202
  - slot{"spring_2020c2":"spring_c2"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart


<!--====================================================-->
<!--====================================================-->

## Story to view financial aid, enrollment, and computer science 590 then
ending the program
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_two{"student_recurr":"ongoing student", "student_recurr":
"freshman", "student_recurr":"sophmore", "student_recurr":"junior",
"student_recurr":"senior", "student_recurr": "master's student",
"student_recurr": "first year  student", "student_recurr":"second year
student", "student_recurr" : "third year student", "student_recurr": "fourth
year student", "student_recurr": "fifth year student",
"student_recurr":"graduate student", "student_recurr": "undergraduate
student", "student_recurr": "international student",
"student_recurr":"recurring student", "student_recurr":"continuing student",
"student_recurr": "recurring undergraduate student", "student_recurr":
"recurring graduate student", "student_recurr": "ongoing undergraduate
student", "student_recurr":"ongoing graduate student"}
  - action_searchrecurringstudent
  - slot{"rinfo":"rstudent"}
  - utter_current_student
* good_thanks
  - utter_great
* yes_ido
  - utter_ask1
* financial_aid_application{"FAFSA_APP":"FAFSA", "FAFSA_APP":"financial aid"}
  - action_FAFSA
  - slot{"fafsa_appp":"fafsa_application"}
  - utter_financial_aid
* ask_a_question
  - utter_ask2
* university_enrollment{"uni_enroll":"enrollment"}
  - action_Eenrollment
  - slot{"university_enroll":"unii_enrollment"}
  - utter_enrollment
* ask_a_question
  - utter_ask3
* course_description{"getting_course": "computer science",
"getting_course":"comp sci", "getting_course": "comp science"}
```

```
  - action_view_cs_course
  - slot{"course_end":"course_start"}
  - utter_computerscience_option
* Spring2020
  - utter_classes_spring_2020
* courses_spring2020{"spring_2020_course1":"csc 590"}
  - action_view_spring2020
  - slot{"spring_2020c1":"spring_c1"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart


<!--====================================================-->
<!--====================================================-->

## Story to view financial aid, enrollment, and computer science 582 then
ending the program
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_two{"student_recurr":"ongoing student", "student_recurr":
"freshman", "student_recurr":"sophmore", "student_recurr":"junior",
"student_recurr":"senior", "student_recurr": "master's student",
"student_recurr": "first year  student", "student_recurr":"second year
student", "student_recurr" : "third year student", "student_recurr": "fourth
year student", "student_recurr": "fifth year student",
"student_recurr":"graduate student", "student_recurr": "undergraduate
student", "student_recurr": "international student",
"student_recurr":"recurring student", "student_recurr":"continuing student",
"student_recurr": "recurring undergraduate student", "student_recurr":
"recurring graduate student", "student_recurr": "ongoing undergraduate
student", "student_recurr":"ongoing graduate student"}
  - action_searchrecurringstudent
  - slot{"rinfo":"rstudent"}
  - utter_current_student
* good_thanks
  - utter_great
* yes_ido
  - utter_ask1
* financial_aid_application{"FAFSA_APP":"FAFSA", "FAFSA_APP":"financial aid"}
  - action_FAFSA
  - slot{"fafsa_appp":"fafsa_application"}
  - utter_financial_aid
* ask_a_question
  - utter_ask2
* university_enrollment{"uni_enroll":"enrollment"}
  - action_Eenrollment
  - slot{"university_enroll":"unii_enrollment"}
  - utter_enrollment
* ask_a_question
  - utter_ask3
```

```
* course_description{"getting_course": "computer science",
"getting_course":"comp sci", "getting_course": "comp science"}
  - action_view_cs_course
  - slot{"course_end":"course_start"}
  - utter_computerscience_option
* Spring2020
  - utter_classes_spring_2020
* courses_spring20202{"spring_2020_course3":"csc 582"}
  - action_view_spring20203
  - slot{"spring_2020c3":"spring_c3"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart


<!--====================================================-->
<!--====================================================-->
```

File: CurrentStudent_V1_Summer.md

```
## Story to view financial aid, enrollment, and computer science 500 then
ending the program for Summer
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_two{"student_recurr":"ongoing student", "student_recurr":
"freshman", "student_recurr":"sophmore", "student_recurr":"junior",
"student_recurr":"senior", "student_recurr": "master's student",
"student_recurr": "first year  student", "student_recurr":"second year
student", "student_recurr" : "third year student", "student_recurr": "fourth
year student", "student_recurr": "fifth year student",
"student_recurr":"graduate student", "student_recurr": "undergraduate
student", "student_recurr": "international student",
"student_recurr":"recurring student", "student_recurr":"continuing student",
"student_recurr": "recurring undergraduate student", "student_recurr":
"recurring graduate student", "student_recurr": "ongoing undergraduate
student", "student_recurr":"ongoing graduate student"}
  - action_searchrecurringstudent
  - slot{"rinfo":"rstudent"}
  - utter_current_student
* good_thanks
  - utter_great
* yes_ido
  - utter_ask1
* financial_aid_application{"FAFSA_APP":"FAFSA", "FAFSA_APP":"financial aid"}
  - action_FAFSA
  - slot{"fafsa_appp":"fafsa_application"}
  - utter_financial_aid
* ask_a_question
  - utter_ask2
```

```
* university_enrollment{"uni_enroll":"enrollment"}
  - action_Eenrollment
  - slot{"university_enroll":"unii_enrollment"}
  - utter_enrollment
* ask_a_question
  - utter_ask3
* course_description{"getting_course": "computer science",
"getting_course":"comp sci", "getting_course": "comp science"}
  - action_view_cs_course
  - slot{"course_end":"course_start"}
  - utter_computerscience_option
* Summer2020
  - utter_classes_summer_20201
* courses_summer2020{"summer_2020_course1":"csc 500"}
  - action_view_summer2020
  - slot{"summer_2020c1":"summer_c1"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart


## Story to view financial aid, enrollment, and computer science 501 then
ending the program for Summer
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_two{"student_recurr":"ongoing student", "student_recurr":
"freshman", "student_recurr":"sophmore", "student_recurr":"junior",
"student_recurr":"senior", "student_recurr": "master's student",
"student_recurr": "first year  student", "student_recurr":"second year
student", "student_recurr" : "third year student", "student_recurr": "fourth
year student", "student_recurr": "fifth year student",
"student_recurr":"graduate student", "student_recurr": "undergraduate
student", "student_recurr": "international student",
"student_recurr":"recurring student", "student_recurr":"continuing student",
"student_recurr": "recurring undergraduate student", "student_recurr":
"recurring graduate student", "student_recurr": "ongoing undergraduate
student", "student_recurr":"ongoing graduate student"}
  - action_searchrecurringstudent
  - slot{"rinfo":"rstudent"}
  - utter_current_student
* good_thanks
  - utter_great
* yes_ido
  - utter_ask1
* financial_aid_application{"FAFSA_APP":"FAFSA", "FAFSA_APP":"financial aid"}
  - action_FAFSA
  - slot{"fafsa_appp":"fafsa_application"}
  - utter_financial_aid
* ask_a_question
  - utter_ask2
* university_enrollment{"uni_enroll":"enrollment"}
```

```
  - action_Eenrollment
  - slot{"university_enroll":"unii_enrollment"}
  - utter_enrollment
* ask_a_question
  - utter_ask3
* course_description{"getting_course": "computer science",
"getting_course":"comp sci", "getting_course": "comp science"}
  - action_view_cs_course
  - slot{"course_end":"course_start"}
  - utter_computerscience_option
* Summer2020
  - utter_classes_summer_20201
* courses_summer20201{"summer_2020_course2":"csc 501"}
  - action_view_summer20202
  - slot{"summer_2020c2":"summer_c2"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart
```

File: CurrentStudent_V2.md

```
## Story to view financial aid, enrollment, and computer science 585 then
ending the program mixed up
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_two{"student_recurr":"ongoing student", "student_recurr":
"freshman", "student_recurr":"sophmore", "student_recurr":"junior",
"student_recurr":"senior", "student_recurr": "master's student",
"student_recurr": "first year  student", "student_recurr":"second year
student", "student_recurr" : "third year student", "student_recurr": "fourth
year student", "student_recurr": "fifth year student",
"student_recurr":"graduate student", "student_recurr": "undergraduate
student", "student_recurr": "international student",
"student_recurr":"recurring student", "student_recurr":"continuing student",
"student_recurr": "recurring undergraduate student", "student_recurr":
"recurring graduate student", "student_recurr": "ongoing undergraduate
student", "student_recurr":"ongoing graduate student"}
  - action_searchrecurringstudent
  - slot{"rinfo":"rstudent"}
  - utter_current_student
* good_thanks
  - utter_great
* yes_ido
  - utter_ask1
* university_enrollment{"uni_enroll":"enrollment"}
  - action_Eenrollment
  - slot{"university_enroll":"unii_enrollment"}
  - utter_enrollment
* no_more_ques
```

```
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart


<!--====================================================-->
<!--====================================================-->

## Story to view financial aid, enrollment, and computer science 590 then
ending the program mixed up
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_two{"student_recurr":"ongoing student", "student_recurr":
"freshman", "student_recurr":"sophmore", "student_recurr":"junior",
"student_recurr":"senior", "student_recurr": "master's student",
"student_recurr": "first year  student", "student_recurr":"second year
student", "student_recurr" : "third year student", "student_recurr": "fourth
year student", "student_recurr": "fifth year student",
"student_recurr":"graduate student", "student_recurr": "undergraduate
student", "student_recurr": "international student",
"student_recurr":"recurring student", "student_recurr":"continuing student",
"student_recurr": "recurring undergraduate student", "student_recurr":
"recurring graduate student", "student_recurr": "ongoing undergraduate
student", "student_recurr":"ongoing graduate student"}
  - action_searchrecurringstudent
  - slot{"rinfo":"rstudent"}
  - utter_current_student
* good_thanks
  - utter_great
* yes_ido
  - utter_ask1
* university_enrollment{"uni_enroll":"enrollment"}
  - action_Eenrollment
  - slot{"university_enroll":"unii_enrollment"}
  - utter_enrollment
* ask_a_question
  - utter_ask2
* financial_aid_application{"FAFSA_APP":"FAFSA", "FAFSA_APP":"financial aid"}
  - action_FAFSA
  - slot{"fafsa_appp":"fafsa_application"}
  - utter_financial_aid
* no_more_ques
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart


<!--====================================================-->
<!--====================================================-->
```

File: CurrentStudent_V2_Spring.md

```
## Story to view financial aid, enrollment, and computer science 585 then
ending the program mixed up SPRING term
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_two{"student_recurr":"ongoing student", "student_recurr":
"freshman", "student_recurr":"sophmore", "student_recurr":"junior",
"student_recurr":"senior", "student_recurr": "master's student",
"student_recurr": "first year  student", "student_recurr":"second year
student", "student_recurr" : "third year student", "student_recurr": "fourth
year student", "student_recurr": "fifth year student",
"student_recurr":"graduate student", "student_recurr": "undergraduate
student", "student_recurr": "international student",
"student_recurr":"recurring student", "student_recurr":"continuing student",
"student_recurr": "recurring undergraduate student", "student_recurr":
"recurring graduate student", "student_recurr": "ongoing undergraduate
student", "student_recurr":"ongoing graduate student"}
  - action_searchrecurringstudent
  - slot{"rinfo":"rstudent"}
  - utter_current_student
* good_thanks
  - utter_great
* yes_ido
  - utter_ask1
* university_enrollment{"uni_enroll":"enrollment"}
  - action_Eenrollment
  - slot{"university_enroll":"unii_enrollment"}
  - utter_enrollment
* ask_a_question
  - utter_ask2
* financial_aid_application{"FAFSA_APP":"FAFSA", "FAFSA_APP":"financial aid"}
  - action_FAFSA
  - slot{"fafsa_appp":"fafsa_application"}
  - utter_financial_aid
* ask_a_question
  - utter_ask3
* course_description{"getting_course": "computer science",
"getting_course":"comp sci", "getting_course": "comp science"}
  - action_view_cs_course
  - slot{"course_end":"course_start"}
  - utter_computerscience_option
* Spring2020
  - utter_classes_spring_2020
* courses_spring20201{"spring_2020_course2":"csc 585"}
  - action_view_spring20202
  - slot{"spring_2020c2":"spring_c2"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart
```

```
<!--====================================================-->
<!--====================================================-->

## Story to view financial aid, enrollment, and computer science 590 then
ending the program mixed up SPRING term
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_two{"student_recurr":"ongoing student", "student_recurr":
"freshman", "student_recurr":"sophmore", "student_recurr":"junior",
"student_recurr":"senior", "student_recurr": "master's student",
"student_recurr": "first year  student", "student_recurr":"second year
student", "student_recurr" : "third year student", "student_recurr": "fourth
year student", "student_recurr": "fifth year student",
"student_recurr":"graduate student", "student_recurr": "undergraduate
student", "student_recurr": "international student",
"student_recurr":"recurring student", "student_recurr":"continuing student",
"student_recurr": "recurring undergraduate student", "student_recurr":
"recurring graduate student", "student_recurr": "ongoing undergraduate
student", "student_recurr":"ongoing graduate student"}
  - action_searchrecurringstudent
  - slot{"rinfo":"rstudent"}
  - utter_current_student
* good_thanks
  - utter_great
* yes_ido
  - utter_ask1
* university_enrollment{"uni_enroll":"enrollment"}
  - action_Eenrollment
  - slot{"university_enroll":"unii_enrollment"}
  - utter_enrollment
* ask_a_question
  - utter_ask2
* financial_aid_application{"FAFSA_APP":"FAFSA", "FAFSA_APP":"financial aid"}
  - action_FAFSA
  - slot{"fafsa_appp":"fafsa_application"}
  - utter_financial_aid
* ask_a_question
  - utter_ask3
* course_description{"getting_course": "computer science",
"getting_course":"comp sci", "getting_course": "comp science"}
  - action_view_cs_course
  - slot{"course_end":"course_start"}
  - utter_computerscience_option
* Spring2020
  - utter_classes_spring_2020
* courses_spring2020{"spring_2020_course1":"csc 590"}
  - action_view_spring2020
  - slot{"spring_2020c1":"spring_c1"}
  - utter_thank_you_uprogram
* goodbye
```

```
  - utter_goodbye
  - action_restart


<!--====================================================-->
<!--====================================================-->

## Story to view financial aid, enrollment, and computer science 582 then
ending the program mixed up SPRING term
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_two{"student_recurr":"ongoing student", "student_recurr":
"freshman", "student_recurr":"sophmore", "student_recurr":"junior",
"student_recurr":"senior", "student_recurr": "master's student",
"student_recurr": "first year  student", "student_recurr":"second year
student", "student_recurr" : "third year student", "student_recurr": "fourth
year student", "student_recurr": "fifth year student",
"student_recurr":"graduate student", "student_recurr": "undergraduate
student", "student_recurr": "international student",
"student_recurr":"recurring student", "student_recurr":"continuing student",
"student_recurr": "recurring undergraduate student", "student_recurr":
"recurring graduate student", "student_recurr": "ongoing undergraduate
student", "student_recurr":"ongoing graduate student"}
  - action_searchrecurringstudent
  - slot{"rinfo":"rstudent"}
  - utter_current_student
* good_thanks
  - utter_great
* yes_ido
  - utter_ask1
* university_enrollment{"uni_enroll":"enrollment"}
  - action_Eenrollment
  - slot{"university_enroll":"unii_enrollment"}
  - utter_enrollment
* ask_a_question
  - utter_ask2
* financial_aid_application{"FAFSA_APP":"FAFSA", "FAFSA_APP":"financial aid"}
  - action_FAFSA
  - slot{"fafsa_appp":"fafsa_application"}
  - utter_financial_aid
* ask_a_question
  - utter_ask3
* course_description{"getting_course": "computer science",
"getting_course":"comp sci", "getting_course": "comp science"}
  - action_view_cs_course
  - slot{"course_end":"course_start"}
  - utter_computerscience_option
* Spring2020
  - utter_classes_spring_2020
* courses_spring20202{"spring_2020_course3":"csc 582"}
  - action_view_spring20203
  - slot{"spring_2020c3":"spring_c3"}
```

```
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart


<!--==========================================================-->
<!--==========================================================-->
```

File: CurrentStudent_V2_Spring.md

```
## Story to view financial aid, enrollment, and computer science 585 then
ending the program mixed up SPRING term
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_two{"student_recurr":"ongoing student", "student_recurr":
"freshman", "student_recurr":"sophmore", "student_recurr":"junior",
"student_recurr":"senior", "student_recurr": "master's student",
"student_recurr": "first year  student", "student_recurr":"second year
student", "student_recurr" : "third year student", "student_recurr": "fourth
year student", "student_recurr": "fifth year student",
"student_recurr":"graduate student", "student_recurr": "undergraduate
student", "student_recurr": "international student",
"student_recurr":"recurring student", "student_recurr":"continuing student",
"student_recurr": "recurring undergraduate student", "student_recurr":
"recurring graduate student", "student_recurr": "ongoing undergraduate
student", "student_recurr":"ongoing graduate student"}
  - action_searchrecurringstudent
  - slot{"rinfo":"rstudent"}
  - utter_current_student
* good_thanks
  - utter_great
* yes_ido
  - utter_ask1
* university_enrollment{"uni_enroll":"enrollment"}
  - action_Eenrollment
  - slot{"university_enroll":"unii_enrollment"}
  - utter_enrollment
* ask_a_question
  - utter_ask2
* financial_aid_application{"FAFSA_APP":"FAFSA", "FAFSA_APP":"financial aid"}
  - action_FAFSA
  - slot{"fafsa_appp":"fafsa_application"}
  - utter_financial_aid
* ask_a_question
```

```
  - utter_ask3
* course_description{"getting_course": "computer science",
"getting_course":"comp sci", "getting_course": "comp science"}
  - action_view_cs_course
  - slot{"course_end":"course_start"}
  - utter_computerscience_option
* Spring2020
  - utter_classes_spring_2020
* courses_spring20201{"spring_2020_course2":"csc 585"}
  - action_view_spring20202
  - slot{"spring_2020c2":"spring_c2"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart


<!--====================================================-->
<!--====================================================-->

## Story to view financial aid, enrollment, and computer science 590 then
ending the program mixed up SPRING term
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_two{"student_recurr":"ongoing student", "student_recurr":
"freshman", "student_recurr":"sophmore", "student_recurr":"junior",
"student_recurr":"senior", "student_recurr": "master's student",
"student_recurr": "first year  student", "student_recurr":"second year
student", "student_recurr" : "third year student", "student_recurr": "fourth
year student", "student_recurr": "fifth year student",
"student_recurr":"graduate student", "student_recurr": "undergraduate
student", "student_recurr": "international student",
"student_recurr":"recurring student", "student_recurr":"continuing student",
"student_recurr": "recurring undergraduate student", "student_recurr":
"recurring graduate student", "student_recurr": "ongoing undergraduate
student", "student_recurr":"ongoing graduate student"}
  - action_searchrecurringstudent
  - slot{"rinfo":"rstudent"}
  - utter_current_student
* good_thanks
  - utter_great
* yes_ido
  - utter_ask1
* university_enrollment{"uni_enroll":"enrollment"}
  - action_Eenrollment
  - slot{"university_enroll":"unii_enrollment"}
  - utter_enrollment
* ask_a_question
  - utter_ask2
* financial_aid_application{"FAFSA_APP":"FAFSA", "FAFSA_APP":"financial aid"}
  - action_FAFSA
  - slot{"fafsa_appp":"fafsa_application"}
```

```
  - utter_financial_aid
* ask_a_question
  - utter_ask3
* course_description{"getting_course": "computer science",
"getting_course":"comp sci", "getting_course": "comp science"}
  - action_view_cs_course
  - slot{"course_end":"course_start"}
  - utter_computerscience_option
* Spring2020
  - utter_classes_spring_2020
* courses_spring2020{"spring_2020_course1":"csc 590"}
  - action_view_spring2020
  - slot{"spring_2020c1":"spring_c1"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart


<!--====================================================-->
<!--====================================================-->

## Story to view financial aid, enrollment, and computer science 582 then
ending the program mixed up SPRING term
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_two{"student_recurr":"ongoing student", "student_recurr":
"freshman", "student_recurr":"sophmore", "student_recurr":"junior",
"student_recurr":"senior", "student_recurr": "master's student",
"student_recurr": "first year  student", "student_recurr":"second year
student", "student_recurr" : "third year student", "student_recurr": "fourth
year student", "student_recurr": "fifth year student",
"student_recurr":"graduate student", "student_recurr": "undergraduate
student", "student_recurr": "international student",
"student_recurr":"recurring student", "student_recurr":"continuing student",
"student_recurr": "recurring undergraduate student", "student_recurr":
"recurring graduate student", "student_recurr": "ongoing undergraduate
student", "student_recurr":"ongoing graduate student"}
  - action_searchrecurringstudent
  - slot{"rinfo":"rstudent"}
  - utter_current_student
* good_thanks
  - utter_great
* yes_ido
  - utter_ask1
* university_enrollment{"uni_enroll":"enrollment"}
  - action_Eenrollment
  - slot{"university_enroll":"unii_enrollment"}
  - utter_enrollment
* ask_a_question
  - utter_ask2
* financial_aid_application{"FAFSA_APP":"FAFSA", "FAFSA_APP":"financial aid"}
```

```
  - action_FAFSA
  - slot{"fafsa_appp":"fafsa_application"}
  - utter_financial_aid
* ask_a_question
  - utter_ask3
* course_description{"getting_course": "computer science",
"getting_course":"comp sci", "getting_course": "comp science"}
  - action_view_cs_course
  - slot{"course_end":"course_start"}
  - utter_computerscience_option
* Spring2020
  - utter_classes_spring_2020
* courses_spring20202{"spring_2020_course3":"csc 582"}
  - action_view_spring20203
  - slot{"spring_2020c3":"spring_c3"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart


<!--====================================================-->
<!--====================================================-->
```

File: CurrentStudent_V2_Summer.md

```
<!--
## Karen Salinas
## CSC 590
## Story mixed up story with three elements
-->

## Story to view campus tour
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_two{"student_recurr":"ongoing student", "student_recurr":
"freshman", "student_recurr":"sophmore", "student_recurr":"junior",
"student_recurr":"senior", "student_recurr": "master's student",
"student_recurr": "first year  student", "student_recurr":"second year
student", "student_recurr" : "third year student", "student_recurr": "fourth
year student", "student_recurr": "fifth year student",
"student_recurr":"graduate student", "student_recurr": "undergraduate
student", "student_recurr": "international student",
"student_recurr":"recurring student", "student_recurr":"continuing student",
"student_recurr": "recurring undergraduate student", "student_recurr":
"recurring graduate student", "student_recurr": "ongoing undergraduate
student", "student_recurr":"ongoing graduate student"}
  - action_searchrecurringstudent
  - slot{"rinfo":"rstudent"}
  - utter_current_student
* good_thanks
```

```
  - utter_great
* yes_ido
  - utter_ask1
* university_enrollment{"uni_enroll":"enrollment"}
  - action_Eenrollment
  - slot{"university_enroll":"unii_enrollment"}
  - utter_enrollment
* ask_a_question
  - utter_ask2
* financial_aid_application{"FAFSA_APP":"FAFSA", "FAFSA_APP":"financial aid"}
  - action_FAFSA
  - slot{"fafsa_appp":"fafsa_application"}
  - utter_financial_aid
* ask_a_question
  - utter_ask3
* course_description{"getting_course": "computer science",
"getting_course":"comp sci", "getting_course": "comp science"}
  - action_view_cs_course
  - slot{"course_end":"course_start"}
  - utter_computerscience_option
* Summer2020
  - utter_classes_summer_20201
* courses_summer2020{"summer_2020_course1":"csc 500"}
  - action_view_summer2020
  - slot{"summer_2020c1":"summer_c1"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart



## Story to view financial aid, enrollment, and computer science 501 then
ending the program mixed up
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_two{"student_recurr":"ongoing student", "student_recurr":
"freshman", "student_recurr":"sophmore", "student_recurr":"junior",
"student_recurr":"senior", "student_recurr": "master's student",
"student_recurr": "first year  student", "student_recurr":"second year
student", "student_recurr" : "third year student", "student_recurr": "fourth
year student", "student_recurr": "fifth year student",
"student_recurr":"graduate student", "student_recurr": "undergraduate
student", "student_recurr": "international student",
"student_recurr":"recurring student", "student_recurr":"continuing student",
"student_recurr": "recurring undergraduate student", "student_recurr":
"recurring graduate student", "student_recurr": "ongoing undergraduate
student", "student_recurr":"ongoing graduate student"}
  - action_searchrecurringstudent
  - slot{"rinfo":"rstudent"}
  - utter_current_student
* good_thanks
```

```
  - utter_great
* yes_ido
  - utter_ask1
* university_enrollment{"uni_enroll":"enrollment"}
  - action_Eenrollment
  - slot{"university_enroll":"unii_enrollment"}
  - utter_enrollment
* ask_a_question
  - utter_ask2
* financial_aid_application{"FAFSA_APP":"FAFSA", "FAFSA_APP":"financial aid"}
  - action_FAFSA
  - slot{"fafsa_appp":"fafsa_application"}
  - utter_financial_aid
* ask_a_question
  - utter_ask3
* course_description{"getting_course": "computer science",
"getting_course":"comp sci", "getting_course": "comp science"}
  - action_view_cs_course
  - slot{"course_end":"course_start"}
  - utter_computerscience_option
* Summer2020
  - utter_classes_summer_20201
* courses_summer20201{"summer_2020_course2":"csc 501"}
  - action_view_summer20202
  - slot{"summer_2020c2":"summer_c2"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart
```

File: Current_V3.md

```
## Story to view financial aid, enrollment, and computer science 500 then
ending the program mixed up
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_two{"student_recurr":"ongoing student", "student_recurr":
"freshman", "student_recurr":"sophmore", "student_recurr":"junior",
"student_recurr":"senior", "student_recurr": "master's student",
"student_recurr": "first year  student", "student_recurr":"second year
student", "student_recurr" : "third year student", "student_recurr": "fourth
year student", "student_recurr": "fifth year student",
"student_recurr":"graduate student", "student_recurr": "undergraduate
student", "student_recurr": "international student",
"student_recurr":"recurring student", "student_recurr":"continuing student",
"student_recurr": "recurring undergraduate student", "student_recurr":
"recurring graduate student", "student_recurr": "ongoing undergraduate
student", "student_recurr":"ongoing graduate student"}
  - action_searchrecurringstudent
```

```
  - slot{"rinfo":"rstudent"}
  - utter_current_student
* good_thanks
  - utter_great
* yes_ido
  - utter_ask1
* on_campus{"campus_CSUDH":"campus"}
  - action_campus_info
  - slot{"campus_information":"campus_info"}
  - utter_campus_info
* on_campus_tour{"campus_tour_CSUDH":"campus tour"}
  - action_oncampus_tour
  - slot{"on_campus_tour_info":"campus_tour_info"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart


## Story to view financial aid, enrollment, and computer science 500 then
ending the program mixed up
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_two{"student_recurr":"ongoing student", "student_recurr":
"freshman", "student_recurr":"sophmore", "student_recurr":"junior",
"student_recurr":"senior", "student_recurr": "master's student",
"student_recurr": "first year  student", "student_recurr":"second year
student", "student_recurr" : "third year student", "student_recurr": "fourth
year student", "student_recurr": "fifth year student",
"student_recurr":"graduate student", "student_recurr": "undergraduate
student", "student_recurr": "international student",
"student_recurr":"recurring student", "student_recurr":"continuing student",
"student_recurr": "recurring undergraduate student", "student_recurr":
"recurring graduate student", "student_recurr": "ongoing undergraduate
student", "student_recurr":"ongoing graduate student"}
  - action_searchrecurringstudent
  - slot{"rinfo":"rstudent"}
  - utter_current_student
* good_thanks
  - utter_great
* yes_ido
  - utter_ask1
* on_campus{"campus_CSUDH":"campus"}
  - action_campus_info
  - slot{"campus_information":"campus_info"}
  - utter_campus_info
* campus_employment{"campus_employment_CSUDH":"employment"}
  - action_employment
  - slot{"campus_employment_stu":"campus_employment"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
```

```
  - action_restart


## Story to view financial aid, enrollment, and computer science 500 then
ending the program mixed up
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_two{"student_recurr":"ongoing student", "student_recurr":
"freshman", "student_recurr":"sophmore", "student_recurr":"junior",
"student_recurr":"senior", "student_recurr": "master's student",
"student_recurr": "first year  student", "student_recurr":"second year
student", "student_recurr" : "third year student", "student_recurr": "fourth
year student", "student_recurr": "fifth year student",
"student_recurr":"graduate student", "student_recurr": "undergraduate
student", "student_recurr": "international student",
"student_recurr":"recurring student", "student_recurr":"continuing student",
"student_recurr": "recurring undergraduate student", "student_recurr":
"recurring graduate student", "student_recurr": "ongoing undergraduate
student", "student_recurr":"ongoing graduate student"}
  - action_searchrecurringstudent
  - slot{"rinfo":"rstudent"}
  - utter_current_student
* good_thanks
  - utter_great
* yes_ido
  - utter_ask1
* on_campus{"campus_CSUDH":"campus"}
  - action_campus_info
  - slot{"campus_information":"campus_info"}
  - utter_campus_info
* campus_map{"campus_map_CSUDH":"campus map"}
  - action_campusmap
  - slot{"campusMapCSUDH":"campus_mapCSUDH"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart
```

File: IncomingGrad_Prog.md

```
## Story 1a: The Incoming Graduate Student WANTS more details on the grad
programs
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
```

```
* student_type_one{"student_incoming":"incoming student",
"student_incoming":"prospective student", "student_incoming": "applicant",
"student_incoming":"incoming undergraduate student",
"student_incoming":"incoming undergrad student", "student_incoming":
"incoming graduate student", "student_incoming": "incoming grad student",
"student_incoming": "incoming freshman", "student_incoming":"transfer
student", "student_incoming":"incoming first year student",
"student_incoming":"incoming master's student"}
  - action_searchincomingstudent
  - slot{"iinfo":"istudent"}
  - utter_future_student
  - utter_future_student_r2
  - utter_future_student_r3
* choose_incoming_grad_program{"program_graduate":"graduate program",
"program_graduate":"grad program"}
  - action_search_graduateprogram
  - slot{"gprograms":"program_graduate"}
  - utter_graduate
  - utter_graduate_program_options
* grad_program_info{"program_masters":"graduate program"}
  - action_graduatedd_program
  - slot{"graduate_program_details":"grad_program"}
  - utter_grad_option_program
* select_grad_program_details_yes{"program_grad_details_yes":"Yupp"}
  - action_grad_prog_yes
  - slot{"graduate_prog_yes":"grad_prog_yes"}
  - utter_grad_requirements
* select_grad_requirements_yes{"program_gradrequirement_details_yes":"Yas"}
  - action_grad_req_yes
  - slot{"graduate_progreq_yes":"grad_progreq_yes"}
  - utter_graduate_ask_school
* select_gradschool_yes{"view_gradschool_yes":"Si"}
  - action_gradinfo_yes
  - slot{"graduateinfo_yes":"gradinfo_yes"}
  - utter_grad_ask_steps
* select_gradsteps_yes{"view_grad_steps_yes":"yasss"}
  - action_grad_steps_apply_yes
  - slot{"graduatesteps_app_yes":"gradsteps_yes"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart


  <!--==========================================================-->
## Story 2a: The Incoming Graduate Student DOES NOT more details on the grad
programs
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_one{"student_incoming":"incoming student",
"student_incoming":"prospective student", "student_incoming": "applicant",
"student_incoming":"incoming undergraduate student",
```

```
"student_incoming":"incoming undergrad student", "student_incoming":
"incoming graduate student", "student_incoming": "incoming grad student",
"student_incoming": "incoming freshman", "student_incoming":"transfer
student", "student_incoming":"incoming first year student",
"student_incoming":"incoming master's student"}
  - action_searchincomingstudent
  - slot{"iinfo":"istudent"}
  - utter_future_student
  - utter_future_student_r2
  - utter_future_student_r3
* choose_incoming_grad_program{"program_graduate":"graduate program",
"program_graduate":"grad program"}
  - action_search_graduateprogram
  - slot{"gprograms":"program_graduate"}
  - utter_graduate
  - utter_graduate_program_options
* grad_program_info{"program_masters":"graduate program"}
  - action_graduatedd_program
  - slot{"graduate_program_details":"grad_program"}
  - utter_grad_option_program
* select_grad_program_details_no{"program_grad_details_no":"Noope"}
  - action_grad_prog_no
  - slot{"graduate_prog_no":"grad_prog_no"}
  - utter_thank_you_uprogram
* goodbye
- utter_goodbye
- action_restart


## Story 2b: The Incoming Graduate Student DOES NOT to view requirements on
the grad programs
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_one{"student_incoming":"incoming student",
"student_incoming":"prospective student", "student_incoming": "applicant",
"student_incoming":"incoming undergraduate student",
"student_incoming":"incoming undergrad student", "student_incoming":
"incoming graduate student", "student_incoming": "incoming grad student",
"student_incoming": "incoming freshman", "student_incoming":"transfer
student", "student_incoming":"incoming first year student",
"student_incoming":"incoming master's student"}
  - action_searchincomingstudent
  - slot{"iinfo":"istudent"}
  - utter_future_student
  - utter_future_student_r2
  - utter_future_student_r3
* choose_incoming_grad_program{"program_graduate":"graduate program",
"program_graduate":"grad program"}
  - action_search_graduateprogram
  - slot{"gprograms":"program_graduate"}
  - utter_graduate
  - utter_graduate_program_options
```

```
* grad_program_info{"program_masters":"graduate program"}
  - action_graduatedd_program
  - slot{"graduate_program_details":"grad_program"}
  - utter_grad_option_program
* select_grad_program_details_yes{"program_grad_details_yes":"Yupp"}
  - action_grad_prog_yes
  - slot{"graduate_prog_yes":"grad_prog_yes"}
  - utter_grad_requirements
* select_grad_requirements_no{"program_gradrequirement_details_no":"Nopee"}
  - action_grad_req_no
  - slot{"graduate_progreq_no":"grad_progreq_no"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart



## Story 2c: The Incoming Graduate Student WANTS more details on the grad
programs
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_one{"student_incoming":"incoming student",
"student_incoming":"prospective student", "student_incoming": "applicant",
"student_incoming":"incoming undergraduate student",
"student_incoming":"incoming undergrad student", "student_incoming":
"incoming graduate student", "student_incoming": "incoming grad student",
"student_incoming": "incoming freshman", "student_incoming":"transfer
student", "student_incoming":"incoming first year student",
"student_incoming":"incoming master's student"}
  - action_searchincomingstudent
  - slot{"iinfo":"istudent"}
  - utter_future_student
  - utter_future_student_r2
  - utter_future_student_r3
* choose_incoming_grad_program{"program_graduate":"graduate program",
"program_graduate":"grad program"}
  - action_search_graduateprogram
  - slot{"gprograms":"program_graduate"}
  - utter_graduate
  - utter_graduate_program_options
* grad_program_info{"program_masters":"graduate program"}
  - action_graduatedd_program
  - slot{"graduate_program_details":"grad_program"}
  - utter_grad_option_program
* select_grad_program_details_yes{"program_grad_details_yes":"Yupp"}
  - action_grad_prog_yes
  - slot{"graduate_prog_yes":"grad_prog_yes"}
  - utter_grad_requirements
* select_grad_requirements_yes{"program_gradrequirement_details_yes":"Yas"}
  - action_grad_req_yes
  - slot{"graduate_progreq_yes":"grad_progreq_yes"}
```

```
  - utter_graduate_ask_school
* select_gradschool_no{"view_gradschool_no":"Nay"}
  - action_gradinfo_no
  - slot{"graduate_progreq_no":"grad_progreq_no"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart



## Story 2d: The Incoming Graduate Student WANTS more details on the grad
programs
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_one{"student_incoming":"incoming student",
"student_incoming":"prospective student", "student_incoming": "applicant",
"student_incoming":"incoming undergraduate student",
"student_incoming":"incoming undergrad student", "student_incoming":
"incoming graduate student", "student_incoming": "incoming grad student",
"student_incoming": "incoming freshman", "student_incoming":"transfer
student", "student_incoming":"incoming first year student",
"student_incoming":"incoming master's student"}
  - action_searchincomingstudent
  - slot{"iinfo":"istudent"}
  - utter_future_student
  - utter_future_student_r2
  - utter_future_student_r3
* choose_incoming_grad_program{"program_graduate":"graduate program",
"program_graduate":"grad program"}
  - action_search_graduateprogram
  - slot{"gprograms":"program_graduate"}
  - utter_graduate
  - utter_graduate_program_options
* grad_program_info{"program_masters":"graduate program"}
  - action_graduatedd_program
  - slot{"graduate_program_details":"grad_program"}
  - utter_grad_option_program
* select_grad_program_details_yes{"program_grad_details_yes":"Yupp"}
  - action_grad_prog_yes
  - slot{"graduate_prog_yes":"grad_prog_yes"}
  - utter_grad_requirements
* select_grad_requirements_yes{"program_gradrequirement_details_yes":"Yas"}
  - action_grad_req_yes
  - slot{"graduate_progreq_yes":"grad_progreq_yes"}
  - utter_graduate_ask_school
* select_gradschool_yes{"view_gradschool_yes":"Si"}
  - action_gradinfo_yes
  - slot{"graduateinfo_yes":"gradinfo_yes"}
  - utter_grad_ask_steps
* select_gradsteps_no{"view_grad_steps_no":"nooooo"}
  - action_grad_steps_apply_no
```

```
  - slot{"graduatesteps_app_no":"gradsteps_no"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart
```

File: IncomingUndergraduate_Prog_topic1.md

```
## Story 1a: The Incoming Undergraduate Student WANTS more details on the
undergraduate programs with ART Program
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_one{"student_incoming":"incoming student",
"student_incoming":"prospective student", "student_incoming": "applicant",
"student_incoming":"incoming undergraduate student",
"student_incoming":"incoming undergrad student", "student_incoming":
"incoming graduate student", "student_incoming": "incoming grad student",
"student_incoming": "incoming freshman", "student_incoming":"transfer
student", "student_incoming":"incoming first year student",
"student_incoming":"incoming master's student"}
  - action_searchincomingstudent
  - slot{"iinfo":"istudent"}
  - utter_future_student
  - utter_future_student_r2
  - utter_future_student_r3
* choose_incoming_undergrad_program{"program_undergrad":"undergraduate
program", "program_undergrad":"undergrad program"}
  - action_search_undergraduateprogram
  - slot{"uprograms":"program_undergrad"}
  - utter_undergraduate
  - utter_undergraduate_program_options
* undergrad_programs_info{"program_undergrad_info":"undergrad program"}
  - action_display_undergradinfo
  - slot{"uprog_info":"uprog_iinfo"}
  - utter_option_program
* select_program_details_yes{"program_details_yes":"Of Course"}
  - action_prog_yes
  - slot{"program_yes":"prog_yes"}
  - utter_program_information
* art
  - utter_art_loc
  - utter_art_program
  - utter_curious_req
* select_requirements_yes{"program_requirement_details_yes":"Indeed"}
  - action_pro_req_yes
  - slot{"program_req_yes":"prog_req_yes"}
  - utter_ask_school
* select_school_info_yes{"school_info_yes":"Yessss"}
  - action_school_yes
  - slot{"schoolinfo_yes":"schooli_yes"}
```

```
  - utter_ask_steps
* select_stepstoapply_yes{"steps_apply_yes":"Steps yes"}
  - action_steps_apply_yes
  - slot{"steps_app_yes":"steps_yes"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart


## Story 2a: The Incoming Undergraduate Student WANTS more details on the
undergraduate requirements with ART Program
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_one{"student_incoming":"incoming student",
"student_incoming":"prospective student", "student_incoming": "applicant",
"student_incoming":"incoming undergraduate student",
"student_incoming":"incoming undergrad student", "student_incoming":
"incoming graduate student", "student_incoming": "incoming grad student",
"student_incoming": "incoming freshman", "student_incoming":"transfer
student", "student_incoming":"incoming first year student",
"student_incoming":"incoming master's student"}
  - action_searchincomingstudent
  - slot{"iinfo":"istudent"}
  - utter_future_student
  - utter_future_student_r2
  - utter_future_student_r3
* choose_incoming_undergrad_program{"program_undergrad":"undergraduate
program", "program_undergrad":"undergrad program"}
  - action_search_undergraduateprogram
  - slot{"uprograms":"program_undergrad"}
  - utter_undergraduate
  - utter_undergraduate_program_options
* undergrad_programs_info{"program_undergrad_info":"undergrad program"}
  - action_display_undergradinfo
  - slot{"uprog_info":"uprog_iinfo"}
  - utter_option_program
* select_program_details_yes{"program_details_yes":"Of Course"}
  - action_prog_yes
  - slot{"program_yes":"prog_yes"}
  - utter_program_information
* art
  - utter_art_loc
  - utter_art_program
  - utter_curious_req
* select_requirements_no{"program_requirement_details_no":"Definately Not"}
  - action_pro_req_no
  - slot{"program_req_no":"prog_req_no"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart
```

```
<!--==========================================================-->
## Story 2b: The Incoming Undergraduate Student NOT want more details on
requirements with ART Program
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_one{"student_incoming":"incoming student",
"student_incoming":"prospective student", "student_incoming": "applicant",
"student_incoming":"incoming undergraduate student",
"student_incoming":"incoming undergrad student", "student_incoming":
"incoming graduate student", "student_incoming": "incoming grad student",
"student_incoming": "incoming freshman", "student_incoming":"transfer
student", "student_incoming":"incoming first year student",
"student_incoming":"incoming master's student"}
  - action_searchincomingstudent
  - slot{"iinfo":"istudent"}
  - utter_future_student
  - utter_future_student_r2
  - utter_future_student_r3
* choose_incoming_undergrad_program{"program_undergrad":"undergraduate
program", "program_undergrad":"undergrad program"}
  - action_search_undergraduateprogram
  - slot{"uprograms":"program_undergrad"}
  - utter_undergraduate
  - utter_undergraduate_program_options
* undergrad_programs_info{"program_undergrad_info":"undergrad program"}
  - action_display_undergradinfo
  - slot{"uprog_info":"uprog_iinfo"}
  - utter_option_program
* select_program_details_no{"program_details_no":"Of Course Not"}
  - action_prog_no
  - slot{"program_no":"prog_no"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart



## Story 2c: The Incoming Undergraduate Student NOT WANT details on school
information with ART Program
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_one{"student_incoming":"incoming student",
"student_incoming":"prospective student", "student_incoming": "applicant",
```

```
"student_incoming":"incoming undergraduate student",
"student_incoming":"incoming undergrad student", "student_incoming":
"incoming graduate student", "student_incoming": "incoming grad student",
"student_incoming": "incoming freshman", "student_incoming":"transfer
student", "student_incoming":"incoming first year student",
"student_incoming":"incoming master's student"}
  - action_searchincomingstudent
  - slot{"iinfo":"istudent"}
  - utter_future_student
  - utter_future_student_r2
  - utter_future_student_r3
* choose_incoming_undergrad_program{"program_undergrad":"undergraduate
program", "program_undergrad":"undergrad program"}
  - action_search_undergraduateprogram
  - slot{"uprograms":"program_undergrad"}
  - utter_undergraduate
  - utter_undergraduate_program_options
* undergrad_programs_info{"program_undergrad_info":"undergrad program"}
  - action_display_undergradinfo
  - slot{"uprog_info":"uprog_iinfo"}
  - utter_option_program
* select_program_details_yes{"program_details_yes":"Of Course"}
  - action_prog_yes
  - slot{"program_yes":"prog_yes"}
  - utter_program_information
* art
  - utter_art_loc
  - utter_art_program
  - utter_curious_req
* select_requirements_yes{"program_requirement_details_yes":"Indeed"}
  - action_pro_req_yes
  - slot{"program_req_yes":"prog_req_yes"}
  - utter_ask_school
* select_school_info_no{"school_info_no":"Nooo"}
  - action_school_no
  - slot{"schoolinfo_no":"schooli_no"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart


## Story 2d: The Incoming Undergraduate Student WANTS more details on the
undergraduate programs with ART Program
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_one{"student_incoming":"incoming student",
"student_incoming":"prospective student", "student_incoming": "applicant",
"student_incoming":"incoming undergraduate student",
"student_incoming":"incoming undergrad student", "student_incoming":
"incoming graduate student", "student_incoming": "incoming grad student",
"student_incoming": "incoming freshman", "student_incoming":"transfer
```

```
student", "student_incoming":"incoming first year student",
"student_incoming":"incoming master's student"}
  - action_searchincomingstudent
  - slot{"iinfo":"istudent"}
  - utter_future_student
  - utter_future_student_r2
  - utter_future_student_r3
* choose_incoming_undergrad_program{"program_undergrad":"undergraduate
program", "program_undergrad":"undergrad program"}
  - action_search_undergraduateprogram
  - slot{"uprograms":"program_undergrad"}
  - utter_undergraduate
  - utter_undergraduate_program_options
* undergrad_programs_info{"program_undergrad_info":"undergrad program"}
  - action_display_undergradinfo
  - slot{"uprog_info":"uprog_iinfo"}
  - utter_option_program
* select_program_details_yes{"program_details_yes":"Of Course"}
  - action_prog_yes
  - slot{"program_yes":"prog_yes"}
  - utter_program_information
* art
  - utter_art_loc
  - utter_art_program
  - utter_curious_req
* select_requirements_yes{"program_requirement_details_yes":"Indeed"}
  - action_pro_req_yes
  - slot{"program_req_yes":"prog_req_yes"}
  - utter_ask_school
* select_school_info_yes{"school_info_yes":"Yessss"}
  - action_school_yes
  - slot{"schoolinfo_yes":"schooli_yes"}
  - utter_ask_steps
* select_stepstoapply_no{"steps_apply_no":"Steps no"}
  - action_steps_apply_no
  - slot{"steps_app_no":"steps_no"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart
```

File: IncomingUndergraduate_Prog_topic2.md

```
## Story 1a: The Incoming Undergraduate Student WANTS more details on the
undergraduate programs with BUSINESS ADMIN
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_one{"student_incoming":"incoming student",
"student_incoming":"prospective student", "student_incoming": "applicant",
"student_incoming":"incoming undergraduate student",
```

```
"student_incoming":"incoming undergrad student", "student_incoming":
"incoming graduate student", "student_incoming": "incoming grad student",
"student_incoming": "incoming freshman", "student_incoming":"transfer
student", "student_incoming":"incoming first year student",
"student_incoming":"incoming master's student"}
  - action_searchincomingstudent
  - slot{"iinfo":"istudent"}
  - utter_future_student
  - utter_future_student_r2
  - utter_future_student_r3
* choose_incoming_undergrad_program{"program_undergrad":"undergraduate
program", "program_undergrad":"undergrad program"}
  - action_search_undergraduateprogram
  - slot{"uprograms":"program_undergrad"}
  - utter_undergraduate
  - utter_undergraduate_program_options
* undergrad_programs_info{"program_undergrad_info":"undergrad program"}
  - action_display_undergradinfo
  - slot{"uprog_info":"uprog_iinfo"}
  - utter_option_program
* select_program_details_yes{"program_details_yes":"Of Course"}
  - action_prog_yes
  - slot{"program_yes":"prog_yes"}
  - utter_program_information
* business_admin
  - utter_business_admin_description
  - utter_business_admin
  - utter_curious_req
* select_requirements_yes{"program_requirement_details_yes":"Indeed"}
  - action_pro_req_yes
  - slot{"program_req_yes":"prog_req_yes"}
  - utter_ask_school
* select_school_info_yes{"school_info_yes":"Yessss"}
  - action_school_yes
  - slot{"schoolinfo_yes":"schooli_yes"}
  - utter_ask_steps
* select_stepstoapply_yes{"steps_apply_yes":"Steps yes"}
  - action_steps_apply_yes
  - slot{"steps_app_yes":"steps_yes"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart


## Story 2a: The Incoming Undergraduate Student WANTS more details on the
undergraduate requirements
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_one{"student_incoming":"incoming student",
"student_incoming":"prospective student", "student_incoming": "applicant",
"student_incoming":"incoming undergraduate student",
```

```
"student_incoming":"incoming undergrad student", "student_incoming":
"incoming graduate student", "student_incoming": "incoming grad student",
"student_incoming": "incoming freshman", "student_incoming":"transfer
student", "student_incoming":"incoming first year student",
"student_incoming":"incoming master's student"}
  - action_searchincomingstudent
  - slot{"iinfo":"istudent"}
  - utter_future_student
  - utter_future_student_r2
  - utter_future_student_r3
* choose_incoming_undergrad_program{"program_undergrad":"undergraduate
program", "program_undergrad":"undergrad program"}
  - action_search_undergraduateprogram
  - slot{"uprograms":"program_undergrad"}
  - utter_undergraduate
  - utter_undergraduate_program_options
* undergrad_programs_info{"program_undergrad_info":"undergrad program"}
  - action_display_undergradinfo
  - slot{"uprog_info":"uprog_iinfo"}
  - utter_option_program
* select_program_details_yes{"program_details_yes":"Of Course"}
  - action_prog_yes
  - slot{"program_yes":"prog_yes"}
  - utter_program_information
* business_admin
  - utter_business_admin_description
  - utter_business_admin
  - utter_curious_req
* select_requirements_no{"program_requirement_details_no":"Definately Not"}
  - action_pro_req_no
  - slot{"program_req_no":"prog_req_no"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart



<!--====================================================================-->
## Story 2b: The Incoming Undergraduate Student NOT want more details on
requirements with BUSINESS ADMIN
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_one{"student_incoming":"incoming student",
"student_incoming":"prospective student", "student_incoming": "applicant",
"student_incoming":"incoming undergraduate student",
"student_incoming":"incoming undergrad student", "student_incoming":
"incoming graduate student", "student_incoming": "incoming grad student",
"student_incoming": "incoming freshman", "student_incoming":"transfer
student", "student_incoming":"incoming first year student",
"student_incoming":"incoming master's student"}
  - action_searchincomingstudent
```

```
  - slot{"iinfo":"istudent"}
  - utter_future_student
  - utter_future_student_r2
  - utter_future_student_r3
* choose_incoming_undergrad_program{"program_undergrad":"undergraduate
program", "program_undergrad":"undergrad program"}
  - action_search_undergraduateprogram
  - slot{"uprograms":"program_undergrad"}
  - utter_undergraduate
  - utter_undergraduate_program_options
* undergrad_programs_info{"program_undergrad_info":"undergrad program"}
  - action_display_undergradinfo
  - slot{"uprog_info":"uprog_iinfo"}
  - utter_option_program
* select_program_details_no{"program_details_no":"Of Course Not"}
  - action_prog_no
  - slot{"program_no":"prog_no"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart


## Story 2c: The Incoming Undergraduate Student NOT WANT details on school
information with BUSINESS ADMIN
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_one{"student_incoming":"incoming student",
"student_incoming":"prospective student", "student_incoming": "applicant",
"student_incoming":"incoming undergraduate student",
"student_incoming":"incoming undergrad student", "student_incoming":
"incoming graduate student", "student_incoming": "incoming grad student",
"student_incoming": "incoming freshman", "student_incoming":"transfer
student", "student_incoming":"incoming first year student",
"student_incoming":"incoming master's student"}
  - action_searchincomingstudent
  - slot{"iinfo":"istudent"}
  - utter_future_student
  - utter_future_student_r2
  - utter_future_student_r3
* choose_incoming_undergrad_program{"program_undergrad":"undergraduate
program", "program_undergrad":"undergrad program"}
  - action_search_undergraduateprogram
  - slot{"uprograms":"program_undergrad"}
  - utter_undergraduate
  - utter_undergraduate_program_options
* undergrad_programs_info{"program_undergrad_info":"undergrad program"}
  - action_display_undergradinfo
  - slot{"uprog_info":"uprog_iinfo"}
  - utter_option_program
* select_program_details_yes{"program_details_yes":"Of Course"}
  - action_prog_yes
```

```
  - slot{"program_yes":"prog_yes"}
  - utter_program_information
* business_admin
  - utter_business_admin_description
  - utter_business_admin
  - utter_curious_req
* select_requirements_yes{"program_requirement_details_yes":"Indeed"}
  - action_pro_req_yes
  - slot{"program_req_yes":"prog_req_yes"}
  - utter_ask_school
* select_school_info_no{"school_info_no":"Nooo"}
  - action_school_no
  - slot{"schoolinfo_no":"schooli_no"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart



## Story 2d: The Incoming Undergraduate Student WANTS more details on the
undergraduate programs with BUSINESS ADMIN
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_one{"student_incoming":"incoming student",
"student_incoming":"prospective student", "student_incoming": "applicant",
"student_incoming":"incoming undergraduate student",
"student_incoming":"incoming undergrad student", "student_incoming":
"incoming graduate student", "student_incoming": "incoming grad student",
"student_incoming": "incoming freshman", "student_incoming":"transfer
student", "student_incoming":"incoming first year student",
"student_incoming":"incoming master's student"}
  - action_searchincomingstudent
  - slot{"iinfo":"istudent"}
  - utter_future_student
  - utter_future_student_r2
  - utter_future_student_r3
* choose_incoming_undergrad_program{"program_undergrad":"undergraduate
program", "program_undergrad":"undergrad program"}
  - action_search_undergraduateprogram
  - slot{"uprograms":"program_undergrad"}
  - utter_undergraduate
  - utter_undergraduate_program_options
* undergrad_programs_info{"program_undergrad_info":"undergrad program"}
  - action_display_undergradinfo
  - slot{"uprog_info":"uprog_iinfo"}
  - utter_option_program
* select_program_details_yes{"program_details_yes":"Of Course"}
  - action_prog_yes
  - slot{"program_yes":"prog_yes"}
  - utter_program_information
* business_admin
```

```
  - utter_business_admin_description
  - utter_business_admin
  - utter_curious_req
* select_requirements_yes{"program_requirement_details_yes":"Indeed"}
  - action_pro_req_yes
  - slot{"program_req_yes":"prog_req_yes"}
  - utter_ask_school
* select_school_info_yes{"school_info_yes":"Yessss"}
  - action_school_yes
  - slot{"schoolinfo_yes":"schooli_yes"}
  - utter_ask_steps
* select_stepstoapply_no{"steps_apply_no":"Steps no"}
  - action_steps_apply_no
  - slot{"steps_app_no":"steps_no"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart
```

File: IncomingUndergraduate_Prog_topic3.md

```
## Story 1a: The Incoming Undergraduate Student WANTS more details on the
undergraduate programs with COMPUTER SCIENCE
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_one{"student_incoming":"incoming student",
"student_incoming":"prospective student", "student_incoming": "applicant",
"student_incoming":"incoming undergraduate student",
"student_incoming":"incoming undergrad student", "student_incoming":
"incoming graduate student", "student_incoming": "incoming grad student",
"student_incoming": "incoming freshman", "student_incoming":"transfer
student", "student_incoming":"incoming first year student",
"student_incoming":"incoming master's student"}
  - action_searchincomingstudent
  - slot{"iinfo":"istudent"}
  - utter_future_student
  - utter_future_student_r2
  - utter_future_student_r3
* choose_incoming_undergrad_program{"program_undergrad":"undergraduate
program", "program_undergrad":"undergrad program"}
  - action_search_undergraduateprogram
  - slot{"uprograms":"program_undergrad"}
  - utter_undergraduate
  - utter_undergraduate_program_options
* undergrad_programs_info{"program_undergrad_info":"undergrad program"}
  - action_display_undergradinfo
  - slot{"uprog_info":"uprog_iinfo"}
  - utter_option_program
* select_program_details_yes{"program_details_yes":"Of Course"}
  - action_prog_yes
```

```
  - slot{"program_yes":"prog_yes"}
  - utter_program_information
* computer_science
  - utter_comp_sci
  - utter_comp_sci_req
  - utter_curious_req
* select_requirements_yes{"program_requirement_details_yes":"Indeed"}
  - action_pro_req_yes
  - slot{"program_req_yes":"prog_req_yes"}
  - utter_ask_school
* select_school_info_yes{"school_info_yes":"Yessss"}
  - action_school_yes
  - slot{"schoolinfo_yes":"schooli_yes"}
  - utter_ask_steps
* select_stepstoapply_yes{"steps_apply_yes":"Steps yes"}
  - action_steps_apply_yes
  - slot{"steps_app_yes":"steps_yes"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart

## Story 2a: The Incoming Undergraduate Student WANTS more details on the
undergraduate requirements with COMPUTER SCIENCE
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_one{"student_incoming":"incoming student",
"student_incoming":"prospective student", "student_incoming": "applicant",
"student_incoming":"incoming undergraduate student",
"student_incoming":"incoming undergrad student", "student_incoming":
"incoming graduate student", "student_incoming": "incoming grad student",
"student_incoming": "incoming freshman", "student_incoming":"transfer
student", "student_incoming":"incoming first year student",
"student_incoming":"incoming master's student"}
  - action_searchincomingstudent
  - slot{"iinfo":"istudent"}
  - utter_future_student
  - utter_future_student_r2
  - utter_future_student_r3
* choose_incoming_undergrad_program{"program_undergrad":"undergraduate
program", "program_undergrad":"undergrad program"}
  - action_search_undergraduateprogram
  - slot{"uprograms":"program_undergrad"}
  - utter_undergraduate
  - utter_undergraduate_program_options
* undergrad_programs_info{"program_undergrad_info":"undergrad program"}
  - action_display_undergradinfo
  - slot{"uprog_info":"uprog_iinfo"}
  - utter_option_program
* select_program_details_yes{"program_details_yes":"Of Course"}
  - action_prog_yes
  - slot{"program_yes":"prog_yes"}
```

```
  - utter_program_information
* computer_science
  - utter_comp_sci
  - utter_comp_sci_req
  - utter_curious_req
* select_requirements_no{"program_requirement_details_no":"Definately Not"}
  - action_pro_req_no
  - slot{"program_req_no":"prog_req_no"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart


<!--==========================================================-->
## Story 2b: The Incoming Undergraduate Student NOT want more details on
requirements with COMPUTER SCIENCE
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_one{"student_incoming":"incoming student",
"student_incoming":"prospective student", "student_incoming": "applicant",
"student_incoming":"incoming undergraduate student",
"student_incoming":"incoming undergrad student", "student_incoming":
"incoming graduate student", "student_incoming": "incoming grad student",
"student_incoming": "incoming freshman", "student_incoming":"transfer
student", "student_incoming":"incoming first year student",
"student_incoming":"incoming master's student"}
  - action_searchincomingstudent
  - slot{"iinfo":"istudent"}
  - utter_future_student
  - utter_future_student_r2
  - utter_future_student_r3
* choose_incoming_undergrad_program{"program_undergrad":"undergraduate
program", "program_undergrad":"undergrad program"}
  - action_search_undergraduateprogram
  - slot{"uprograms":"program_undergrad"}
  - utter_undergraduate
  - utter_undergraduate_program_options
* undergrad_programs_info{"program_undergrad_info":"undergrad program"}
  - action_display_undergradinfo
  - slot{"uprog_info":"uprog_iinfo"}
  - utter_option_program
* select_program_details_no{"program_details_no":"Of Course Not"}
  - action_prog_no
  - slot{"program_no":"prog_no"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart
```

```
## Story 2c: The Incoming Undergraduate Student NOT WANT details on school
information with COMPUTER SCIENCE
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_one{"student_incoming":"incoming student",
"student_incoming":"prospective student", "student_incoming": "applicant",
"student_incoming":"incoming undergraduate student",
"student_incoming":"incoming undergrad student", "student_incoming":
"incoming graduate student", "student_incoming": "incoming grad student",
"student_incoming": "incoming freshman", "student_incoming":"transfer
student", "student_incoming":"incoming first year student",
"student_incoming":"incoming master's student"}
  - action_searchincomingstudent
  - slot{"iinfo":"istudent"}
  - utter_future_student
  - utter_future_student_r2
  - utter_future_student_r3
* choose_incoming_undergrad_program{"program_undergrad":"undergraduate
program", "program_undergrad":"undergrad program"}
  - action_search_undergraduateprogram
  - slot{"uprograms":"program_undergrad"}
  - utter_undergraduate
  - utter_undergraduate_program_options
* undergrad_programs_info{"program_undergrad_info":"undergrad program"}
  - action_display_undergradinfo
  - slot{"uprog_info":"uprog_iinfo"}
  - utter_option_program
* select_program_details_yes{"program_details_yes":"Of Course"}
  - action_prog_yes
  - slot{"program_yes":"prog_yes"}
  - utter_program_information
* computer_science
  - utter_comp_sci
  - utter_comp_sci_req
  - utter_curious_req
* select_requirements_yes{"program_requirement_details_yes":"Indeed"}
  - action_pro_req_yes
  - slot{"program_req_yes":"prog_req_yes"}
  - utter_ask_school
* select_school_info_no{"school_info_no":"Nooo"}
  - action_school_no
  - slot{"schoolinfo_no":"schooli_no"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart


## Story 2d: The Incoming Undergraduate Student WANTS more details on the
undergraduate programs with COMPUTER SCIENCE
* greet
  - utter_introduction
```

```
  - utter_introduction_r2
* choice_text{"choice":"text"}
  - action_choice_two
  - slot{"choice_two":"c_two"}
  - utter_ask_type_of_student
* student_type_one{"student_incoming":"incoming student",
"student_incoming":"prospective student", "student_incoming": "applicant",
"student_incoming":"incoming undergraduate student",
"student_incoming":"incoming undergrad student", "student_incoming":
"incoming graduate student", "student_incoming": "incoming grad student",
"student_incoming": "incoming freshman", "student_incoming":"transfer
student", "student_incoming":"incoming first year student",
"student_incoming":"incoming master's student"}
  - action_searchincomingstudent
  - slot{"iinfo":"istudent"}
  - utter_future_student
  - utter_future_student_r2
  - utter_future_student_r3
* choose_incoming_undergrad_program{"program_undergrad":"undergraduate
program", "program_undergrad":"undergrad program"}
  - action_search_undergraduateprogram
  - slot{"uprograms":"program_undergrad"}
  - utter_undergraduate
  - utter_undergraduate_program_options
* undergrad_programs_info{"program_undergrad_info":"undergrad program"}
  - action_display_undergradinfo
  - slot{"uprog_info":"uprog_iinfo"}
  - utter_option_program
* select_program_details_yes{"program_details_yes":"Of Course"}
  - action_prog_yes
  - slot{"program_yes":"prog_yes"}
  - utter_program_information
* computer_science
  - utter_comp_sci
  - utter_comp_sci_req
  - utter_curious_req
* select_requirements_yes{"program_requirement_details_yes":"Indeed"}
  - action_pro_req_yes
  - slot{"program_req_yes":"prog_req_yes"}
  - utter_ask_school
* select_school_info_yes{"school_info_yes":"Yessss"}
  - action_school_yes
  - slot{"schoolinfo_yes":"schooli_yes"}
  - utter_ask_steps
* select_stepstoapply_no{"steps_apply_no":"Steps no"}
  - action_steps_apply_no
  - slot{"steps_app_no":"steps_no"}
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart
```

File: Inquiries.md

```
## Slots of payloads
* greet
```

```
  - utter_introduction
  - utter_introduction_r2
* choice_payloads{"choice":"payloads"}
  - action_choice_one
  - slot{"choice_one":"c_one"}
  <!-- - utter_information
* help_inq -->
  - utter_menu

## Inquiry Student Makes with more details
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_payloads{"choice":"payloads"}
  - action_choice_one
  - slot{"choice_one":"c_one"}
  <!-- - utter_information
* help_inq -->
  - utter_menu
* begin_lead
  - utter_lead_q1
  - lead_form_p1
  - form{"name": "lead_form_p1"}
  - form{"name": null}
* accept
  - lead_form_p2
  - form{"name": "lead_form_p2"}
  - form{"name": null}
  - lead_form_p3
  - form{"name": "lead_form_p3"}
  - form{"name": null}
  - utter_lead_q3
  - utter_lead_q4
  - utter_lead_q5
  - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart

## Inquiry Student Makes With Less Details
* greet
  - utter_introduction
  - utter_introduction_r2
* choice_payloads{"choice":"payloads"}
  - action_choice_one
  - slot{"choice_one":"c_one"}
  <!-- - utter_information
* help_inq -->
  - utter_menu
* begin_lead
    - utter_lead_q1
    - lead_form_p1
    - form{"name": "lead_form_p1"}
    - form{"name": null}
* reject
    - lead_form_p3
    - form{"name": "lead_form_p3"}
```

```
    - form{"name": null}
    - utter_lead_q3
    - utter_lead_q4
    - utter_lead_q5
    - utter_thank_you_uprogram
* goodbye
  - utter_goodbye
  - action_restart
```

File: Stories.md

```
## default_fallback
* bot_challenge
  - action_default_fallback
```

File: index.html

```
<!-- Credit to obtaining template from JiteshGaikwad on
https://github.com/JiteshGaikwad/Chatbot-Widget,. Used many
of his methods, but some were not working and opted out to
 using some of his features.-->
<html>

<head>
    <title>Chatbot Widget</title>

    <!--Let browser know website is optimized for mobile-->
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <!--Import Google Icon Font-->
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
rel="stylesheet">
    <link
href="https://fonts.googleapis.com/css?family=Raleway:500&display=swap"
rel="stylesheet">

    <!--Import Font Awesome Icon Font-->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css" integrity="sha256-
eZrrJcwDc/3uDhsdt61sL2oOBY362qM3lon1gyExkL0=" crossorigin="anonymous" />

    <!--Import materialize.css-->
    <link rel="stylesheet" type="text/css"
href="static/css/materialize.min.css">

    <!--Main css-->
    <link rel="stylesheet" type="text/css" href="static/css/style.css">
    <meta name="viewport" content="width=device-width, initial-scale=1">

</head>

<body>
    <div class="container">
```

```
        <!-- Modal for rendering the charts, declare this if you want to
render charts,
         else you remove the modal -->
        <div id="modal1" class="modal">
            <canvas id="modal-chart"></canvas>
        </div>

        <!--chatbot widget -->
        <div class="widget">
            <div class="chat_header">

                <!--Add the name of the bot here -->
                <span class="chat_header_title">CSUDHBot</span>
                <span class="dropdown-trigger" href='#' data-
target='dropdown1'>
                  <i class="material-icons">
                  more_vert
                  </i>
               </span>

                <!-- Dropdown menu-->
                <ul id='dropdown1' class='dropdown-content'>
                    <li><a href="#" id="clear">Clear</a></li>
                    <li><a href="#" id="restart">Restart</a></li>
                    <li><a href="#" id="close">Close</a></li>
                </ul>
            </div>

            <!--Chatbot contents goes here -->
            <div class="chats" id="chats">
                <div class="clearfix"></div>

            </div>

            <!--keypad for user to type the message -->
            <div class="keypad">
                <textarea id="userInput" placeholder="Type a message..."
class="usrInput"></textarea>
                <div id="sendButton"><i class="fa fa-paper-plane" aria-
hidden="true"></i></div>
            </div>
        </div>

        <!--bot profile-->
        <div class="profile_div" id="profile_div">
            <img class="imgProfile" src="static/img/DominguezToros.png" />
        </div>

        <!-- Bot pop-up intro -->
        <div class="tap-target" data-target="profile_div">
            <div class="tap-target-content">
                <h5 class="white-text"> Hi there! 👋</h5>
                <p class="white-text">I am CSUDHBot!</p>
            </div>
        </div>
```

```
    </div>

    <!--JavaScript at end of body for optimized loading-->
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></scri
pt>
    <script type="text/javascript"
src="static/js/materialize.min.js"></script>

    <!--Main Script -->
    <script type="text/javascript" src="static/js/script.js"></script>

    <!--Chart.js Script -->
    <script type="text/javascript" src="static/js/chart.min.js"></script>

    <iframe src = "https://www.csudh.edu/" width="100%" height="100%"
</iframe>

</body>

</html>
```

File: config.yml

```
language: en
pipeline: supervised_embeddings
policies:
- name: FormPolicy
- name: AugmentedMemoizationPolicy
- name: TEDPolicy
- name: MappingPolicy

## Many errors persisted when using the FallbackPolicy, such as
"utter_custom", therefore opted out
- name: FallbackPolicy
  nlu_threshold: 0.4
  core_threshold: 0.4
  fallback_action_name: "action_default_fallback"
```

File: endpoints.yml

```
action_endpoint:
  url: http://localhost:5055/webhook

#Tracker store using PostgreSQL Database
tracker_store:
  type: SQL
  dialect: "postgresql"
  url: "localhost"
  db: "postgres"
  username: "postgres"
  password: "Ks126352"
```

File: credentials.yml

```
# This file contains the credentials for the voice & chat platforms
# which your bot is using.
# https://rasa.com/docs/rasa/user-guide/messaging-and-voice-channels/

rest:

##using socketio to connect to web-chat widget.
socketio:
  user_message_evt: user_uttered
  bot_message_evt: bot_uttered
  session_persistence: true/false

rasa:
  url: "http://localhost:5002/api"
```

File:script.js

```
// Credit to to script from JiteshGaikwad on
https://github.com/JiteshGaikwad/Chatbot-Widget,
//Bot pop-up intro
document.addEventListener('DOMContentLoaded', function() {
    var elemsTap = document.querySelector('.tap-target');
    var instancesTap = M.TapTarget.init(elemsTap, {});
    instancesTap.open();
    setTimeout(function() { instancesTap.close(); }, 4000);

});


//initialization
$(document).ready(function() {


    //Bot pop-up intro
    $("div").removeClass("tap-target-origin")

    //drop down menu for close, restart conversation & clear the chats.
    $('.dropdown-trigger').dropdown();

    //initiate the modal for displaying the charts, if you dont have charts,
then you comment the below line
    $('.modal').modal();



    //enable this if u have configured the bot to start the conversation.
    // showBotTyping();
    // $("#userInput").prop('disabled', true);

    //global variables
    action_name = "action_greet_user";
    user_id = "ksalinas9";

    //if you want the bot to start the conversation
```

```
    // action_trigger();

})

// ========================== restart conversation ======================
function restartConversation() {
    $("#userInput").prop('disabled', true);
    //destroy the existing chart
    $('.collapsible').remove();

    if (typeof chatChart !== 'undefined') { chatChart.destroy(); }

    $(".chart-container").remove();
    if (typeof modalChart !== 'undefined') { modalChart.destroy(); }
    $(".chats").html("");
    $(".usrInput").val("");
    send("/restart");
}

// ============= let the bot start the conversation ======================
function action_trigger() {

    // send an event to the bot, so that bot can start the conversation by
greeting the user
    $.ajax({
        url: `http://localhost:5005/conversations/${user_id}/execute`,
        type: "POST",
        contentType: "application/json",
        data: JSON.stringify({ "name": action_name, "policy":
"MappingPolicy", "confidence": "0.98" }),
        success: function(botResponse, status) {
            console.log("Response from Rasa: ", botResponse, "\nStatus: ",
status);

            if (botResponse.hasOwnProperty("messages")) {
                setBotResponse(botResponse.messages);
            }
            $("#userInput").prop('disabled', false);
        },
        error: function(xhr, textStatus, errorThrown) {

            // if there is no response from rasa server
            setBotResponse("");
            console.log("Error from bot end: ", textStatus);
            $("#userInput").prop('disabled', false);
        }
    });
}

//==================  user enter or sends the message ====================
$(".usrInput").on("keyup keypress", function(e) {
    var keyCode = e.keyCode || e.which;

    var text = $(".usrInput").val();
    if (keyCode === 13) {

        if (text == "" || $.trim(text) == "") {
```

```
            e.preventDefault();
            return false;
        } else {
            //destroy the existing chart, if yu are not using charts, then
comment the below lines
            $('.collapsible').remove();
            if (typeof chatChart !== 'undefined') { chatChart.destroy(); }

            $(".chart-container").remove();
            if (typeof modalChart !== 'undefined') { modalChart.destroy(); }



            $("#paginated_cards").remove();
            $(".suggestions").remove();
            $(".quickReplies").remove();
            $(".usrInput").blur();
            setUserResponse(text);
            send(text);
            e.preventDefault();
            return false;
        }
    }
});

$("#sendButton").on("click", function(e) {
    var text = $(".usrInput").val();
    if (text == "" || $.trim(text) == "") {
        e.preventDefault();
        return false;
    } else {
        //destroy the existing chart

        chatChart.destroy();
        $(".chart-container").remove();
        if (typeof modalChart !== 'undefined') { modalChart.destroy(); }

        $(".suggestions").remove();
        $("#paginated_cards").remove();
        $(".quickReplies").remove();
        $(".usrInput").blur();
        setUserResponse(text);
        send(text);
        e.preventDefault();
        return false;
    }
})

//================================== Set user response ==========
function setUserResponse(message) {
    var UserResponse = '<img class="userAvatar" src=' +
"./static/img/userAvatar.jpg" + '><p class="userMsg">' + message + ' </p><div
class="clearfix"></div>';
    $(UserResponse).appendTo(".chats").show("slow");

    $(".usrInput").val("");
    scrollToBottomOfResults();
```

```
    showBotTyping();
    $(".suggestions").remove();
}

//=========== Scroll to the bottom of the chats after new message has been
added to chat ======
function scrollToBottomOfResults() {

    var terminalResultsDiv = document.getElementById("chats");
    terminalResultsDiv.scrollTop = terminalResultsDiv.scrollHeight;
}

//============== send the user message to rasa server ================
function send(message) {

    $.ajax({
        url: "http://localhost:5005/webhooks/rest/webhook",
        type: "POST",
        contentType: "application/json",
        data: JSON.stringify({ message: message, sender: user_id }),
        success: function(botResponse, status) {
            console.log("Response from Rasa: ", botResponse, "\nStatus: ",
status);

            // if user wants to restart the chat and clear the existing chat
contents
            if (message.toLowerCase() == '/restart') {
                $("#userInput").prop('disabled', false);

                //if you want the bot to start the conversation after restart
                // action_trigger();
                return;
            }
            setBotResponse(botResponse);

        },
        error: function(xhr, textStatus, errorThrown) {

            if (message.toLowerCase() == '/restart') {
                // $("#userInput").prop('disabled', false);

                //if you want the bot to start the conversation after the
restart action.
                // action_trigger();
                // return;
            }

            // if there is no response from rasa server
            setBotResponse("");
            console.log("Error from bot end: ", textStatus);
        }
    });
}

//================== set bot response in the chats ==============
function setBotResponse(response) {
```

```
    //display bot response after 500 milliseconds
    setTimeout(function() {
        hideBotTyping();
        if (response.length < 1) {
            //if there is no response from Rasa, send  fallback message to
the user
            var fallbackMsg = "I am facing some issues, please try again
later!!!";

            var BotResponse = '<img class="botAvatar"
src="./static/img/DominguezToros.png"/><p class="botMsg">' + fallbackMsg +
'</p><div class="clearfix"></div>';

            $(BotResponse).appendTo(".chats").hide().fadeIn(1000);
            scrollToBottomOfResults();
        } else {

            //if we get response from Rasa
            for (i = 0; i < response.length; i++) {

                //check if the response contains "text"
                if (response[i].hasOwnProperty("text")) {
                    var BotResponse = '<img class="botAvatar"
src="./static/img/DominguezToros.png"/><p class="botMsg">' + response[i].text
+ '</p><div class="clearfix"></div>';
                    $(BotResponse).appendTo(".chats").hide().fadeIn(1000);
                }

                //check if the response contains "images"
                if (response[i].hasOwnProperty("image")) {
                    var BotResponse = '<div class="singleCard">' + '<img
class="imgcard" src="' + response[i].image + '">' + '</div><div
class="clearfix">';
                    $(BotResponse).appendTo(".chats").hide().fadeIn(1000);
                }


                //check if the response contains "buttons"
                if (response[i].hasOwnProperty("buttons")) {
                    addSuggestion(response[i].buttons);
                }

                //check if the response contains "attachment"
                if (response[i].hasOwnProperty("attachment")) {

                    //check if the attachment type is "video"
                    if (response[i].attachment.type == "video") {
                        video_url = response[i].attachment.payload.src;

                        var BotResponse = '<div class="video-container">
<iframe src="' + video_url + '" frameborder="0" allowfullscreen></iframe>
</div>'

$(BotResponse).appendTo(".chats").hide().fadeIn(1000);
                    }

                }
```

```
                //check if the response contains "custom" message
                if (response[i].hasOwnProperty("custom")) {

                    //check if the custom payload type is "quickReplies"
                    if (response[i].custom.payload == "quickReplies") {
                        quickRepliesData = response[i].custom.data;
                        showQuickReplies(quickRepliesData);
                        return;
                    }

                    //check if the custom payload type is "pdf_attachment"
                    if (response[i].custom.payload == "pdf_attachment") {

                        renderPdfAttachment(response[i]);
                        return;
                    }

                    //check if the custom payload type is "dropDown"
                    if (response[i].custom.payload == "dropDown") {
                        dropDownData = response[i].custom.data;
                        renderDropDwon(dropDownData);
                        return;
                    }

                    //check if the custom payload type is "location"
                    if (response[i].custom.payload == "location") {
                        $("#userInput").prop('disabled', true);
                        getLocation();
                        scrollToBottomOfResults();
                        return;
                    }

                    //check if the custom payload type is "cardsCarousel"
                    if (response[i].custom.payload == "cardsCarousel") {
                        restaurantsData = (response[i].custom.data)
                        showCardsCarousel(restaurantsData);
                        return;
                    }

                    //check if the custom payload type is "chart"
                    if (response[i].custom.payload == "chart") {

                        // sample format of the charts data:
                        // var chartData = { "title": "Leaves", "labels":
["Sick Leave", "Casual Leave", "Earned Leave", "Flexi Leave"],
"backgroundColor": ["#36a2eb", "#ffcd56", "#ff6384", "#009688", "#c45850"],
"chartsData": [5, 10, 22, 3], "chartType": "pie", "displayLegend": "true" }

                        //store the below parameters as global variable,
                        // so that it can be used while displaying the charts
in modal.
                        chartData = (response[i].custom.data)
                        title = chartData.title;
                        labels = chartData.labels;
                        backgroundColor = chartData.backgroundColor;
                        chartsData = chartData.chartsData;
                        chartType = chartData.chartType;
```

```
                        displayLegend = chartData.displayLegend;

                        // pass the above variable to createChart function
                        createChart(title, labels, backgroundColor,
chartsData, chartType, displayLegend)
                        return;
                    }

                    //check of the custom payload type is "collapsible"
                    if (response[i].custom.payload == "collapsible") {
                        data = (response[i].custom.data);
                        //pass the data variable to createCollapsible
function
                        createCollapsible(data);
                    }
                }
            }
            scrollToBottomOfResults();
        }
    }, 2000);
}

//==================================== Toggle chatbot ===============
$("#profile_div").click(function() {
    $(".profile_div").toggle();
    $(".widget").toggle();
});


//==================================== Render Pdf attachment ========
function renderPdfAttachment(data) {
    pdf_url = data.custom.url;
    pdf_title = data.custom.title;
    pdf_attachment =
        '<div class="pdf_attachment">' +
        '<div class="row">' +
        '<div class="col s3 pdf_icon"><i class="fa fa-file-pdf-o" aria-
hidden="true"></i></div>' +
        '<div class="col s9 pdf_link">' +
        '<a href="' + pdf_url + '" target="_blank">' + pdf_title + ' </a>' +
        '</div>' +
        '</div>' +
        '</div>'
    $(".chats").append(pdf_attachment);
    scrollToBottomOfResults();

}



//==================================== DropDown =================
//render the dropdown messageand handle user selection
function renderDropDwon(data) {
    var options = "";
    for (i = 0; i < data.length; i++) {
        options += '<option value="' + data[i].value + '">' + data[i].label +
'</option>'
```

```
    }
    var select = '<div class="dropDownMsg"><select class="browser-default
dropDownSelect"> <option value="" disabled selected>Choose your
option</option>' + options + '</select></div>'
    $(".chats").append(select);
    scrollToBottomOfResults();

    //add event handler if user selects a option.
    $("select").change(function() {
        var value = ""
        var label = ""
        $("select option:selected").each(function() {
            label += $(this).text();
            value += $(this).val();
        });

        setUserResponse(label);
        send(value);
        $(".dropDownMsg").remove();
    });
}

//====================================== Suggestions ==============

function addSuggestion(textToAdd) {
    setTimeout(function() {
        var suggestions = textToAdd;
        var suggLength = textToAdd.length;
        $(' <div class="singleCard"> <div class="suggestions"><div
class="menu"></div></div></diV>').appendTo(".chats").hide().fadeIn(1000);
        // Loop through suggestions
        for (i = 0; i < suggLength; i++) {
            $('<div class="menuChips" data-payload=\'' +
(suggestions[i].payload) + '\'>' + suggestions[i].title +
"</div>").appendTo(".menu");
        }
        scrollToBottomOfResults();
    }, 1000);
}

// on click of suggestions, get the value and send to rasa
$(document).on("click", ".menu .menuChips", function() {
    var text = this.innerText;
    var payload = this.getAttribute('data-payload');
    console.log("payload: ", this.getAttribute('data-payload'))
    setUserResponse(text);
    send(payload);

    //delete the suggestions once user click on it
    $(".suggestions").remove();

});

//====================== functions for drop-down menu of the bot  ==========

//restart function to restart the conversation.
$("#restart").click(function() {
```

```
    restartConversation()
});

//clear function to clear the chat contents of the widget.
$("#clear").click(function() {
    $(".chats").fadeOut("normal", function() {
        $(".chats").html("");
        $(".chats").fadeIn();
    });
});

//close function to close the widget.
$("#close").click(function() {
    $(".profile_div").toggle();
    $(".widget").toggle();
    scrollToBottomOfResults();
});

//====================================== Cards Carousel ====================

function showCardsCarousel(cardsToAdd) {
    var cards = createCardsCarousel(cardsToAdd);

    $(cards).appendTo(".chats").show();


    if (cardsToAdd.length <= 2) {
        $(".cards_scroller>div.carousel_cards:nth-of-type(" + i +
")").fadeIn(3000);
    } else {
        for (var i = 0; i < cardsToAdd.length; i++) {
            $(".cards_scroller>div.carousel_cards:nth-of-type(" + i +
")").fadeIn(3000);
        }
        $(".cards .arrow.prev").fadeIn("3000");
        $(".cards .arrow.next").fadeIn("3000");
    }


    scrollToBottomOfResults();

    const card = document.querySelector("#paginated_cards");
    const card_scroller = card.querySelector(".cards_scroller");
    var card_item_size = 225;

    card.querySelector(".arrow.next").addEventListener("click",
scrollToNextPage);
    card.querySelector(".arrow.prev").addEventListener("click",
scrollToPrevPage);


    // For paginated scrolling, simply scroll the card one item in the given
    // direction and let css scroll snaping handle the specific alignment.
    function scrollToNextPage() {
        card_scroller.scrollBy(card_item_size, 0);
    }
```

```
    function scrollToPrevPage() {
        card_scroller.scrollBy(-card_item_size, 0);
    }

}

function createCardsCarousel(cardsData) {

    var cards = "";

    for (i = 0; i < cardsData.length; i++) {
        title = cardsData[i].name;
        ratings = Math.round((cardsData[i].ratings / 5) * 100) + "%";
        data = cardsData[i];
        item = '<div class="carousel_cards in-left">' + '<img
class="cardBackgroundImage" src="' + cardsData[i].image + '"><div
class="cardFooter">' + '<span class="cardTitle" title="' + title + '">' +
title + "</span> " + '<div class="cardDescription">' + '<div class="stars-
outer">' + '<div class="stars-inner" style="width:' + ratings + '" ></div>' +
"</div>" + "</div>" + "</div>" + "</div>";

        cards += item;
    }

    var cardContents = '<div id="paginated_cards" class="cards"> <div
class="cards_scroller">' + cards + '  <span class="arrow prev fa fa-chevron-
circle-left "></span> <span class="arrow next fa fa-chevron-circle-right"
></span> </div> </div>';

    return cardContents;
}

//==================================== Quick Replies ======================

function showQuickReplies(quickRepliesData) {
    var chips = ""
    for (i = 0; i < quickRepliesData.length; i++) {
        var chip = '<div class="chip" data-payload=\'' +
(quickRepliesData[i].payload) + '\'>' + quickRepliesData[i].title + '</div>'
        chips += (chip)
    }

    var quickReplies = '<div class="quickReplies">' + chips + '</div><div
class="clearfix"></div>'
    $(quickReplies).appendTo(".chats").fadeIn(1000);
    scrollToBottomOfResults();
    const slider = document.querySelector('.quickReplies');
    let isDown = false;
    let startX;
    let scrollLeft;

    slider.addEventListener('mousedown', (e) => {
        isDown = true;
        slider.classList.add('active');
        startX = e.pageX - slider.offsetLeft;
        scrollLeft = slider.scrollLeft;
    });
```

```
    slider.addEventListener('mouseleave', () => {
        isDown = false;
        slider.classList.remove('active');
    });
    slider.addEventListener('mouseup', () => {
        isDown = false;
        slider.classList.remove('active');
    });
    slider.addEventListener('mousemove', (e) => {
        if (!isDown) return;
        e.preventDefault();
        const x = e.pageX - slider.offsetLeft;
        const walk = (x - startX) * 3; //scroll-fast
        slider.scrollLeft = scrollLeft - walk;
    });

}

// on click of quickreplies, get the value and send to rasa
$(document).on("click", ".quickReplies .chip", function() {
    var text = this.innerText;
    var payload = this.getAttribute('data-payload');
    console.log("chip payload: ", this.getAttribute('data-payload'))
    setUserResponse(text);
    send(payload);

    //delete the quickreplies
    $(".quickReplies").remove();

});

//====================================== Get User Location =================
function getLocation() {
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(getUserPosition,
handleLocationAccessError);
    } else {
        response = "Geolocation is not supported by this browser.";
    }
}

function getUserPosition(position) {
    response = "Latitude: " + position.coords.latitude + " Longitude: " +
position.coords.longitude;
    console.log("location: ", response);

    //here you add the intent which you want to trigger
    response = '/inform{"latitude":' + position.coords.latitude +
',"longitude":' + position.coords.longitude + '}';
    $("#userInput").prop('disabled', false);
    send(response);
    showBotTyping();
}

function handleLocationAccessError(error) {

    switch (error.code) {
```

```
        case error.PERMISSION_DENIED:
            console.log("User denied the request for Geolocation.")
            break;
        case error.POSITION_UNAVAILABLE:
            console.log("Location information is unavailable.")
            break;
        case error.TIMEOUT:
            console.log("The request to get user location timed out.")
            break;
        case error.UNKNOWN_ERROR:
            console.log("An unknown error occurred.")
            break;
    }

    response = '/inform{"user_location":"deny"}';
    send(response);
    showBotTyping();
    $(".usrInput").val("");
    $("#userInput").prop('disabled', false);


}

//==================================bot typing animation ================
function showBotTyping() {

    var botTyping = '<img class="botAvatar" id="botAvatar"
src="./static/img/DominguezToros.png"/><div class="botTyping">' + '<div
class="bounce1"></div>' + '<div class="bounce2"></div>' + '<div
class="bounce3"></div>' + '</div>'
    $(botTyping).appendTo(".chats");
    $('.botTyping').show();
    scrollToBottomOfResults();
}

function hideBotTyping() {
    $('#botAvatar').remove();
    $('.botTyping').remove();
}

//==================================== Collapsible ========================

// function to create collapsible,
// for more info refer:https://materializecss.com/collapsible.html
function createCollapsible(data) {
    //sample data format:
    //var
data=[{"title":"abc","description":"xyz"},{"title":"pqr","description":"jkl"}
]
    list = "";
    for (i = 0; i < data.length; i++) {
        item = '<li>' +
            '<div class="collapsible-header">' + data[i].title + '</div>' +
            '<div class="collapsible-body"><span>' + data[i].description +
'</span></div>' +
            '</li>'
        list += item;
```

```
    }
    var contents = '<ul class="collapsible">' + list + '</uL>';
    $(contents).appendTo(".chats");

    // initialize the collapsible
    $('.collapsible').collapsible();
    scrollToBottomOfResults();
}


//=================================== creating Charts ==================

//function to create the charts & render it to the canvas
function createChart(title, labels, backgroundColor, chartsData, chartType,
displayLegend) {

    //create the ".chart-container" div that will render the charts in canvas
as required by charts.js,
    // for more info. refer: https://www.chartjs.org/docs/latest/getting-
started/usage.html
    var html = '<div class="chart-container"> <span class="modal-trigger"
id="expand" title="expand" href="#modal1"><i class="fa fa-external-link"
aria-hidden="true"></i></span> <canvas id="chat-chart" ></canvas> </div> <div
class="clearfix"></div>'
    $(html).appendTo('.chats');

    //create the context that will draw the charts over the canvas in the
".chart-container" div
    var ctx = $('#chat-chart');

    // Once you have the element or context, instantiate the chart-type by
passing the configuration,
    //for more info. refer:
https://www.chartjs.org/docs/latest/configuration/
    var data = {
        labels: labels,
        datasets: [{
            label: title,
            backgroundColor: backgroundColor,
            data: chartsData,
            fill: false
        }]
    };
    var options = {
        title: {
            display: true,
            text: title
        },
        layout: {
            padding: {
                left: 5,
                right: 0,
                top: 0,
                bottom: 0
            }
        },
        legend: {
```

```
            display: displayLegend,
            position: "right",
            labels: {
                boxWidth: 5,
                fontSize: 10
            }
        }
    }

    //draw the chart by passing the configuration
    chatChart = new Chart(ctx, {
        type: chartType,
        data: data,
        options: options
    });

    scrollToBottomOfResults();
}

// on click of expand button, get the chart data from gloabl variable &
render it to modal
$(document).on("click", "#expand", function() {

    //the parameters are declared gloabally while we get the charts data from
rasa.
    createChartinModal(title, labels, backgroundColor, chartsData, chartType,
displayLegend)
});

//function to render the charts in the modal
function createChartinModal(title, labels, backgroundColor, chartsData,
chartType, displayLegend) {
    //if you want to display the charts in modal, make sure you have
configured the modal in index.html
    //create the context that will draw the charts over the canvas in the
"#modal-chart" div of the modal
    var ctx = $('#modal-chart');

    // Once you have the element or context, instantiate the chart-type by
passing the configuration,
    //for more info. refer:
https://www.chartjs.org/docs/latest/configuration/
    var data = {
        labels: labels,
        datasets: [{
            label: title,
            backgroundColor: backgroundColor,
            data: chartsData,
            fill: false
        }]
    };
    var options = {
        title: {
            display: true,
            text: title
        },
        layout: {
```

```
            padding: {
                left: 5,
                right: 0,
                top: 0,
                bottom: 0
            }
        },
        legend: {
            display: displayLegend,
            position: "right"
        },

    }

    modalChart = new Chart(ctx, {
        type: chartType,
        data: data,
        options: options
    });

}
```

File: style.css

```
body {
    font-family: 'Raleway', sans-serif;
    background: url('../img/body.png') no-repeat center center fixed;
    background-size: cover;
}


/* ============= css related to chats ============== */

.widget {
    display: none;
    width: 350px;
    right: 15px;
    height: 500px;
    bottom: 5%;
    position: fixed;
    background: #f7f7f7;
    border-radius: 10px 10px 10px 10px;
    box-shadow: 0 0px 1px 0 rgba(0, 0, 0, 0.16), 0 0px 10px 0 #00000096;
}

.chat_header {
    height: 60px;
    background: rgb(134, 0, 56);
    border-radius: 10px 10px 0px 0px;
    padding: 5px;
    font-size: 20px;

}

.chat_header_title {
```

```
    color: white;
    float: left;
    margin-top: 3%;
    margin-left: 35%;
}

.chats {
    /* display: none; */
    height: 385px;
    padding: 2px;
    border-radius: 1px;
    overflow-y: scroll;
    margin-top: 1px;
    transition: 0.2s;
}

div.chats::-webkit-scrollbar {
    overflow-y: hidden;
    width: 0px;
    /* remove scrollbar space /
    background: transparent;
    / optional: just make scrollbar invisible */
}

.clearfix {
    margin-top: 2px;
    margin-bottom: 2px;
}

.botAvatar {
    border-radius: 50%;
    width: 1.5em;
    height: 1.5em;
    float: left;
    margin-left: 5px;
    /* border: 2px solid  #2c53af ; */
}

.botMsg {
    float: left;
    margin-top: 5px;
    background: white;
    color: black;
    box-shadow: 2px 3px 9px 0px #860038;
    margin-left: 0.5em;
    padding: 10px;
    border-radius: 1.5em;
    max-width: 60%;
    min-width: 25%;
    font-size: 13px;
    word-wrap: break-word;
    border-radius: 0 20px 20px 20px;
}

textarea {
    box-shadow: none;
    resize: none;
```

```
    outline: none;
    overflow: hidden;
    font-family: Raleway;
}

textarea::-webkit-input-placeholder {
    font-family: Raleway;
}

textarea-webkit-scrollbar {
    width: 0 !important
}

.userMsg {
    animation: animateElement linear 0.2s;
    animation-iteration-count: 1;
    margin-top: 5px;
    word-wrap: break-word;
    padding: 10px;
    float: right;
    margin-right: 0.5em;
    background: #860038;
    color: white;
    margin-bottom: 0.15em;
    font-size: 13px;
    max-width: 65%;
    min-width: 15%;
    border-radius: 20px 0px 20px 20px;
    box-shadow: 0px 2px 5px 0px #860038;
}

.userAvatar {
    animation: animateElement linear 0.3s;
    animation-iteration-count: 1;
    border-radius: 50%;
    width: 1.5em;
    height: 1.5em;
    float: right;
    margin-right: 5px;
    /* border: 2px solid  #2c53af ; */
}

.usrInput {
    padding: 0.5em;
    width: 80%;
    margin-left: 4%;
    border: 0;
    padding-left: 15px;
    height: 40px;
}

.keypad {
    /* display: none;  */
    background: white;
    height: 50px;
    position: absolute;
    bottom: 0px;
```

```
    width: 100%;
    padding: 5px;
    border-radius: 0px 0px 10px 10px;
}

#sendButton {
    height: 20px;
    width: 20px;
    border-radius: 50%;
    display: inline-block;
    float: right;
    margin-right: 30px;
    text-align: center;
    padding: 5px;
    font-size: 20px;
    cursor: pointer;
    color: #860038;
}

.imgProfile {
    box-shadow: 0 0px 1px 0 rgba(0, 0, 0, 0.16), 0 0px 10px 0 #00000096;
    border-radius: 50px;
    width: 60%;
}

.profile_div {
    /* display: none; */
    position: fixed;
    padding: 5px;
    width: 10%;
    bottom: 0;
    right: 0;
    cursor: pointer;
}


/* css for dropdown menu of the bot */

#close,
#restart,
#clear {
    cursor: pointer;
}

.dropdown-trigger {
    cursor: pointer;
    color: white;
    margin-right: 5px;
    float: right;
    margin-top: 3%;
}

.dropdown-content li>a,
.dropdown-content li>span {
    color: #860038;
}
```

```
@keyframes animateElement {
    0% {
        opacity: 0;
        transform: translate(0px, 10px);
    }
    100% {
        opacity: 1;
        transform: translate(0px, 0px);
    }
}


/*========== css related to chats elements============= */


/* css for card details of carousel cards */

.content {
    transition: 0.2s;
    display: none;
    width: 350px;
    right: 15px;
    height: 500px;
    bottom: 5%;
    position: fixed;
    background: white;
    border-radius: 10px 10px 10px 10px;
    box-shadow: 0px 2px 10px 1px #860038;
}

.content_header {
    object-fit: cover;
    width: 100%;
    height: 40%;
    border-radius: 10px 10px 0px 0px;
}

.content_data {
    overflow-y: scroll;
    height: 70%;
    position: absolute;
    left: 0px;
    width: 100%;
    top: 30%;
    z-index: 1000;
    border-radius: 10px;
    background: white;
    padding: 5px;
    box-shadow: 0px -1px 20px 3px #860038;
}

.content_title {
    color: black;
    font-weight: 600;
    word-wrap: break-word;
    padding-left: 5px;
    font-size: 1.2em;
```

```
    width: 80%;
    border-radius: .28571429rem;
}

.votes {
    font-size: 12px;
    color: lightslategray;
}

.ratings {
    margin-top: 5px;
    background: #860038;
    padding: 5px;
    color: white;
    border-radius: 5px;
}

.user_ratings {
    border-radius: .28571429rem;
    color: #fff;
    font-weight: 600;
    font-size: 15px;
}

.total_ratings {
    font-size: 12px;
    opacity: .5;
    margin-left: 5px;
}

.content_data>.row .col {
    padding: 5px;
}

.metadata_1,
.metadata_2,
.metadata_3,
.metadata_4,
.row {
    margin-bottom: 0px;
}

.metadata_1 {
    color: lightslategrey;
    padding: 5px;
}

.order,
#closeContents {
    color: #860038;
}

.metadata_2,
.metadata_3,
.metadata_4,
.metadata_5,
.metadata_6 {
```

```
    color: lightslategrey;
    padding: 5px;
}

.average_cost,
.timings,
.location,
.cuisines {
    width: 70%;
    float: right;
    margin-right: 25%;
}

.fa .fa-user-o {
    font-size: 15px;
}

.stars-outer {
    display: inline-block;
    position: relative;
    font-family: FontAwesome;
}

.stars-outer::before {
    content: "\f006 \f006 \f006 \f006 \f006";
}

.stars-inner {
    position: absolute;
    top: 0;
    left: 0;
    white-space: nowrap;
    overflow: hidden;
    width: 0;
}

.stars-inner::before {
    content: "\f005 \f005 \f005 \f005 \f005";
    color: #f8ce0b;
}

div.content::-webkit-scrollbar {
    width: 0 !important
}

div.content_data::-webkit-scrollbar {
    overflow-y: hidden;
    width: 0px;
    /* remove scrollbar space /
    background: transparent;
    / optional: just make scrollbar invisible */
}


/* css for single card */

.singleCard {
```

```
    padding-left: 10%;
    padding-right: 10px;
}


/* css for image card */

.imgcard {
    object-fit: cover;
    width: 80%;
    height: 50%;
    border-radius: 10px;
    margin-left: 1%;
}


/* css for suggestions buttons */

.suggestions {
    padding: 5px;
    width: 80%;
    border-radius: 10px;
    background: #ffffff;
    box-shadow: 2px 5px 5px 1px #860038;
}

.menuTitle {
    padding: 5px;
    margin-top: 5px;
    margin-bottom: 5px;
}

.menu {
    padding: 5px;
}

.menuChips {
    display: block;
    background: #860038;
    color: #fff;
    text-align: center;
    padding: 5px;
    margin-bottom: 5px;
    cursor: pointer;
    border-radius: 15px;
    font-size: 14px;
    word-wrap: break-word;
}


/* cards carousels */

.cards {
    display: none;
    position: relative;
    max-width: 300px;
}
```

```
.cards_scroller {
    overflow-x: scroll;
    overflow-y: hidden;
    display: flex;
    height: 210px;
    width: 300px;
    transition: width 0.5s ease;
    margin-left: 5px;
    /* Enable Safari touch scrolling physics which is needed for scroll snap
*/
    -webkit-overflow-scrolling: touch;
}

.cards_scroller img {
    border-radius: 10px;
}

.cards div.note {
    position: absolute;
    /* vertically align center */
    top: 50%;
    transform: translateY(-50%);
    left: 0;
    right: 0;
    background: rgba(0, 0, 0, 0.6);
    padding: 20px;
    position: absolute;
    font-size: 4em;
    color: white;
}

.cards .arrow {
    position: absolute;
    top: 50%;
    transform: translateY(-50%);
    height: 30px;
    width: 30px;
    border-radius: 2px;
    background-position: 50% 50%;
    background-repeat: no-repeat;
    z-index: 1;
}

.cards .arrow.next {
    display: none;
    font-size: 2em;
    color: #ffffff;
    right: 10px;
}

.cards .arrow.prev {
    display: none;
    font-size: 2em;
    color: #ffffff;
    left: 10px;
}
```

```
.cards_scroller::-webkit-scrollbar {
    width: 0 !important
}

.cards_scroller>div.carousel_cards {
    margin: 5px;
    scroll-snap-align: center;
    position: relative;
    scroll-behavior: smooth;
}

.cards_scroller div.carousel_cards {
    min-width: 50%;
    min-height: 70%;
    background: #860038;
    border-radius: 10px;
}

@-webkit-keyframes fadeInLeft {
    from {
        opacity: 0;
        -webkit-transform: translatex(-10px);
        -moz-transform: translatex(-10px);
        -o-transform: translatex(-10px);
        transform: translatex(-10px);
    }
    to {
        opacity: 1;
        -webkit-transform: translatex(0);
        -moz-transform: translatex(0);
        -o-transform: translatex(0);
        transform: translatex(0);
    }
}

.in-left {
    -webkit-animation-name: fadeInLeft;
    -moz-animation-name: fadeInLeft;
    -o-animation-name: fadeInLeft;
    animation-name: fadeInLeft;
    -webkit-animation-fill-mode: both;
    -moz-animation-fill-mode: both;
    -o-animation-fill-mode: both;
    animation-fill-mode: both;
    -webkit-animation-duration: 0.8s;
    -moz-animation-duration: 0.8s;
    -o-animation-duration: 0.8s;
    animation-duration: 0.8s;
    -webkit-animation-delay: 0.8s;
    -moz-animation-delay: 0.8s;
    -o-animation-duration: 0.8s;
    animation-delay: 0.8s;
}


/* css for cards within the cards carousels */
```

```
.cardBackgroundImage {
    width: 100%;
    border-radius: 5px;
    height: 100%;
    position: absolute;
    object-fit: cover;
}

.cardFooter {
    background: rgba(56, 53, 60, 0.86);
    border-radius: 0px 0px 5px 5px;
    position: absolute;
    z-index: 3;
    color: white;
    bottom: 0;
    width: 100%;
    height: 30%;
    word-wrap: break-word;
    padding: 1px;
}

.cardTitle {
    overflow: hidden;
    white-space: nowrap;
    text-overflow: ellipsis;
    width: 90%;
    height: 40%;
    display: inline-block;
    margin-bottom: 1px;
    font-size: 15px;
    /* font-weight: 600; */
    padding: 5px;
    /* color: #2c53af; */
    color: #ffffff;
    cursor: pointer;
}

.cardDescription {
    padding: 5px;
    font-size: 13px;
    color: white;
    line-height: 15px;
}


/*css for dropDown messages*/

.dropDownMsg {
    float: left;
    margin-top: 5px;
    background: white;
    color: black;
    box-shadow: 2px 3px 9px 0px #860038;
    margin-left: 0.5em;
    padding: 10px;
    border-radius: 1.5em;
```

```
    max-width: 60%;
    min-width: 25%;
    font-size: 13px;
    word-wrap: break-word;
}

.dropDownMsg>select {
    border: 0px solid #f2f2f2;
}


/* css for quick replies */

.quickReplies {
    padding: 5px;
    position: relative;
    width: 100%;
    overflow-x: scroll;
    overflow-y: hidden;
    white-space: nowrap;
    transition: all 0.5s;
    transform: scale(0.98);
    will-change: transform;
    user-select: none;
    cursor: pointer;
    scroll-behavior: smooth;
}

.quickReplies::-webkit-scrollbar {
    width: 0 !important
}

.quickReplies.active {
    cursor: grabbing;
    cursor: -webkit-grabbing;
    transform: scale(1);
}

.chip {
    background-color: white;
    border: 1px solid #860038;
    color: #860038;
    font-weight: bolder;
    text-align: center;
}

.chip:hover {
    background-color: #860038;
    color: white;
}


/* css for charts */

#expand:hover {
    font-size: 18px;
}
```

```
#expand {
    position: absolute;
    right: 10px;
    top: 10px;
}

.modal {
    height: 60%;
    border-radius: 10px;
}

.chart-container {
    position: relative;
    margin: auto;
    height: 25vh;
    width: 20vw;
    border-radius: 10px 10px 10px 10px;
    margin-left: 5%;
    background: white;
    box-shadow: 2px 3px 9px 0px #860038;
    margin-top: 5px;
}


/* css for collapsible */

.collapsible {
    margin-top: 5px;
    margin-left: 5%;
    max-width: 60%;
    min-width: 25%;
    font-size: 13px;
    word-wrap: break-word;
    border-radius: 20px;
    background: white;
}

.collapsible-body {
    padding: 15px;
}

.collapsible-header {
    font-weight: bold;
    color: #860038;
}

ul.collapsible>li:first-child .collapsible-header {
    border-radius: 20px 20px 0px 0px;
}

ul.collapsible>li:last-child .collapsible-header {
    color: #860038;
    border-radius: 0px 0px 20px 20px;
}

ul.collapsible>li:last-child .collapsible-body {
```

```
    border-bottom: none;
}

.collapsible-header {
    padding: 10px;
}


/* css for botTyping */

.botTyping {
    float: left;
    margin-top: 5px;
    background: white;
    color: #860038;
    box-shadow: 2px 3px 9px 0px #860038;
    margin-left: 0.5em;
    padding: 15px;
    border-radius: 0 20px 20px 20px;
    max-width: 60%;
    min-width: 20%;
    word-wrap: break-word;
    border-radius: 0 20px 20px 20px;
}

.botTyping>div {
    width: 10px;
    height: 10px;
    background-color: #860038;
    border-radius: 100%;
    display: inline-block;
    -webkit-animation: sk-bouncedelay 1.4s infinite ease-in-out both;
    animation: sk-bouncedelay 1.4s infinite ease-in-out both;
    margin-right: 5px;
}

.botTyping .bounce1 {
    -webkit-animation-delay: -0.32s;
    animation-delay: -0.32s;
}

.botTyping .bounce2 {
    -webkit-animation-delay: -0.16s;
    animation-delay: -0.16s;
}

@-webkit-keyframes sk-bouncedelay {
    0%,
    80%,
    100% {
        -webkit-transform: scale(0)
    }
    40% {
        -webkit-transform: scale(1.0)
    }
}
```

```
@keyframes sk-bouncedelay {
    0%,
    80%,
    100% {
        -webkit-transform: scale(0);
        transform: scale(0);
    }
    40% {
        -webkit-transform: scale(1.0);
        transform: scale(1.0);
    }
}

input:focus {
    outline: none;
}

video:focus {
    outline: none;
}

.video-container iframe,
.video-container object,
.video-container embed {
    position: absolute;
    top: 0;
    left: 0;
    width: 90%;
    height: 100%;
    margin-left: 5%;
    border-radius: 10px;
    border-style: none;
}

link:focus {
    outline: none;
}
.link-container {
    padding: 5% !important;
    white-space: nowrap;
}

.link-container a {
    color: #860038;
}


/* Bot pop-up intro */

.tap-target {
    color: #fff;
    background: #860038;
}

.pdf_attachment {
    border: 0.5px solid #00000014;
    width: 60%;
```

```
    height: 14%;
    border-radius: 10px;
    margin-left: 10%;
    box-shadow: 2px 3px 9px 0px #860038;
}

.pdf_icon {
    border-radius: 10px 0px 0px 10px;
    height: 100%;
    font-size: 25px;
    padding: 7% !important;
    background-color: #860038;
    color: white;
}

.pdf_link {
    padding: 5% !important;
    white-space: nowrap;
    overflow: hidden;
    text-overflow: ellipsis;
}

.pdf_link a {
    color: #860038;
}
```

APPENDIX B: POSTGRESQL STATEMENTS

The database was created to mimic information for the user to view the school's data. A school API does not exist, and web scraping data would be time consuming to extract large amounts of data from the website.

==========CREATE THE DATABASE==========
CREATE DATABASE *masterdb*;

==========CREATE USER_TYPE==========
The user is created to mimic the type of student that will view the rest of the database.

```
CREATE TABLE user_type (
id serial PRIMARY KEY,
type VARCHAR (150) NOT NULL
);
```

==========CREATE SCHOOL==========
The purpose of the table "school" is to contain information used in the project.

```
CREATE TABLE school (
school_id serial PRIMARY KEY,
school_name VARCHAR (150) NOT NULL
```

```
);

================================================================================
=======================Relationship : school + user_type=======================
CREATE TABLE attends
(
  id integer NOT NULL,
  school_id integer NOT NULL,
  PRIMARY KEY (id, school_id),
  CONSTRAINT user_type_id_fkey FOREIGN KEY (id)
        REFERENCES user_type (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT school_id_fkey FOREIGN KEY (school_id)
        REFERENCES school (school_id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION
);

================================================================================
============================CREATE DEPARTMENT=============================
```

The purpose of the table “school” is to contain information used in the project.

```
CREATE TABLE department (
department_id serial PRIMARY KEY,
department_name VARCHAR (150) NOT NULL
);

================================================================================
==================Relationship: school + department=========================
CREATE TABLE belongs_to_school_depart
(
  school_id integer NOT NULL,
  department_id integer NOT NULL,
  PRIMARY KEY (school_id, department_id),
  CONSTRAINT school_id_dp_fkey FOREIGN KEY (school_id)
        REFERENCES school (school_id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION,
        CONSTRAINT department_id_sc_fkey FOREIGN KEY (department_id)
        REFERENCES department (department_id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION
);

================================================================================
=========================CREATE ENROLLMENT============================
The user can view the enrollment semester

CREATE TABLE enrollment (
enrollment_id serial PRIMARY KEY,
semester VARCHAR (150) NOT NULL
);
```

```
==========================================================================
====================Relationship: user_type+enrollment=====================

CREATE TABLE views_enrollment
(
  id integer NOT NULL,
  enrollment_id integer NOT NULL,
  PRIMARY KEY (id, enrollment_id),
  CONSTRAINT user_id_en_fkey FOREIGN KEY (id)
        REFERENCES user_type (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION,
        CONSTRAINT enrollment_id_ut_fkey FOREIGN KEY (enrollment_id)
        REFERENCES enrollment (enrollment_id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION
);

==========================CREATE COURSE==============================
The course contains the course number that belongs to the course's subject.

CREATE TABLE course (
course_id serial PRIMARY KEY,
course_number integer NULL
);

==========================================================================
====================Relationship: enrollment + course =====================

CREATE TABLE has_course
(
  enrollment_id integer NOT NULL,
  course_id integer NOT NULL,
  PRIMARY KEY (enrollment_id, course_id),
  CONSTRAINT enrollment_id_course_fkey FOREIGN KEY (enrollment_id)
        REFERENCES enrollment (enrollment_id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT course_id_en_fkey FOREIGN KEY (course_id)
        REFERENCES course (course_id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION
);

==========================CREATE SUBJECT==============================
CREATE TABLE subject (
subject_id serial PRIMARY KEY,
subject_type VARCHAR (150) NOT NULL
);

==========================================================================
====================Relationship: course + subject =====================
CREATE TABLE contains_subject
```

```
(
  course_id integer NOT NULL,
  subject_id integer NOT NULL,
  PRIMARY KEY (course_id, subject_id),
  CONSTRAINT course_id_sub_fkey FOREIGN KEY (course_id)
          REFERENCES course (course_id) MATCH SIMPLE
          ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT subject_id_cs_fkey FOREIGN KEY (subject_id)
          REFERENCES subject (subject_id) MATCH SIMPLE
          ON UPDATE NO ACTION ON DELETE NO ACTION
);



==========CREATE CLASS==========
CREATE TABLE class (
class_id serial PRIMARY KEY,
class_number INTEGER NULL
);

================================================================================
==========Relationship: subject + class ==========
CREATE TABLE offers_class
(
  subject_id integer NOT NULL,
  class_id integer NOT NULL,
  PRIMARY KEY (subject_id, class_id),
  CONSTRAINT subject_id_cls_fkey FOREIGN KEY (subject_id)
          REFERENCES subject (subject_id) MATCH SIMPLE
          ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT class_id_sub_fkey FOREIGN KEY (class_id)
          REFERENCES class (class_id) MATCH SIMPLE
          ON UPDATE NO ACTION ON DELETE NO ACTION
);


==========CREATE PROFESSOR==========
Create the professor to have a student view.

CREATE TABLE professor (
professor_id serial PRIMARY KEY,
fname VARCHAR (150) NULL,
lname VARCHAR (150) NULL,
email VARCHAR (200) NULL
);

================================================================================
==========Relationship: class + professor==========
CREATE TABLE teaches
(
```

```
  professor_id integer NOT NULL,
  class_id integer NOT NULL,
  PRIMARY KEY (professor_id, class_id),
  CONSTRAINT professor_id_cl_fkey FOREIGN KEY (professor_id)
          REFERENCES professor (professor_id) MATCH SIMPLE
          ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT class_id_prof_fkey FOREIGN KEY (class_id)
          REFERENCES class (class_id) MATCH SIMPLE
          ON UPDATE NO ACTION ON DELETE NO ACTION
);


=========================================================================
==================Relationship: professor + department=====================

CREATE TABLE belongs_to_department
(
  professor_id integer NOT NULL,
  department_id integer NOT NULL,
  PRIMARY KEY (professor_id, department_id),
  CONSTRAINT professor_id_dt_fkey FOREIGN KEY (professor_id)
          REFERENCES professor (professor_id) MATCH SIMPLE
          ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT department_id_pf_fkey FOREIGN KEY (department_id)
          REFERENCES department (department_id) MATCH SIMPLE
          ON UPDATE NO ACTION ON DELETE NO ACTION
);


==========================CREATE PROGRAM==========================
CREATE TABLE program (
program_id serial PRIMARY KEY,
program_type VARCHAR (150) NOT NULL
);

=========================================================================
==================Relationship: department + program=====================

CREATE TABLE contains_program
(
  department_id integer NOT NULL,
  program_id integer NOT NULL,
  PRIMARY KEY (department_id, program_id),
  CONSTRAINT department_id_pf_fkey FOREIGN KEY (department_id)
          REFERENCES department (department_id) MATCH SIMPLE
          ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT program_id_dp_fkey FOREIGN KEY (program_id)
          REFERENCES program (program_id) MATCH SIMPLE
          ON UPDATE NO ACTION ON DELETE NO ACTION
);
```

```
==========CREATE TIME==========

CREATE TABLE class_time (
class_time_id serial PRIMARY KEY,
day_time VARCHAR (150) NULL,
start_time TIME NULL,
end_time TIME NULL
);

==========Relationship: class + time==========

CREATE TABLE meets
(
  class_id integer NOT NULL,
  class_time_id integer NOT NULL,
  PRIMARY KEY (class_id, class_time_id),
  CONSTRAINT class_id_time_fkey FOREIGN KEY (class_id)
        REFERENCES class (class_id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT class_time_fkey FOREIGN KEY (class_time_id)
        REFERENCES class_time (class_time_id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION
);

==========CREATE FAFSA==========

CREATE TABLE fafsa (
fafsa_id serial PRIMARY KEY,
fafsa_type VARCHAR (150) NOT NULL
);

==========Relationship: school + fafsa==========

CREATE TABLE offers_fafsa
(
  school_id integer NOT NULL,
  fafsa_id integer NOT NULL,
  PRIMARY KEY (school_id, fafsa_id),
  CONSTRAINT school_id_fafsa_fkey FOREIGN KEY (school_id)
        REFERENCES school (school_id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fafsa_id_school_fkey FOREIGN KEY (fafsa_id)
        REFERENCES fafsa (fafsa_id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION
);
```

```
==========CREATE INFORMATION==========

CREATE TABLE information (
info_id serial PRIMARY KEY,
description VARCHAR (600) NOT NULL
);

==========Relationship: school + information==========
CREATE TABLE contains_info
(
  school_id integer NOT NULL,
  info_id integer NOT NULL,
  PRIMARY KEY (school_id, info_id),
  CONSTRAINT school_id_info_fkey FOREIGN KEY (school_id)
        REFERENCES school (school_id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT info_id_fkey FOREIGN KEY (info_id)
        REFERENCES information (info_id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION
);


==========CREATE STEP INFORMATION ==========
CREATE TABLE steps_info (
steps_id serial PRIMARY KEY,
step_name VARCHAR (250) NULL,
step_description VARCHAR (600) NULL
);

==========Relationship: information + steps
info==========
CREATE TABLE views_steps
(
  school_id integer NOT NULL,
  steps_id integer NOT NULL,
  PRIMARY KEY (school_id, steps_id),
  CONSTRAINT school_id_steps_fkey FOREIGN KEY (school_id)
        REFERENCES school (school_id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT step_info_fkey FOREIGN KEY (steps_id)
        REFERENCES steps_info (steps_id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION
);

==========CREATE CAMPUS==========

CREATE TABLE campus (
campus_id serial PRIMARY KEY,
```

```
campus_type VARCHAR (250) NULL
);

=========================================================================
====================Relationship: school + campus ============================

CREATE TABLE shows_campus
(
  school_id integer NOT NULL,
  campus_id integer NOT NULL,
  PRIMARY KEY (school_id, campus_id),
  CONSTRAINT school_id_steps_fkey FOREIGN KEY (school_id)
        REFERENCES school (school_id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT campus_fkey1 FOREIGN KEY (campus_id)
        REFERENCES campus (campus_id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION
);

====================CREATE CAMPUS EMPLOYMENT=======================
CREATE TABLE campus_employment (
campus_employ_id serial PRIMARY KEY,
employ_description VARCHAR (600) NULL
);

=========================================================================
====================Relationship: campus + employment=======================
CREATE TABLE contains_employment
(
  campus_id integer NOT NULL,
  campus_employ_id integer NOT NULL,
  PRIMARY KEY (campus_id, campus_employ_id),
  CONSTRAINT campus_fkey1 FOREIGN KEY (campus_id)
        REFERENCES campus (campus_id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT campus_employment_fkey FOREIGN KEY (campus_employ_id)
        REFERENCES campus_employment (campus_employ_id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION
);

====================CREATE CAMPUS TOURS============================

CREATE TABLE campus_tours (
campus_tour_id serial PRIMARY KEY,
campus_desc VARCHAR (600) NULL
);

=========================================================================
====================Relationship: campus + tours ============================
CREATE TABLE provides_tours
```

```
(
  campus_id integer NOT NULL,
  campus_tour_id integer NOT NULL,
  PRIMARY KEY (campus_id, campus_tour_id),
  CONSTRAINT campus_fkey2 FOREIGN KEY (campus_id)
        REFERENCES campus (campus_id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT campus_tours_fkey FOREIGN KEY (campus_tour_id)
        REFERENCES campus_tours (campus_tour_id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION
);
```

APPENDIX C: DATABASE RELATIONSHIP TABLES

1. Attends table is used with school + usertype, by which the user_type attends school. This is a many to one relationship since many students go to one school.

Table: attends

id [PK] integer	school_id [PK] integer
1	1
2	1
3	1
4	1

2. Belongs_to_school_depart is the relationship between school and department with the cardinality many to one relationship, since many departments belong to one school.

Table: belongs_to_school_depart

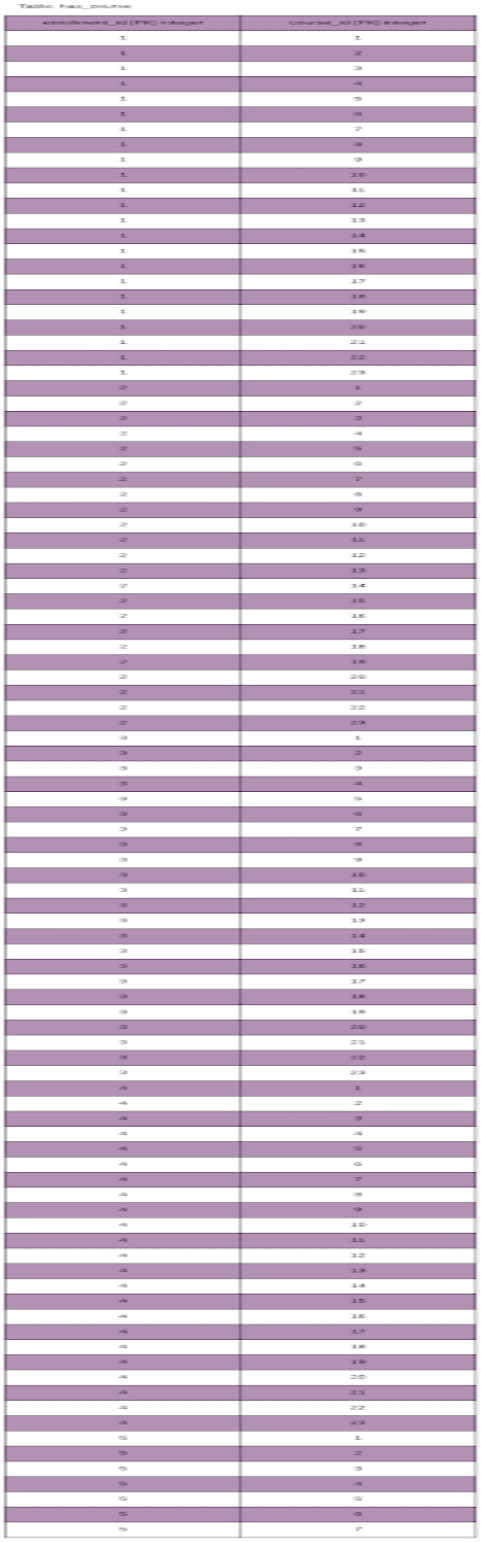
school_id [PK] integer	department_id [PK] integer
1	1
1	2
1	3
1	4
1	5
1	6

3. Views_enrollment is the relationship between enrollment and user_type where many user_types view enrollment semesters.

Table: views_enrollment

id [PK] integer	enrollment_id [PK] integer
1	1
1	2
1	3
1	4
1	5
1	6
2	1
2	2
2	3
2	4
2	5
2	6
3	1
3	2
3	3
3	4
3	5
3	6
4	1
4	2
4	3
4	4
4	5
4	6
5	1
5	2
5	3
5	4
5	5
5	6
6	1
6	2
6	3
6	4
6	5
6	6

4. Has_course is the relationship between enrollment and course where enrollment has the course.



5. Contains_subject is the relationship between course and subject, where the course contains the subject.

Table: contains_subject

course_id [PK] integer	subject_id [PK] integer
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1
12	1
12	2
13	1
14	1
15	1
16	1
17	1
18	1
19	1
20	1
21	1
22	1
23	1

6. Offers_class is the relationship between subject and class, where the subject offers the classes.

Table: offers_class

subject_id [PK] integer	class_id [PK] integer
1	1
1	2
1	3
1	4
1	5
1	6
1	7
1	8
1	9
1	10
1	11
1	12
1	13
1	14
1	15
1	16
1	17
1	18
1	19
1	20
1	21
1	22
1	23
1	24
1	25
1	26
1	27
2	28

7. Teaches is the relationship between the class and professor, where the professor teaches multiple classes.

Table: teaches

professor_id [PK] integer	class_id [PK] integer
1	1
1	9
2	2
3	3
4	4
4	7
5	5
6	6
6	10
7	11
8	8
8	12
9	13
9	20
10	14
10	18
10	21
11	15
12	16
12	23
12	24
12	26
13	27
14	17
14	25
15	19
16	22
17	28

8. Belongs_to_department contains the professor and department where the professor belongs to many departments.

Table: belongs_to_department

professor_id [PK] integer	department_id [PK] integer
1	6
2	6
3	6
4	6
5	6
6	6
7	6
8	6
9	6
10	6
11	6
12	6
13	6
14	6
15	6
16	6
17	6

9. Contains_program is the relationship between program and department where the department contains multiple programs.

Table: contains_program

department_id [PK] integer	program_id [PK[integer
6	1
6	2

10. Meets is the relationship between class_time and class because the class meets at a class_time.

Table: meets

class_id [PK] integer	class_time_id [PK] integer
1	1
2	2
3	3
4	4
5	11
6	5
7	11
8	6
9	11
10	6
11	7
12	7
13	8
14	9
15	10
16	11
17	12
18	13
19	1
20	8
21	9
22	14
23	15
24	1
25	16
26	3
27	1
28	1

11. Offers_fafsa is the relationship between school and fafsa, since fafsa provides the school with the fafsa programs.

Table: offers_fafsa

school_id [PK] integer	fafsa_id [PK] integer
1	1
1	2
1	3

12. Show_campus is the relationship between school and campus since the campus extends to the rest of the campus tables.

Table: shows_campus

school_id [PK] integer	campus_id [PK] integer
1	1
1	2
1	3

13. Provides_tours is the relationship between campus and campus_tours that provides the user with campus tour details.

Table: provides_tours

campus_id [PK] integer	campus_tour_id [PK] integer
2	1
2	2
2	3
2	4
2	5
2	6

14. Contains_employment is the relationship between the campus and the campus_employement that provides employment information that redirects the student to the resources.

Table: contains_employment

campus_id [PK] integer	campus_employ_id [PK] integer
1	1
1	2
1	3
1	4
1	5
1	6
1	7
1	8
1	9
1	10
1	11
1	12
1	13
1	14
1	15
1	16
1	17
1	18
1	19
1	20
1	21
1	22

APPENDIX D: GLOSSARY

1. NLP: Natural Language Processing: Subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human.
2. RASA X: Toolset that helps improve conversations between human and bot.
3. Machine Learning: Study of computer algorithms that improve automatically through experience.
4. Artificial Intelligence: The theory and development of computer systems able to perform tasks that normally require human intelligence, such as visual perception, speech recognition, decision-making, translation between languages.
5. Rasa NLU: Open source natural language processing tool for intent classification, response retrieval, and entity extraction in chat-bots.
6. Rasa Core: Is a dialogue engine for building AI assistant. Handles conversation flow, utterances, and actions.
7. Intents: Describe how the user messages should be categorized.
8. Slots: Virtual assistance memory
9. Domain: Defines the intents, entities, slots and actions in Rasa.
10. Actions: The things the bot runs in response to user input.
11. Tracker-Store: Conversations stored within the tracker store.
12. Microservices Architecture: Architectural style that structures an application as a collection of services.
13. Web-socket: Computer communications protocol, providing full-duplex communication channels over a single TCP connection.

14. Chat-Bot: Computer program to simulate conversation with human users, specially over the internet.