12-2018

California State University, San Bernardino Chatbot

Krutarth Desai
005452196@coyote.csusb.edu

CALIFORNIA STATE UNIVERSITY, SAN BERNARDINO CHATBOT

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

---

by

Krutarth Desai

December 2018

CALIFORNIA STATE UNIVERSITY, SAN BERNARDINO CHATBOT

A Project

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Krutarth Desai

December 2018

Approved by:

Dr. Tong Lai Yu, Advisor, School of Computer Science and Engineering

Dr. Ernesto Gomez, Committee Member

Dr. Owen Murphy, Committee Member

ABSTRACT

Now-a-days the chatbot development has been moving from the field of Artificial-Intelligence labs to the desktops and mobile domain experts. In the fastest growing technology world, most smartphone users spend major time in the messaging apps such as Facebook messenger. A chatbot is a computer program that uses messaging channels to interact with users using natural Languages. Chatbot uses appropriate mapping techniques to transform user inputs into a relational database and fetch the data by calling an existing API and then sends an appropriate response to the user to drive its chats. Drawbacks include the need to learn and use chatbot specific languages such as AIML (Artificial Intelligence Markup Language), high botmaster interference, and the use of non-matured technology. In this project, Facebook messenger based chatbot is proposed to provide domain independent, an easy to use, smart, scalable, dynamic and conversational agent in order to get information about CSUSB. It has the unique functionalities which identify user interactions made by their natural language, and the flawless support of various application domains. This provides an ample of unique scalabilities and abilities that will be evaluated in the future phases of this project.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor Dr. Tong Lai Yu, for all his guidance and help during the project. His patient guidance, enthusiastic encouragement, and scholarly advice have helped me to a great extent to accomplish this research work. He is not only a professor for me, he is more like a close friend and a respected person that worth learning in life.

I would also like to extend my thanks to Dr. Ernesto Gomez and Dr. Owen Murphy for being the committee members. Thank you for your valuable advice and support.

I am also very thankful for the help of the Department of Computer Science at California State University, San Bernardino. Especially, my graduate advisor Dr. Ernesto Gomez, who gave me great help during the graduate study.

Finally, I want to thank my family for the support and encouragement throughout my study.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

What is Chatbot

Chatbots are computer algorithms that interact with humans using a conversational interface [1]. Chatbots satisfy user requirements by replying to questions in a simpler yet efficient way. They are user-friendly and always available for the user when needed. Chatbots are designed to simulate an interaction with another human. In the fast-growing technologically-advanced world, most smartphone users spend major time in messaging apps such as Facebook messenger. That gives great opportunity to create interactive chatbots that understand user questions and provide them with the answers. Chatbots are great for business growth, upsells and marketing purposes.

A chatbot is a computer program that interacts with a human through a chat interface and is designed to simulate a human [2]. The chatbot systems employ simply a dialogue system based on a natural language. Therefore, they can be used as interfaces for a vast number of applications including entertainment applications, educational applications, e-learning platforms, research engines, and ecommerce web-site navigations.

Background

In the current world, there are a lot of voice-based personal assistants available. Siri designed by Apple, Google home, Alexa by Amazon, Cortana by Microsoft are technically chatbots. They are chat-based conversational agents. We can chat with a bot the same way as we speak to a friend or a coworker, and it responds back in a human-like tone that demonstrates the personality that the bot creator has structured.

Machines are learning the art of human expressions and conversations. Algorithms and technologies are filling the gap in human computer interaction. Joseph Weisenbaum made the very first chatbot in 1966. It was called ELIZA [3]. ELIZA was made to recognize human interaction by using simple pattern recognition. It was released to administrative staff in the guise of a psychiatrist. People thought that it was a real doctor and Weisenbaum was shocked to see how humans behaved with the computer program. Users got emotional while chatting with ELIZA, which was incredible. From ELIZA, computer scientists learn that humans have a desire to communicate with a computer in much the same way as they do with another human. Facebook launched the messenger platform in 2016. As a result, other companies started heavy investments in messenger bots, artificial intelligence and machine learning. Bots are created in different industries such as news, weather, shopping, retail and much more. Bots serve a lot of user traffic in the current world [5].

A messenger bot uses pattern matching or Natural Language Processing to parse user inputs. The bots use Node.js, Python or Java to parse user data and to log, analyze and send answers back to the user. Bots are deployed to web servers and fulfil users' requests all the time [6]

CHAPTER TWO

PROJECT OVERVIEW

Objective

The objective of this project is to create a messenger bot using the Facebook messenger platform. The bot would be available online all the time which can facilitate users' questions and inquiries regarding Professors and Courses of Computer Science and Engineering department of California State University, San Bernardino. A user may ask questions about professors to check their office hours, email, phone, office location and list of all the professors of Computer Science and Engineering Department. A user may also ask questions about Computer Science courses to check class location, room number, lab hours, course name or title, number of units and course instructor. The user is allowed to ask about the course catalog to know about what classes are offered in a quarter. The bot would listen to user's questions on Facebook messenger and answers accordingly. A bot understands the intention of the users and replies back in human language. The users do not have to search on the website to get the information about professors and courses of California State University, San Bernardino. The messenger bot is rapid, innovative and human friendly.

The NLP module is responsible for processing user input in a way that facilitates the mission of getting the needed answer. The pattern matching dictionary consists of patterns which find the intent of the question.

Technologies

I intend to create a messenger bot using the Facebook messenger platform. The bot app will be built on a Node.js module that serves as an interface to the messenger platform. I will create a REST APIs for CSUSB in Java to fetch the data from the database and it will be hosted in Microsoft Azure. The data will be stored in a MySQL database. The bot code in Node.js will be deployed on the Heroku server. The admin portal will be created in C#, ASP.Net to add professors and course details which will be deployed in Microsoft Azure.

I am going to use xregexp Node.js library for regular expression to create the RegEx pattern dictionary, which extracts the actionable piece of data from the given sentence to process further. I am going to use Windows 10 as the operating system.

I am going to use IntelliJ Idea to create node.js app for bot. I will use Microsoft Visual Studio 2013 to write a C#, ASP.Net for building an admin portal. I will use Eclipse to create RestAPIs in Java. Below is the advantage of choosing Node.Js for my project.

- It is faster than other languages.
- It almost never blocks.

- It yields great concurrency.
- Everything is asynchronous.
- It offers a unified programming language and data type.

It is highly scalable.

CHAPTER THREE

INTRODUCTION TO REGULAR EXPRESSIONS

What is Regular Expression

A Regular Expression or RegEx are an extremely powerful way of identifying patterns in strings [9]. A RegEx is available across most computer languages with flavor specific variations. The usage of RegEx is mentioned below. [9]

- Validating passwords, email address, IP address, pin codes, phone number, credit card numbers and more.
- Turning all email id and URLs in a blog post to valid hyperlinks.
- Searching and replacing text based on a given pattern.
- Finding / renaming files based on a given pattern.

Principles of Regular Expression

A Regular Expression begins and ends with a Forward Slash (“/”). This performs like a container which will hold a pattern. The next thing is a pattern itself which sets inside those forward slashes. Lastly, it has optional flags which can be used to enable things like case insensitive search among other things. [10]

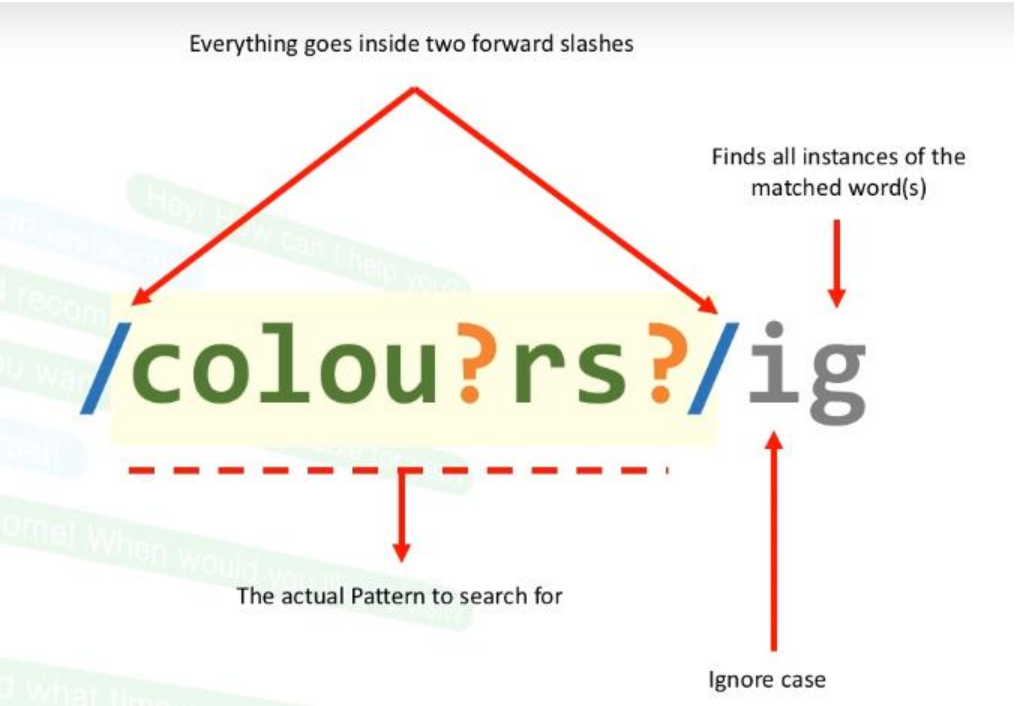


Figure 1. Regular Expression

Figure 1 shows a format of Regular Expression. A word “colours” is an actual pattern where “?” is an optional quantifier. Over here characters “U” and “S” are optional which means a word “colours” may also be written like “color”. Finally, we have optional flags “i” and “g” at the end. A flag “i” stands for Ignore Case while a flag “g” stands for “Global” which means it should find all instances of the word that matches in a given data set. So pattern in figure 1 matches word like “color”, ”colour”, ”colors”, ”colours”, ”CoLouRs”, ”CoLoRs”, “COLORS”. [10]

In order to build and check patterns you can use RegExr which is online tool to help you learn, test and build Regular Expressions. Here is a link

https://www.regexr.com to use this tool. Figure 2 is the picture of RegExr Tool which identifies a set of pattern from given paragraph.

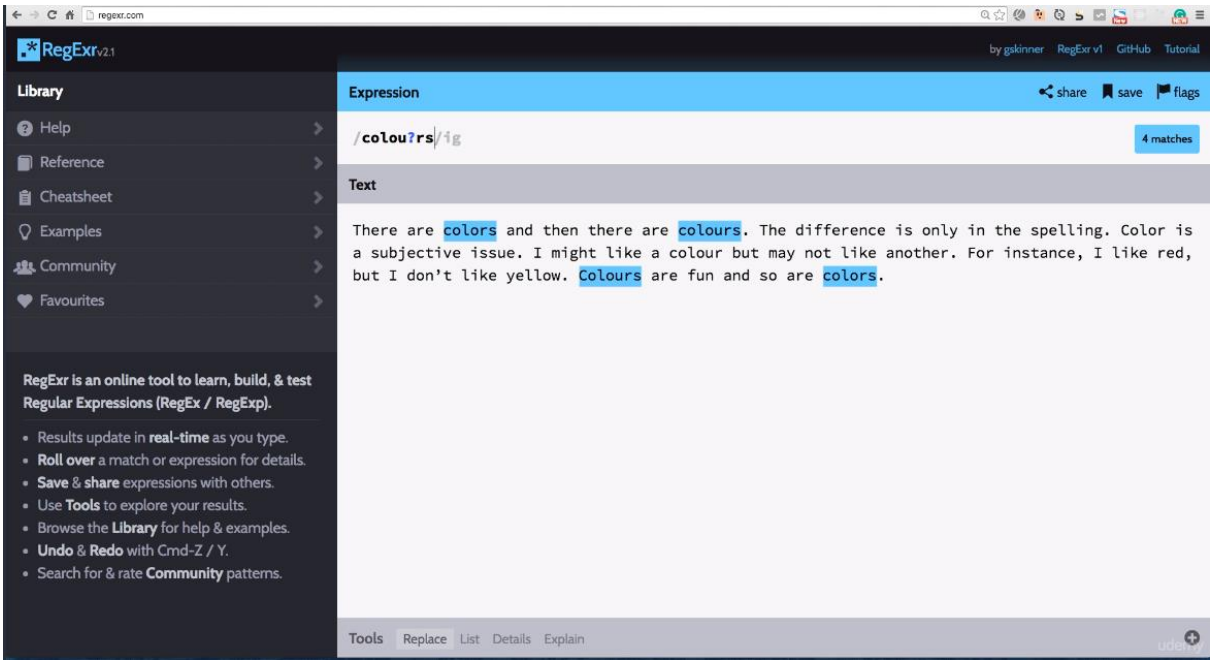


Figure 2. RegExr Tool

Let's move forward with another quantifier which is "\b". Here, "\b" stands for Boundary which expects a word isolated on both side by an empty space. A word boundary instructs a RegEx that given a word is an isolated word surround by an empty space but not the part of any word. Figure 3 shows an example of a word boundary using "/b". In a given example there is a pattern to match a word "the" which is enclosed by "/b". Given pattern will match a word "the" only if it is an isolated word but not a part of any word. In a given paragraph word "the" is a part of a word "there" but it won't be matched because a given pattern is enclosed by "/b". [10]

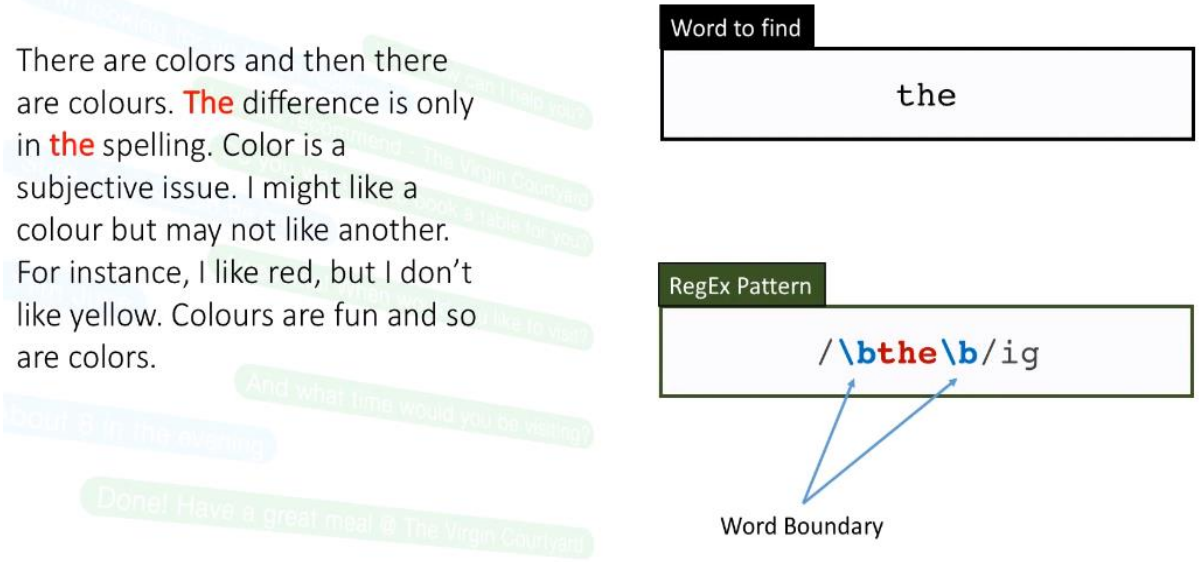


Figure 3. Word Boundary

Let's take a look of another quantifier "|" which is known as "OR" quantifier. Since we have two words to match, it will match all the words which are separated by "|". Figure 4 is an example of how "|" quantifier works to match given set of words. In this example a user wants to match words "hot" and "cold" at the same time. Both words are separated by "|". [10]

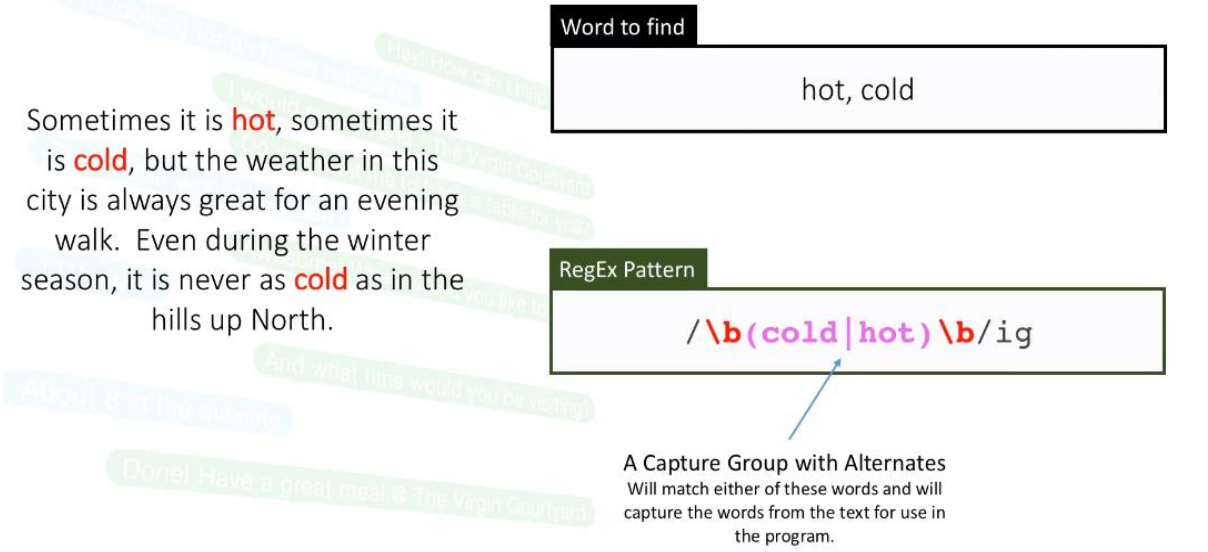


Figure 4. OR Quantifier

Let's move further with another quantifier "\d" which stands for a Digit. "\d" is used to match digits from 0 to 9. In order to match a word with white space "\s" is being used. "\s" will match words separated by a white space. Figure 5 is an example of a pattern for matching a zip code (pin code) of a city. It shows a pattern which is matched for zip code with and without an empty space. [10]

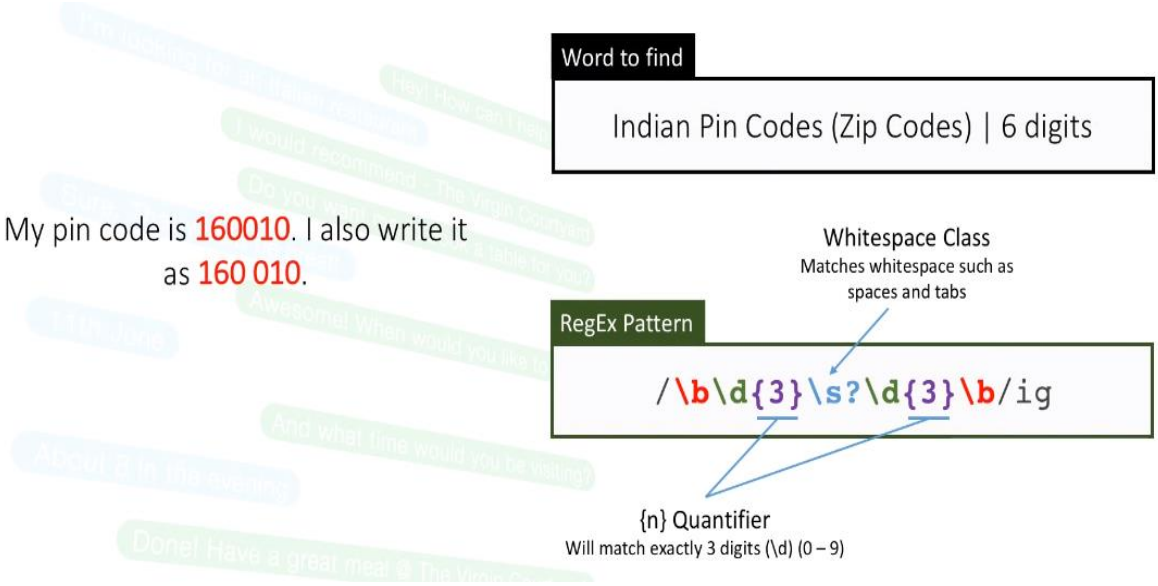


Figure 5. Pattern of Zip Code

Finally, our last example is to match an email addresses from a given paragraph. It will start with a word boundary “\b”. Let’s first focus to the username part of the email id which appears before @ sign. A username may contents alphabets from A to Z which is defined as “[a-z]”, numbers from 0 to 9 which is defined as [0-9] and a symbol of a Hyphen (-) or an Under Score (_) which are followed by “\b”. This set of characters could be more than one, so “+” quantifier will be used to match all of these. In order to put this in definable pattern, let’s now place @ sign followed by a back slash. Now for the domain, it should contain alphabets and numbers. After then we will add a dot for the top level domain such as .com, .net, .co.us and we will use an “{n}” quantifier which allows us to set a number of characters from 2 to 24. The largest top level domain is

about 24 characters long. Let's take a look of figure 3.2.6 to understand a pattern of email address. [10]

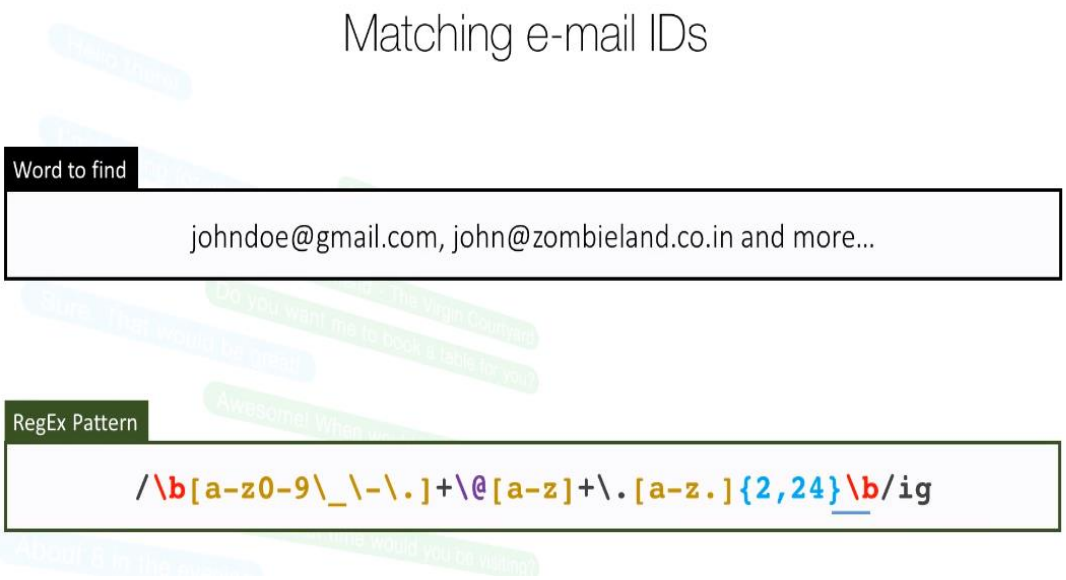


Figure 6. Pattern of Email Address

In order to test this pattern I will use online tool RegExr which is mentioned in figure 7. In this figure you can see it only selects Email Ids as expected.

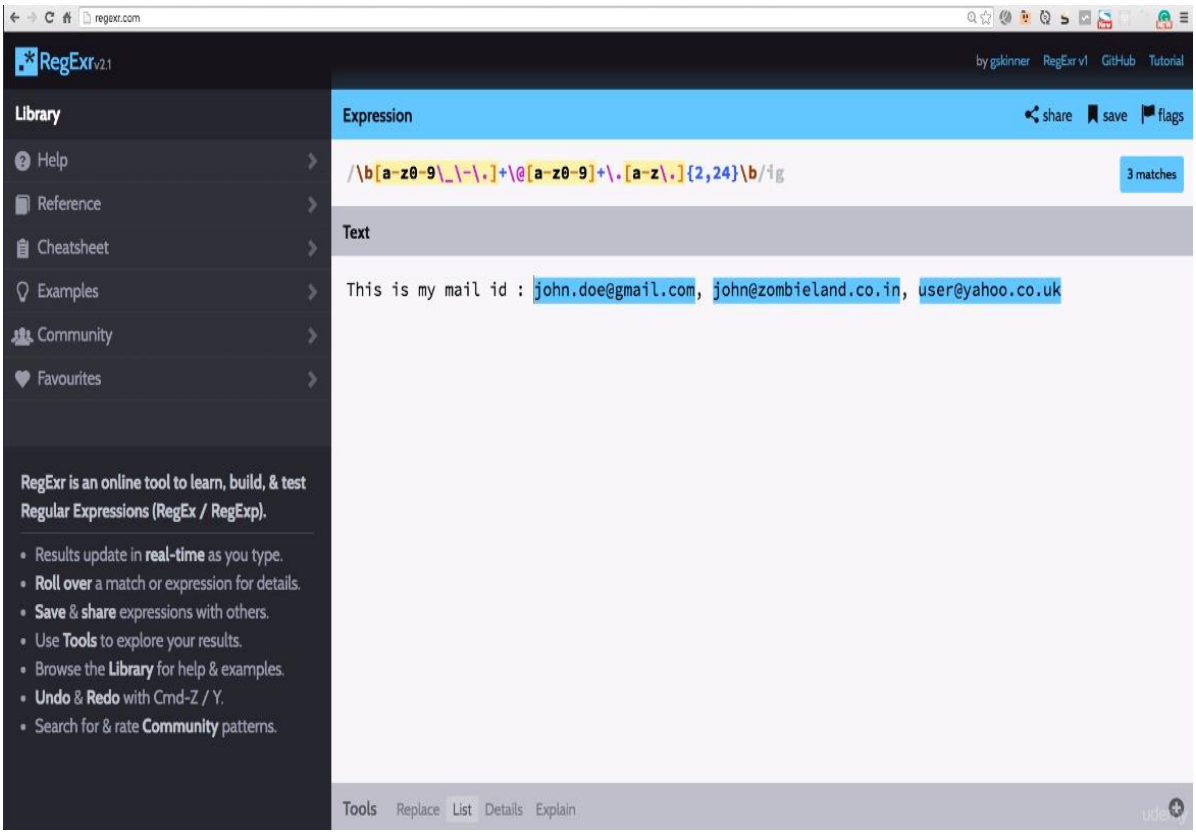


Figure 7. Email Address in RegExr Tool

CHAPTER FOUR

CHATBOT MECHANISM

Anatomy of Chatbot

The first piece in a chatbot anatomy is the human. The human is the most important component here because he is the user who has to be impressed by how the bot performs. A user needs a channel to interact on. This could be Facebook Messenger, WeChat, Skype or Slack as well. Messages that user types in are then sent by this channels to the chatbot. Here, the chatbot is an application that expects a user messages and begins a conversation session. A session is an active as long as the current set of messages make up for an actively persuade conversation. These messages are then sent to a parser. A parser job is to process user's message, sent in a human language such as an English and then to convert it into structured data that can be consumed. In this case most system will produce two kinds of data elements. The first is called Intent. Intent suggests what the user intends to do. [11]

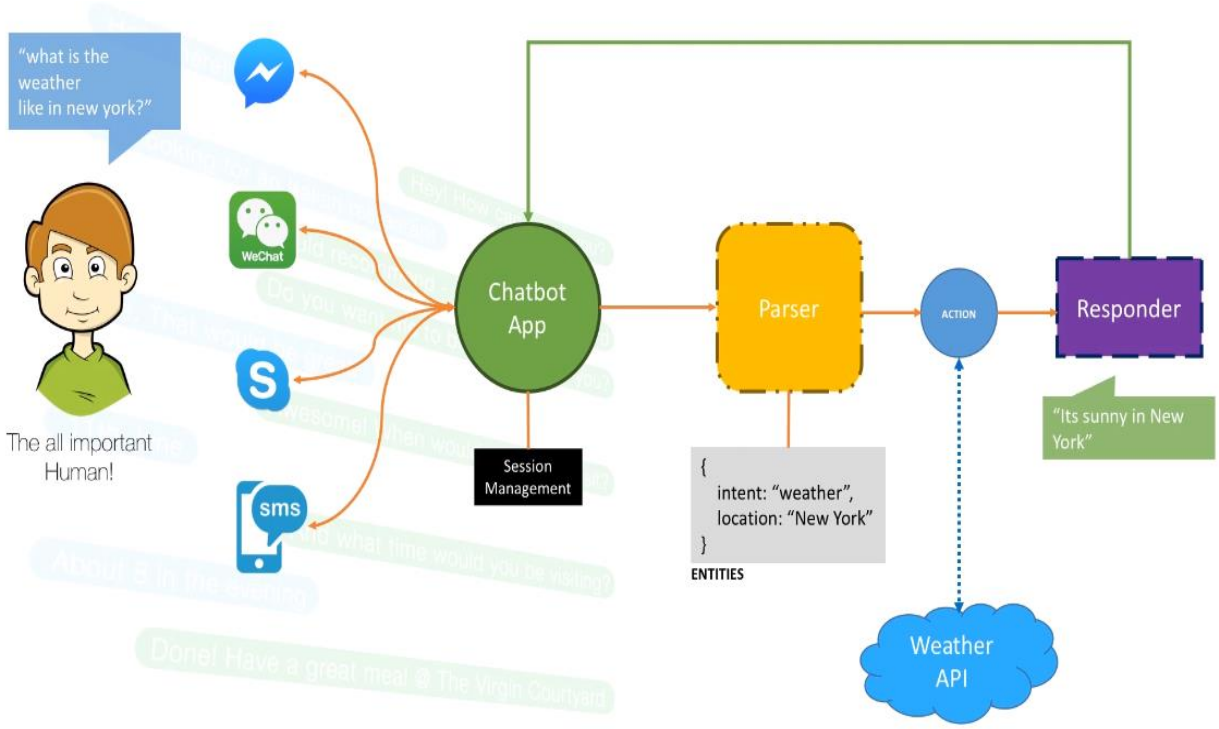


Figure 8. Chatbot Anatomy

A figure 8 shows how essentially a chatbot works. For instance, in the sentence “What is the weather like in New York?” a user intends to know about the weather. This is the first piece of data that parser must compute. After this is done, parser also needs to extract actionable data. So in the given example, actionable piece of data is set to New York. This actionable data is called Entities which coupled with Intent are required to produce a suitable response to process data. A function designed to handle this intent can then take the location entity and call a third party API to fetch weather data and send it back to the user. Once the data are processed, or fetched from an API, the next anatomical

component is the Responder. A Responder takes the data and produces a human like output and sends back the response to a channel from where it came and onward to a user. [11]

Let's take an example of one more question of a project. In figure 9 a user asks question "Who is the instructor of CSE 202 in fall 2018?" Let's examine this question. Most people are not expected to be a grammatical genius when they are in a phone typing a message. So this question may also be asked like "Who teaches CSE 202 in fall 2018?" In both cases intent of the user is the same but sentence formation is a bit different. But we just emphasize on the keywords but not in sentence formation.

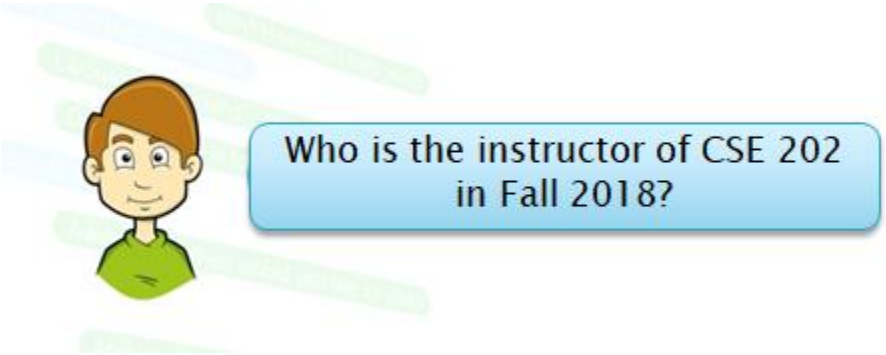


Figure 9. User's Question

Figure 10 shows that in the given sentence there are three actionable pieces of data that we want to recognize and extract which are "Instructor", "CSE 202" and "Fall 2018". Words "Instructor" and "Teaches" refer to the professor's name, a word "CSE 202" refers to a course name and a word "Fall 2018" refers to the specific quarter. These keywords represent as Entity. In the given

sentence, Intent is to know about professor's name for given course that user wants to check for.

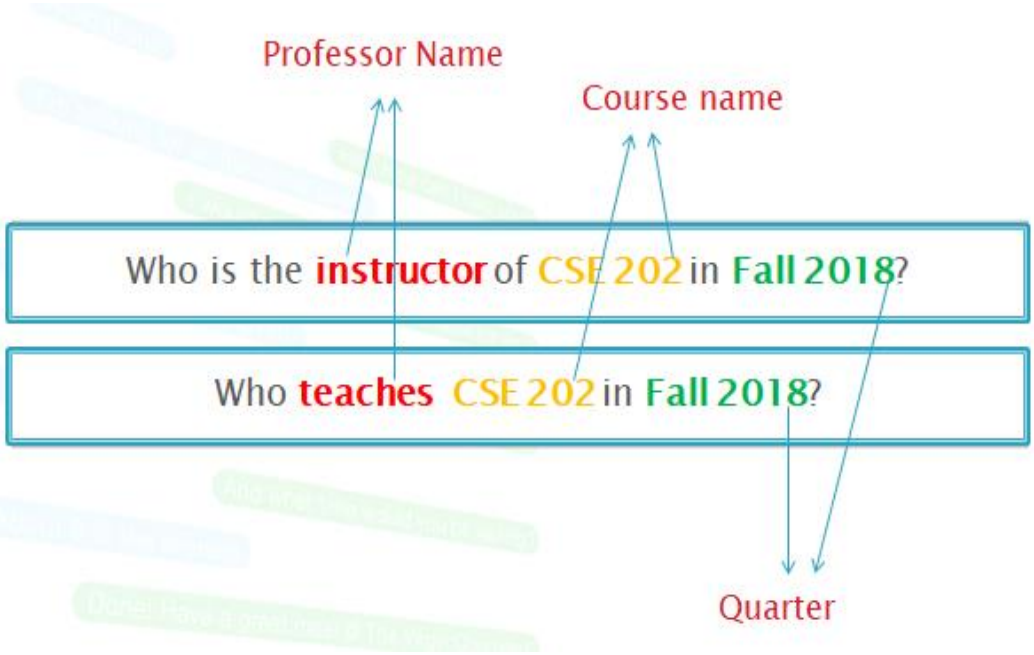


Figure 10. Keywords Extraction

Figure 11 shows this is how a bot is going to function in a nutshell. Once a sentence goes into a bot, very first step is attempt to match a pattern from given set of rules. Once a pattern is matched, it goes on to extract intent and entities from the sentence. Once it gets an actionable piece of data, it proceeds further by calling a Course API to fetch an actual data which are stored in a database. Once we get response from an API, we parse and process the data to create a suitable reply for user to read.

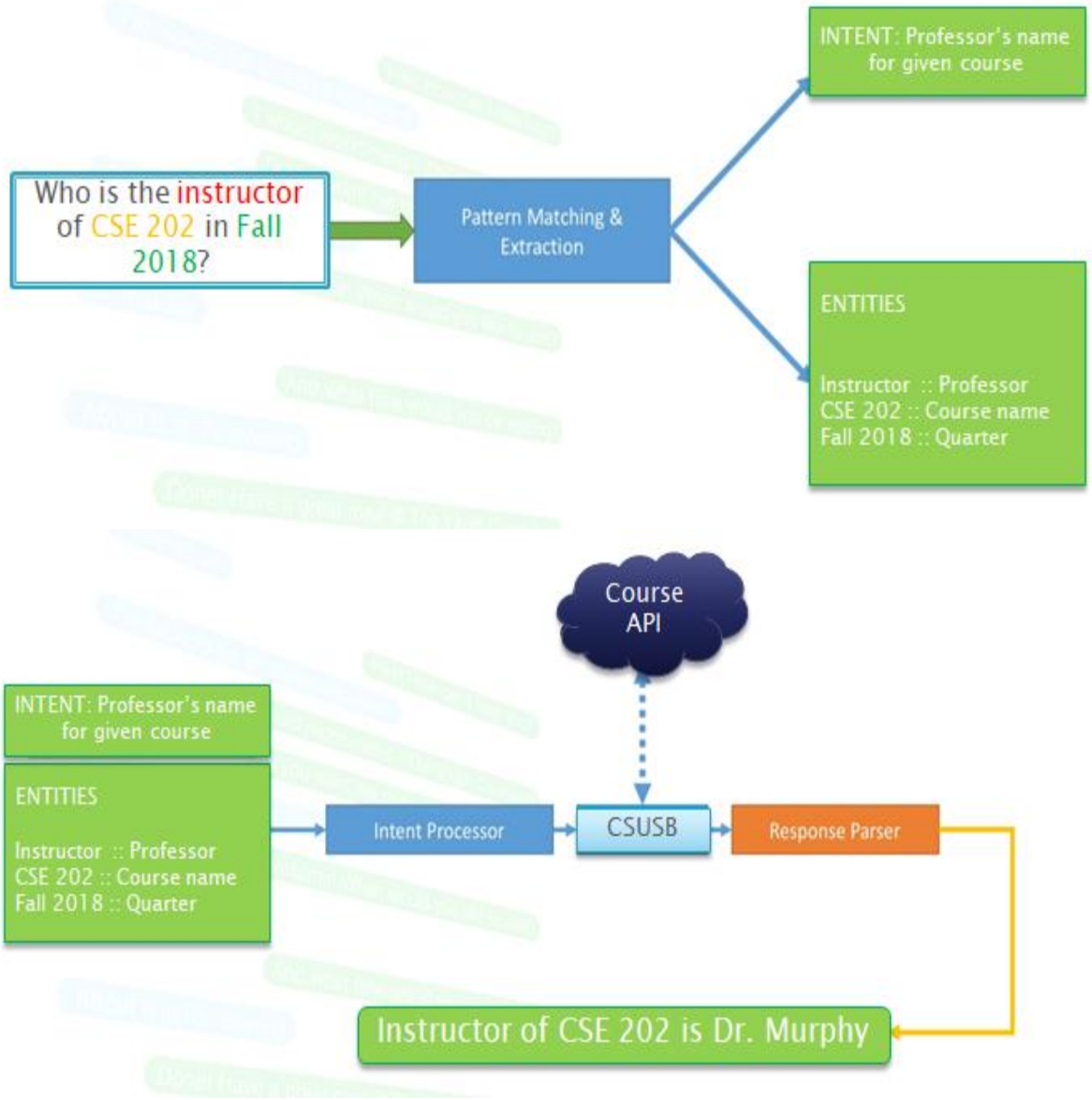


Figure 11. Chatbot in Nutshell

CHAPTER FIVE

PROJECT ARCHITECTURE

How CSUSB Chatbot Internally Works

A CSUSB Chatbot begins with human input. Human has to ask a question by typing in messaging app and that question has to pass through set of rules. If the given input does match with one of the given set of rules, the app should then move on to extract intent and entities. Intent is what the conversation is all about and Entity is actionable data components that bot has to extract from the user's messages to be able to process and send back an appropriate response. [11]

Once a human input goes into a bot then it will attempt to match a given set of patterns by RegEx Patterns Dictionary. If pattern matcher function matches one of the patterns, it goes on to extract Entities from the sentence. At the same time it also identifies Intent. For instance, user asks question like "Who is the location of class CSE 202". In the given sentence, a word "Location" and a course "CSE 202" are actionable piece of data which are known as entities and intent is, to find a location of given a class. Once it gets intent and entities, it goes on to process by calling an API to fetch the actual data based on the entities that have been extracted. Once it gets response from API service, it moves further to parse and process the response to create a suitable reply for the user to read which is then sent back to user.

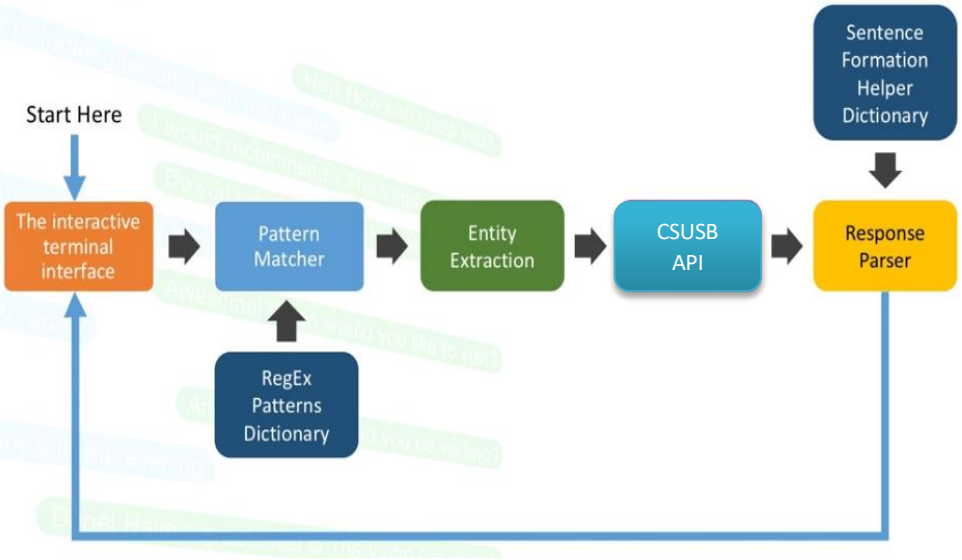


Figure 12. CSUSB Chatbot Mechanism

Figure 12 shows that process of a chatbot starts from The Interactive Terminal Interface which is nothing but a platform of Facebook Messenger. Once user type a message into a bot, the pattern matcher function will run the message through a set of regular expression rules. Regular Expression is a universal syntax that is used to detect patterns and extract data from a given sentence. If a pattern matcher function does find a pattern in the user's input, it extracts an actionable data from the sentence and then it runs through CSUSB API to get actual data. This data comes back as a JSON object which is parsed to get what user wants. JSON is a JavaScript Object Notation which is a light weight object to data interchange. JSON object is used for serializing and

transmitting data over a network connection. Once the JSON response is parsed, it will generate an appropriate reply which is printed back for user to read.

Connecting Chatbot With Facebook

Building a chatbot is a one part of this project other part is to connect a chatbot with Facebook Messenger platform to type messages. Figure 13 shows that process of deploying a chatbot on messenger platform begins with creating a Facebook page. This is absolutely essential. This page could be for the chatbot itself or it could be existing Facebook page such as the one for a business or an organization. The following step is to create a Facebook application and add the messenger platform product. CSUSB Chatbot links to a messenger platform using web-hooks. Web-hooks are specially designated URLs where messages from users are received by a chatbot. At the same time chatbot application talks to messenger by making calls to the send APIs. Once the connection between chatbot and messenger platform is made, next step is to test and train a bot to understand and respond to user. Once all of these are done, last step is to submit a chatbot to Facebook for approval to make it available to use in public. [12]

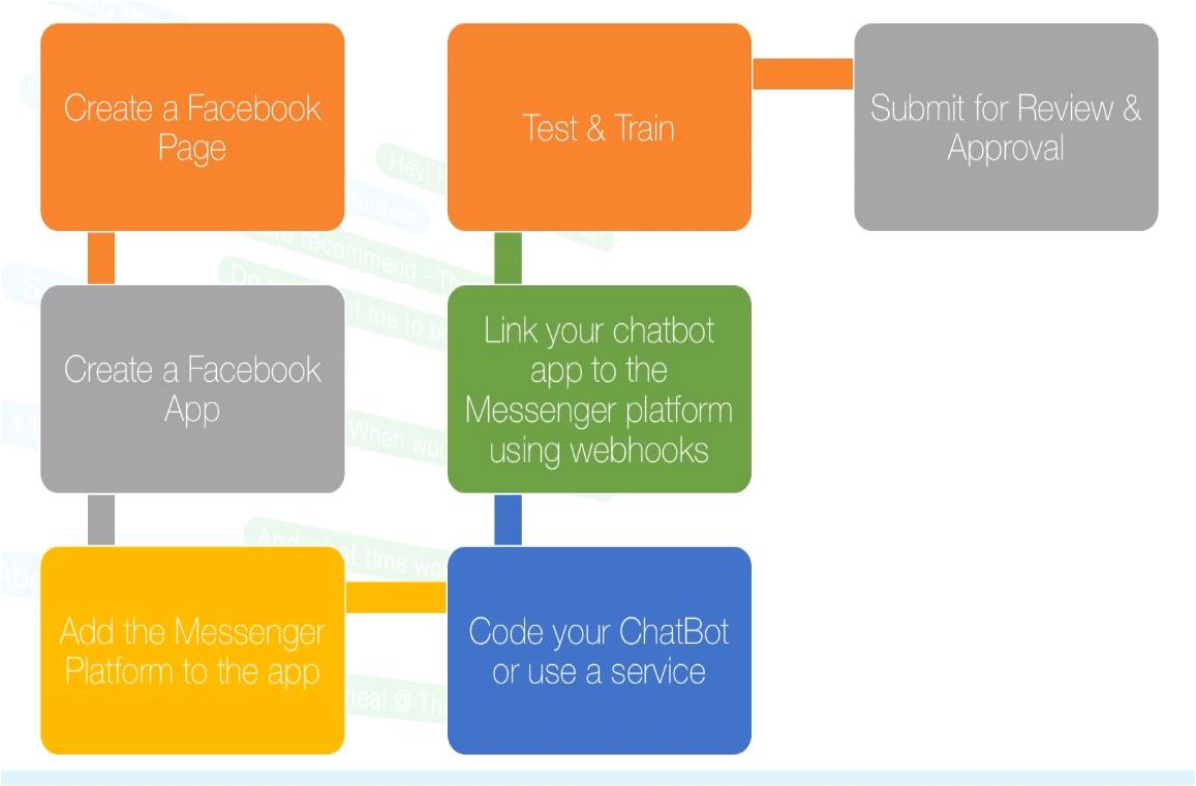


Figure 13. Chatbot on Messenger

CHAPTER SIX

UML DIAGRAMS

Use Case Diagram

Use Case Diagram represents user's interaction with Professors and Courses. Figure 14 shows that user can check List of all Professors, Email Address, Phone Number, Office Hours and Office Location of all faculties.

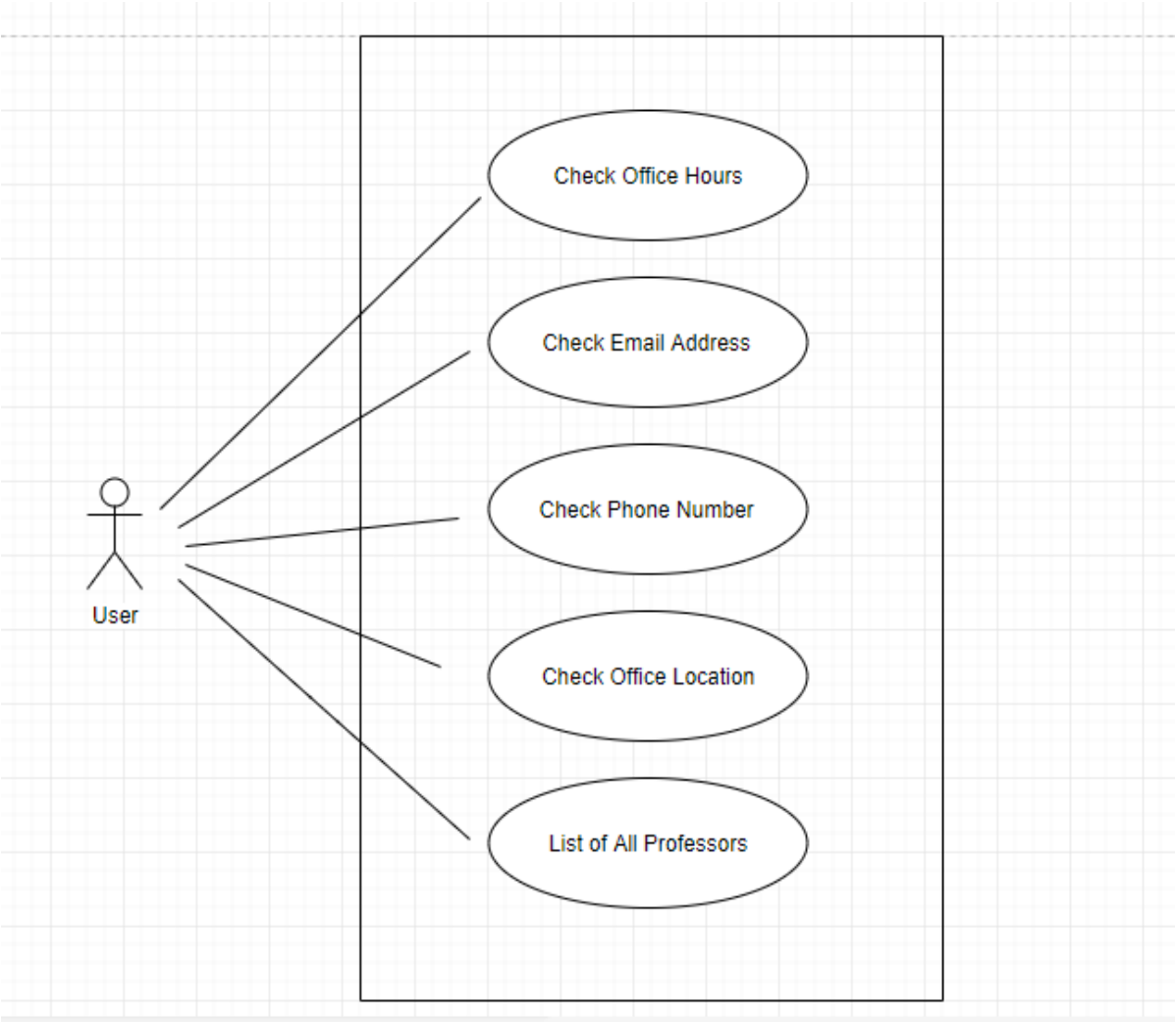


Figure 14. Use Case Diagram of Professors

Figure 15 describes that user is able to check Course Catalog, Course Title, Timings of Courses, Class Location, Number of Units and Course Instructor.

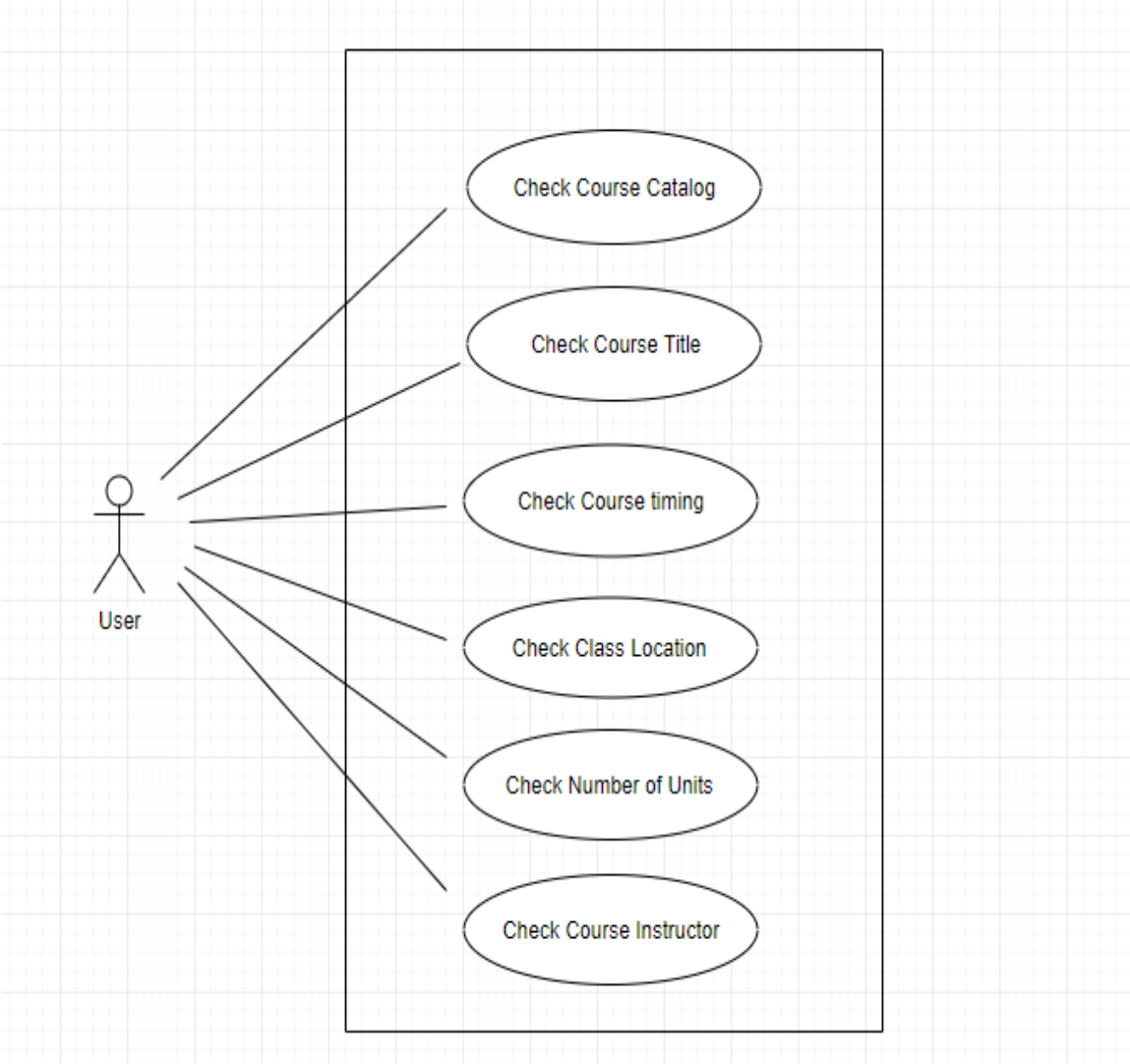


Figure 15. Use Case Diagram of Courses

Class Diagram

Figure 16 describes static structure and relationship between two classes Professors and Courses. It represents all data members and methods which are used in class Professors and Courses. It states that relationship between class Professor and class Course is One-to-Many which represents that one professor can teach more than one courses.

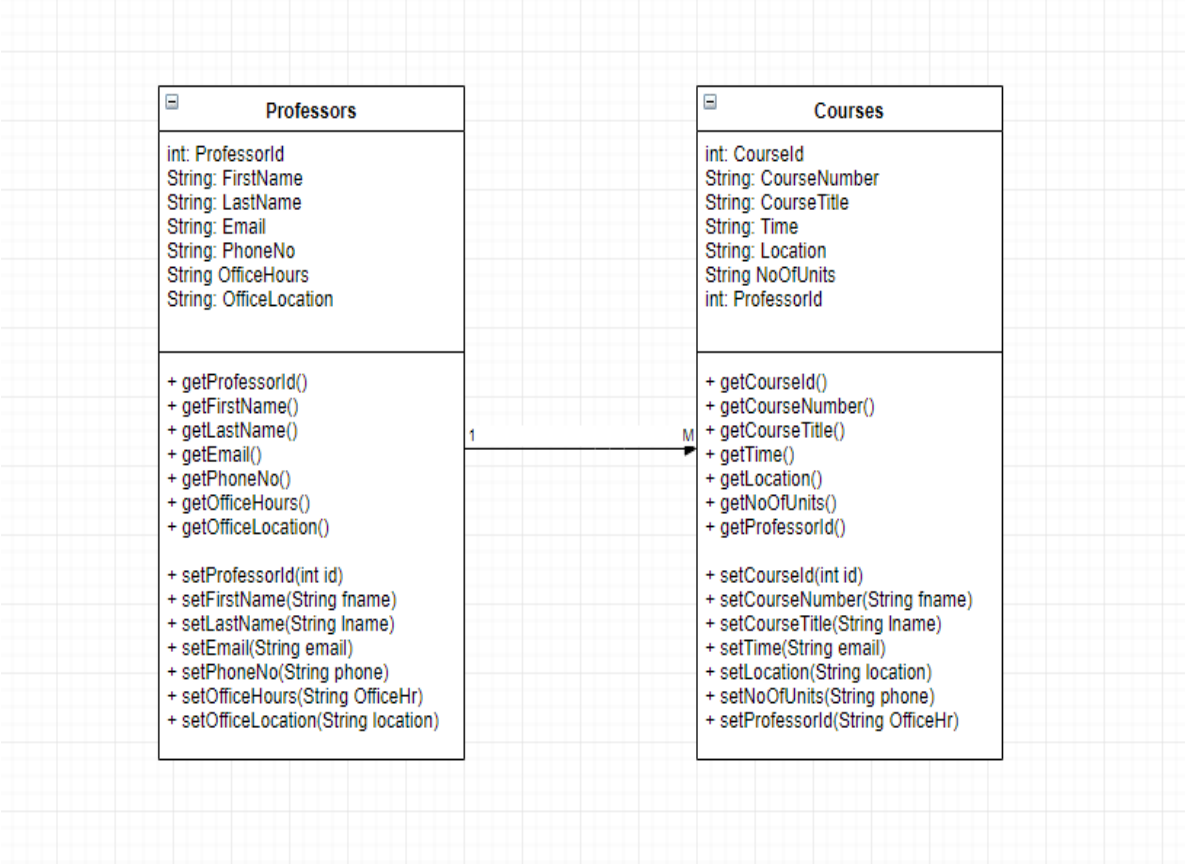


Figure 16. Class Diagram

Sequence Diagram

Figure 17 shows object interactions between User and Professors and also between User and Courses arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the User and Professors and also between User and Courses to carry out the functionality of the scenario.

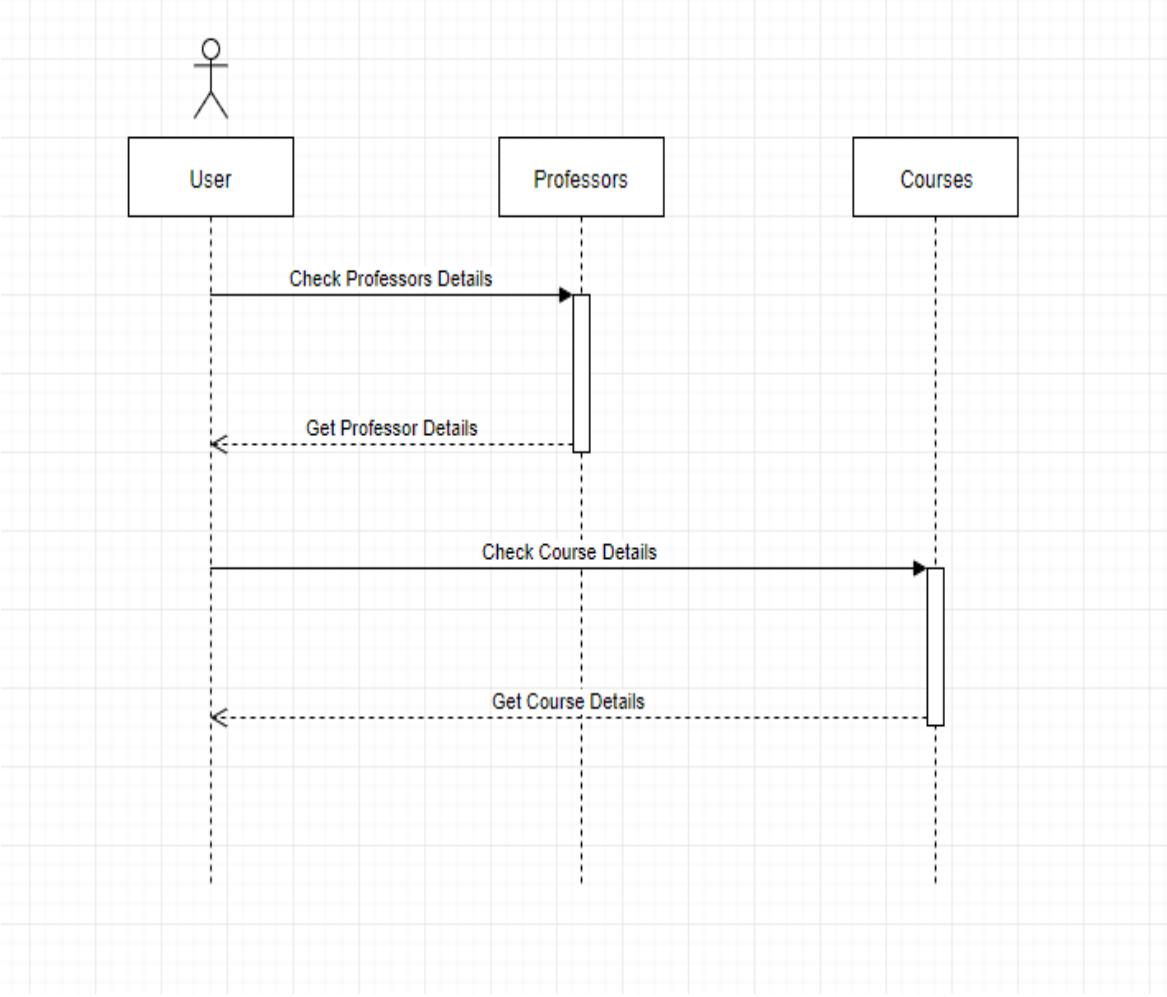


Figure 17. Sequance Diagram

Component Diagram

Figure 18 depicts the number of components and how different components are interect with each other. Here it is shown Professors Component and Courses Component are interect with User Component.

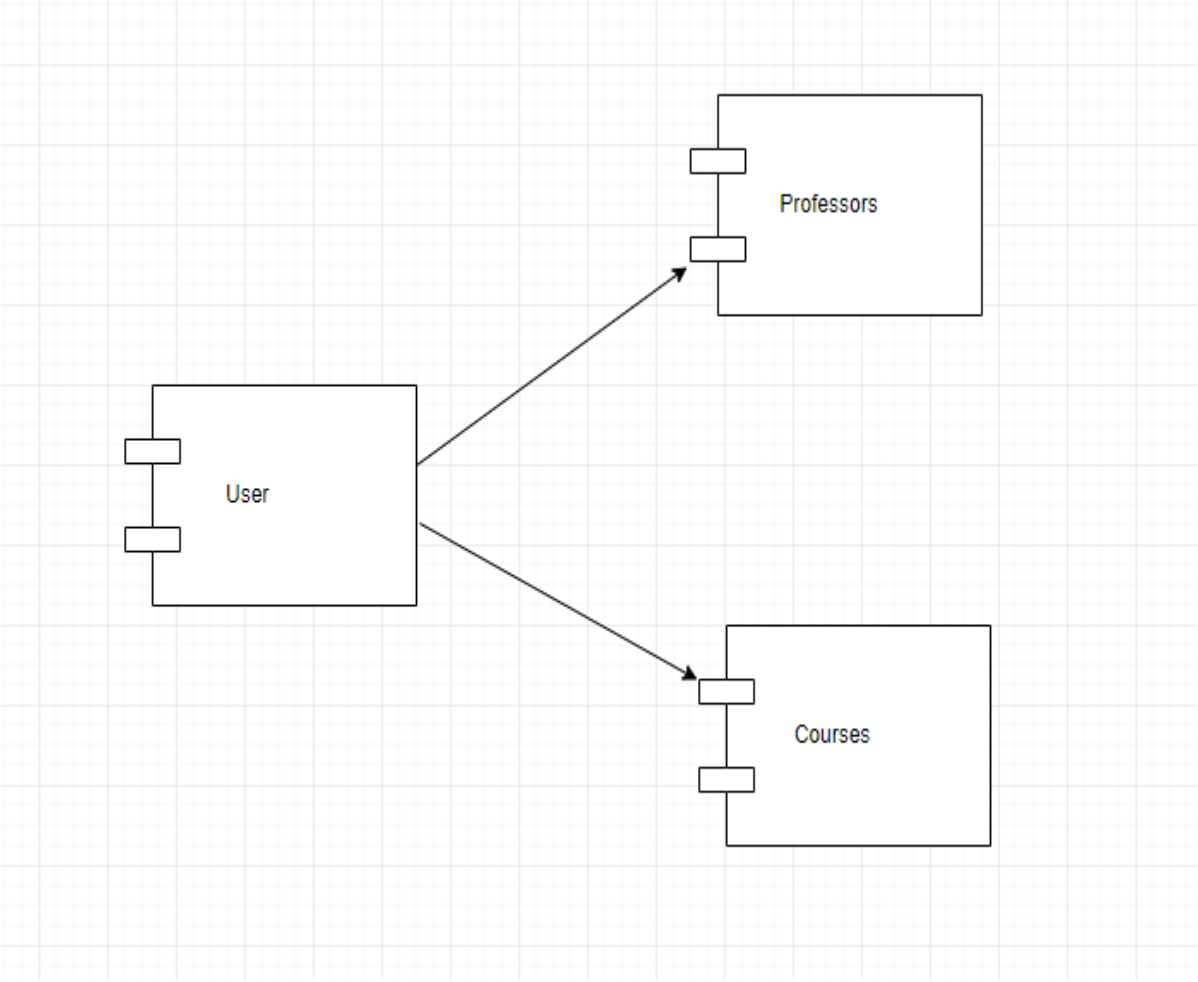


Figure 18. Component Diagram

State Diagram

Figure 19 shows that process of chatbot starts with user input which passes through pattern matcher to extract actionable piece of data. Once it extracts enity it process further to call CSUSB API to get an actual data. This data would be in JSON format which parses throgh Response Parser to convert in actual data which responds back to user.

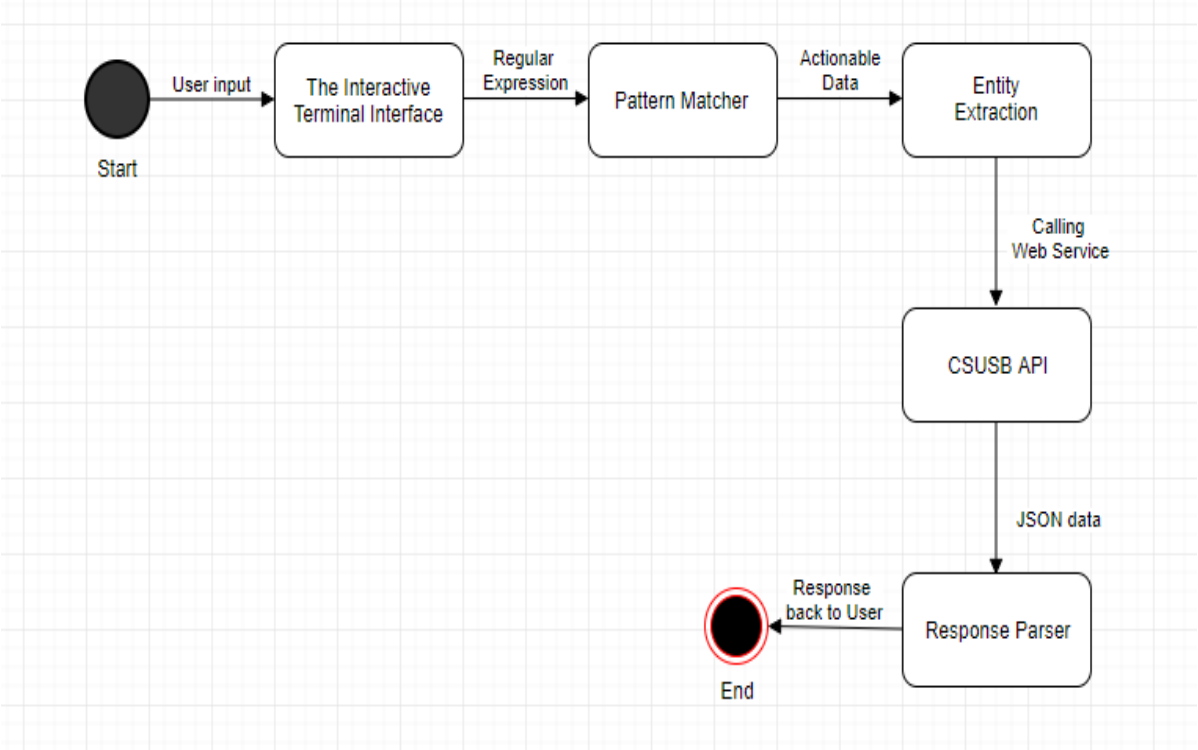


Figure 19. State Diagram

CHAPTER SEVEN

CONCLUSIONS

The CSUSB chatbot project demonstrates the ease of use of the Facebook chatbot for the purpose of finding information on professors and courses of Computer Science. The chatbot code uses regular expression, pattern matching and intent of the questions. It creates buckets based on the intent of the question and assigns the task to the appropriate module. It uses Natural Language Understanding to create the buckets of the user's intension. The chatbot uses Entity, Context and Intent model. Entity demonstrates the system of the chatbot. Intent demonstrates the action of the user. Context shows the state of the intension. Each module calls APIs to fetch the data specific to the question. Rest APIs are designed to fetch data from mySQL database. APIs are created on .NET platform and hosted on Azure cloud. A chatbot application executor executes and parses the API responses and forms answer sentences. The application code is hosted on the Heroku and salesforce cloud. It uses Facebook exposed APIs webhook to post the messages on Facebook messenger. Chatbots are scalable, resilient and rapid applications.

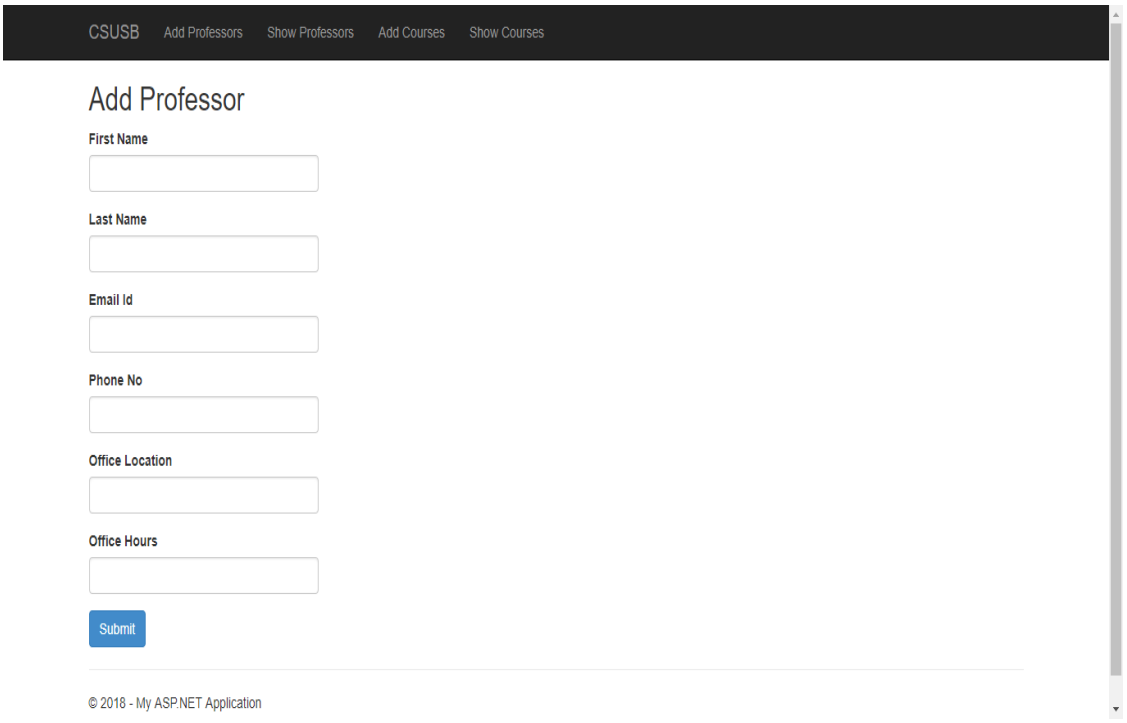
The Chatbots are obedient enough to answer all the questions. They can be also enhanced with Artificial Intelligence and Machine Learning. There is a lot of research going on about Natural Language Processing to understand the intent of the question. It can be achieved using generative and selective

modelling approach. The machine learned models can be trained using the questions asked by users. Sequence-to-Sequence is a famous modelling technique to train language models. The chatbots learn the context of the questions and forms reply. Tech companies also use Artificial Neural Network to train chatbot models. Data scientists and researchers all over the world are solving problems to learn the intent of the questions and reply intelligently.

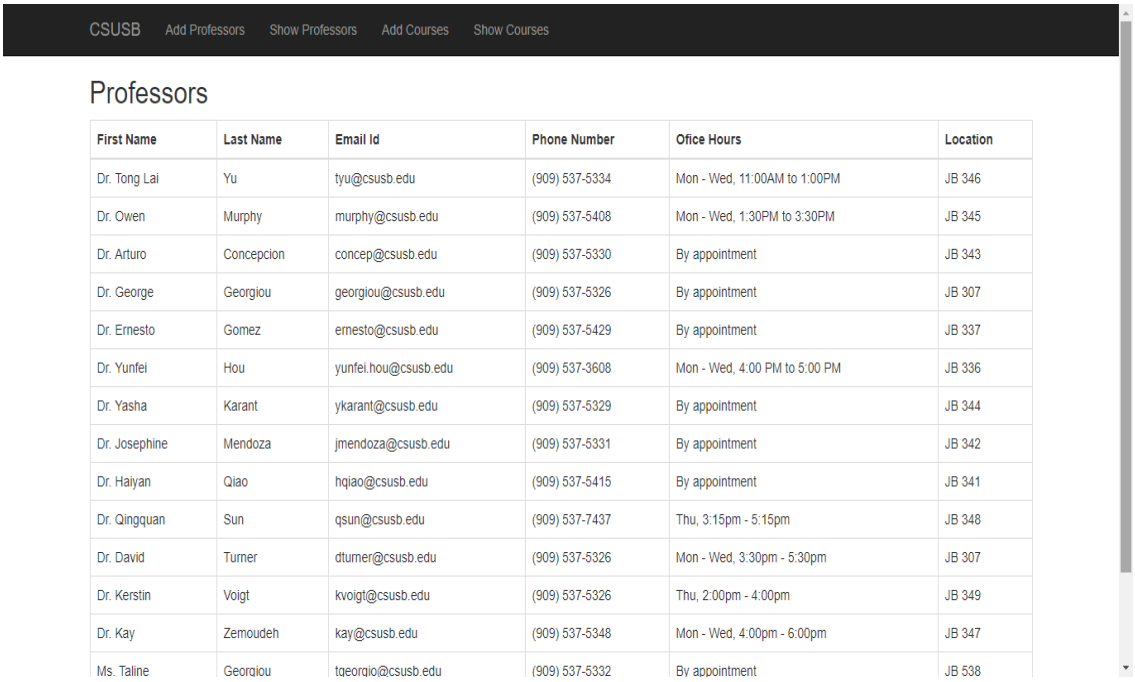
In the current world, businesses are investing a lot in developing chatbots. There are technology companies like aivo, botsify and chatfuel who provide infrastructure frameworks to create chatbots using artificial intelligence. Companies in Banking, eCommerce, Financial Services and IT advertising industry develop bots on mobile applications, Facebook platform or other messenger platform to help customers. The Chatbots fulfills customer's need such as weather information, crimes, news, package tracking updates, FAQs, feedback, shopping etc.

APPENDIX A

PROJECT SCREENSHOTS

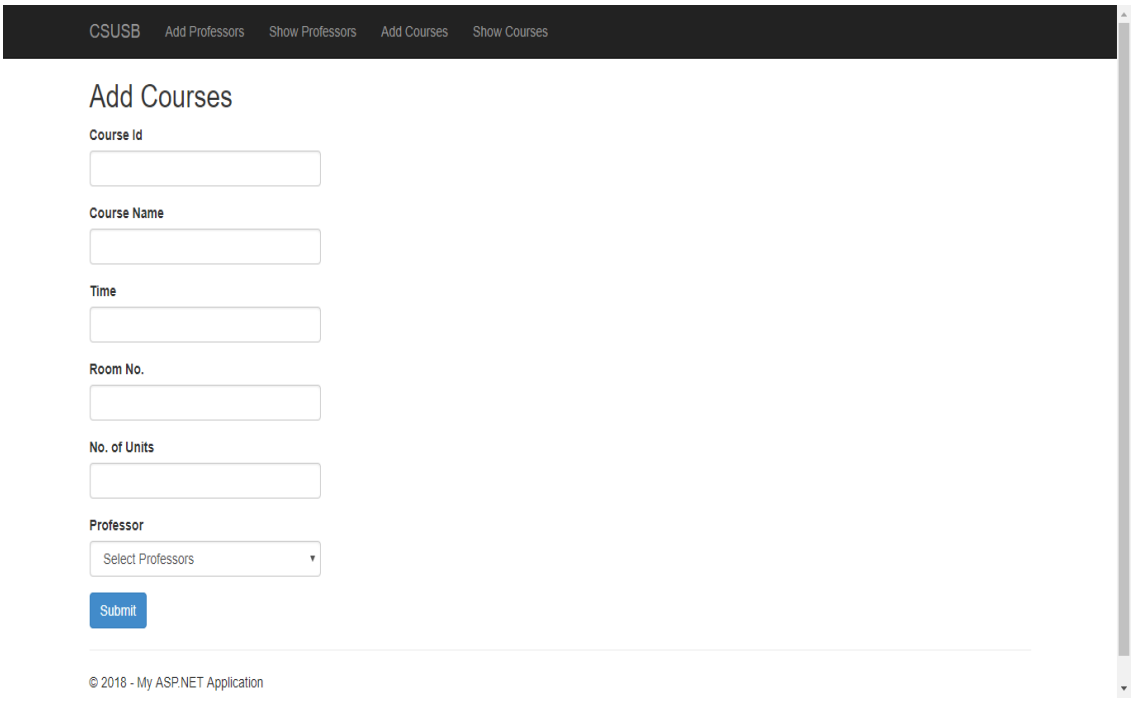


Add professor's details

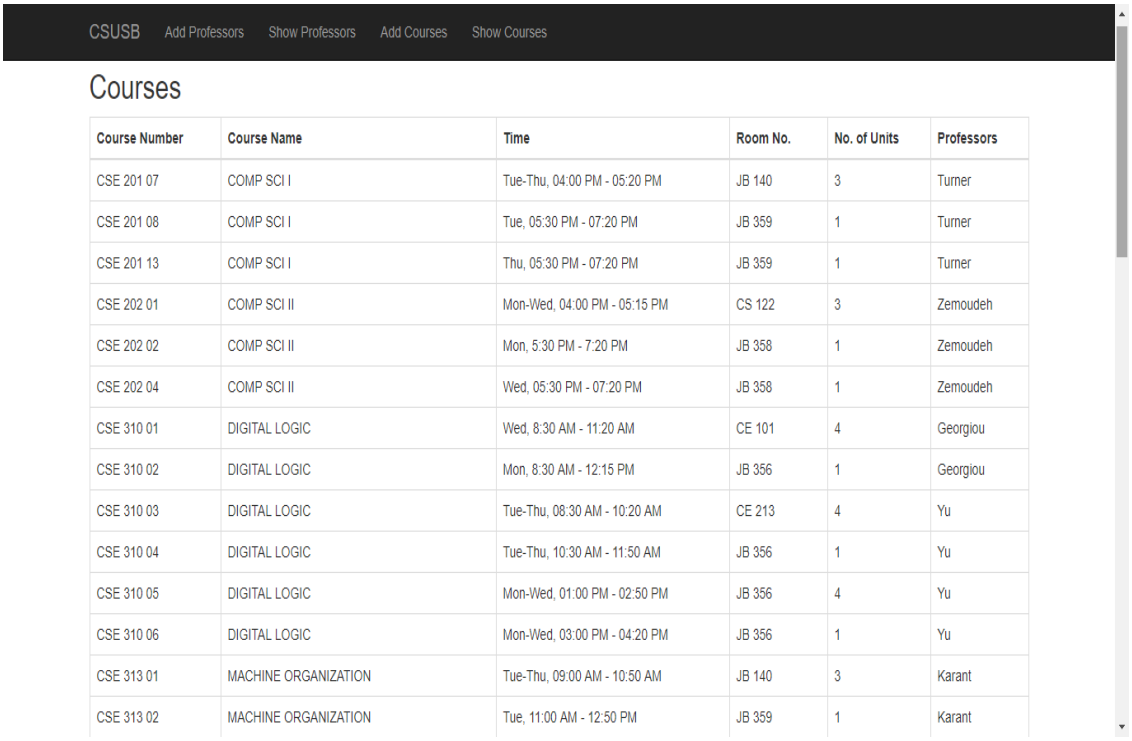


First Name	Last Name	Email Id	Phone Number	Ofice Hours	Location
Dr. Tong Lai	Yu	tyu@csusb.edu	(909) 537-5334	Mon - Wed, 11:00AM to 1:00PM	JB 346
Dr. Owen	Murphy	murphy@csusb.edu	(909) 537-5408	Mon - Wed, 1:30PM to 3:30PM	JB 345
Dr. Arturo	Concepcion	concep@csusb.edu	(909) 537-5330	By appointment	JB 343
Dr. George	Georgiou	georgiou@csusb.edu	(909) 537-5326	By appointment	JB 307
Dr. Ernesto	Gomez	ernesto@csusb.edu	(909) 537-5429	By appointment	JB 337
Dr. Yunfei	Hou	yunfei.hou@csusb.edu	(909) 537-3608	Mon - Wed, 4:00 PM to 5:00 PM	JB 336
Dr. Yasha	Karant	ykarant@csusb.edu	(909) 537-5329	By appointment	JB 344
Dr. Josephine	Mendoza	jmendoza@csusb.edu	(909) 537-5331	By appointment	JB 342
Dr. Haiyan	Qiao	hqiao@csusb.edu	(909) 537-5415	By appointment	JB 341
Dr. Qingquan	Sun	qsun@csusb.edu	(909) 537-7437	Thu, 3:15pm - 5:15pm	JB 348
Dr. David	Turner	dturner@csusb.edu	(909) 537-5326	Mon - Wed, 3:30pm - 5:30pm	JB 307
Dr. Kerstin	Voigt	kvoigt@csusb.edu	(909) 537-5326	Thu, 2:00pm - 4:00pm	JB 349
Dr. Kay	Zemoudeh	kay@csusb.edu	(909) 537-5348	Mon - Wed, 4:00pm - 6:00pm	JB 347
Ms. Taline	Georgiou	tgeorgio@csusb.edu	(909) 537-5332	By appointment	JB 538

Show professor's details

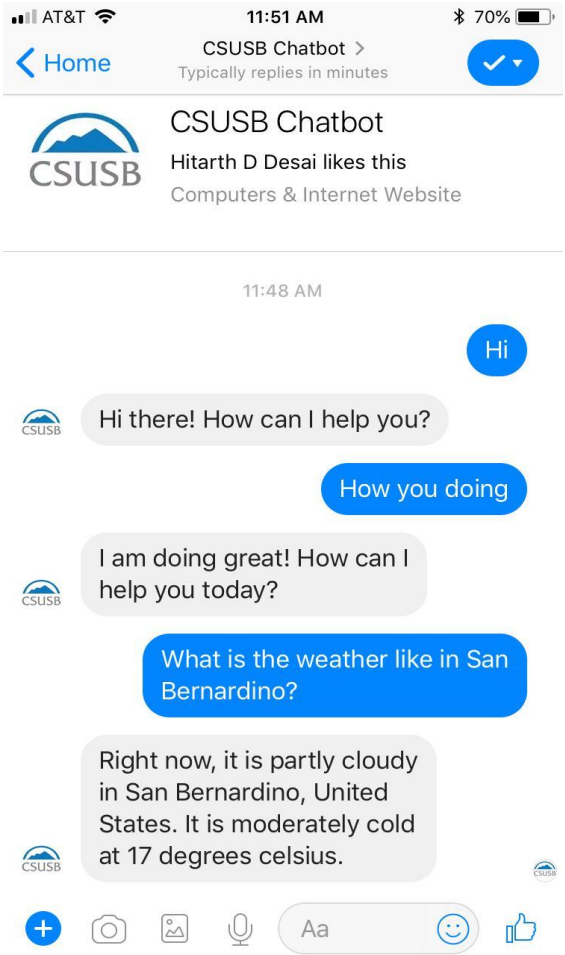


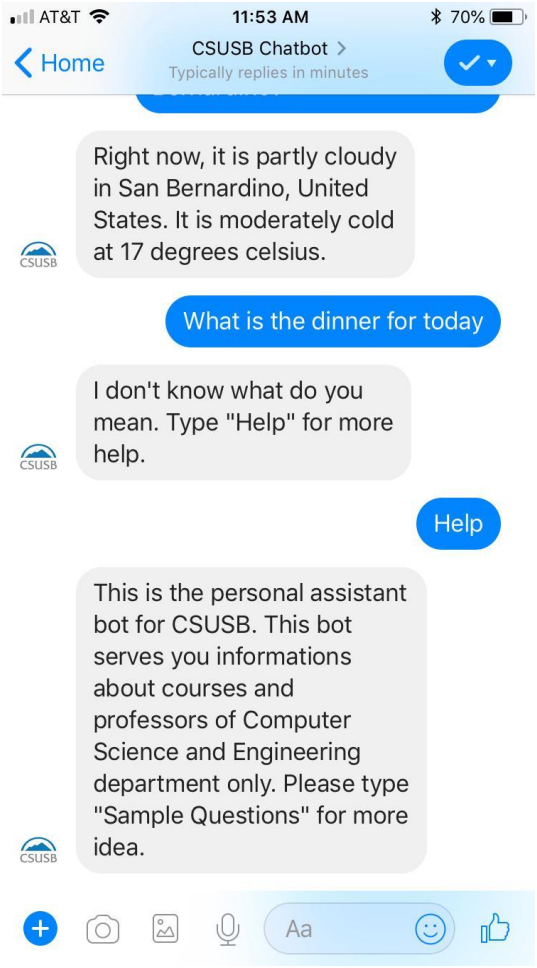
Add course details

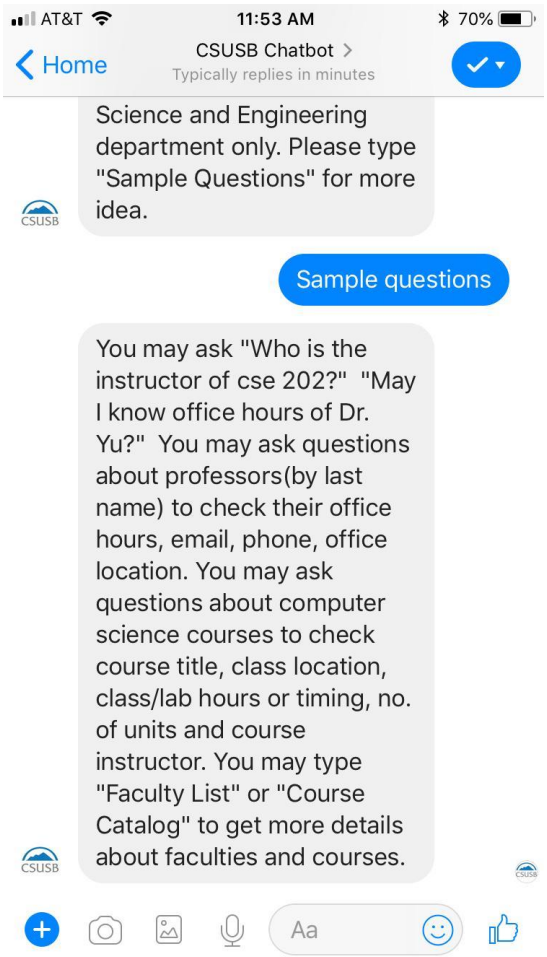


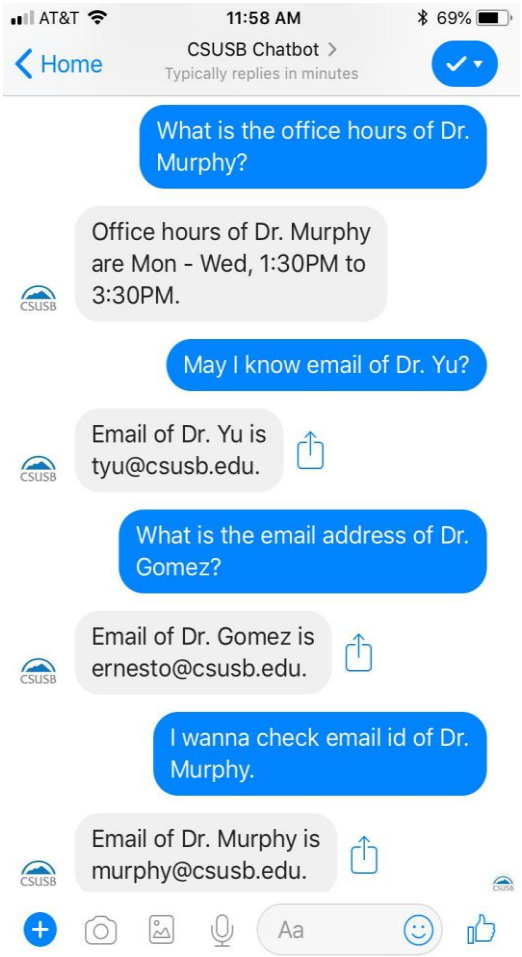
Course Number	Course Name	Time	Room No.	No. of Units	Professors
CSE 201 07	COMP SCI I	Tue-Thu, 04:00 PM - 05:20 PM	JB 140	3	Turner
CSE 201 08	COMP SCI I	Tue, 05:30 PM - 07:20 PM	JB 359	1	Turner
CSE 201 13	COMP SCI I	Thu, 05:30 PM - 07:20 PM	JB 359	1	Turner
CSE 202 01	COMP SCI II	Mon-Wed, 04:00 PM - 05:15 PM	CS 122	3	Zemoudeh
CSE 202 02	COMP SCI II	Mon, 5:30 PM - 7:20 PM	JB 358	1	Zemoudeh
CSE 202 04	COMP SCI II	Wed, 05:30 PM - 07:20 PM	JB 358	1	Zemoudeh
CSE 310 01	DIGITAL LOGIC	Wed, 8:30 AM - 11:20 AM	CE 101	4	Georgiou
CSE 310 02	DIGITAL LOGIC	Mon, 8:30 AM - 12:15 PM	JB 356	1	Georgiou
CSE 310 03	DIGITAL LOGIC	Tue-Thu, 08:30 AM - 10:20 AM	CE 213	4	Yu
CSE 310 04	DIGITAL LOGIC	Tue-Thu, 10:30 AM - 11:50 AM	JB 356	1	Yu
CSE 310 05	DIGITAL LOGIC	Mon-Wed, 01:00 PM - 02:50 PM	JB 356	4	Yu
CSE 310 06	DIGITAL LOGIC	Mon-Wed, 03:00 PM - 04:20 PM	JB 356	1	Yu
CSE 313 01	MACHINE ORGANIZATION	Tue-Thu, 09:00 AM - 10:50 AM	JB 140	3	Karant
CSE 313 02	MACHINE ORGANIZATION	Tue, 11:00 AM - 12:50 PM	JB 359	1	Karant

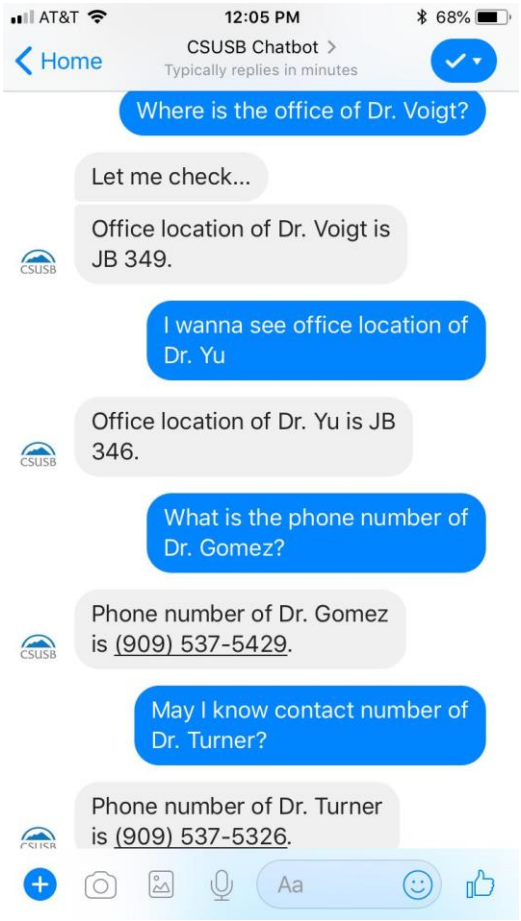
Show course details

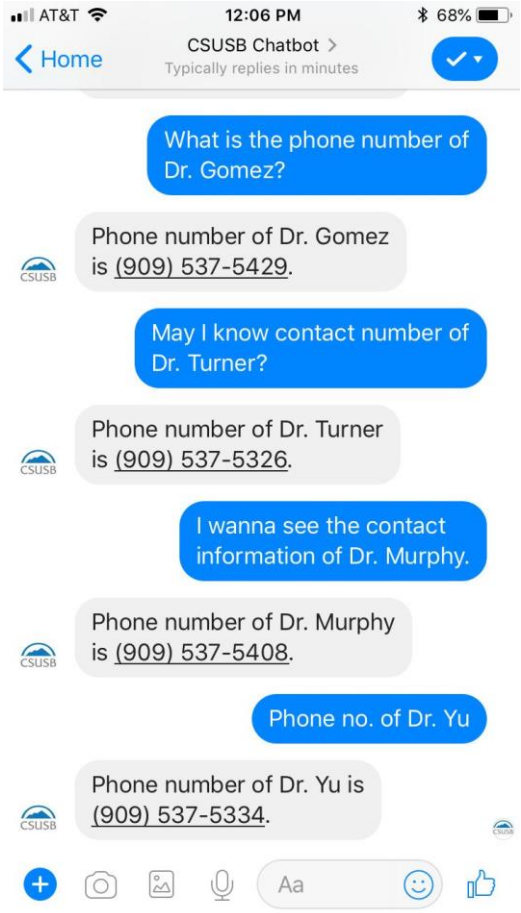

AT&T
11:51 AM
70%
Home
CSUSB Chatbot
Typically replies in minutes
CSUSB Chatbot
Hitarth D Desai likes this
Computers & Internet Website
11:48 AM
Hi
Hi there! How can I help you?
How you doing
I am doing great! How can I help you today?
What is the weather like in San Bernardino?
Right now, it is partly cloudy in San Bernardino, United States. It is moderately cold at 17 degrees celsius.
Aa

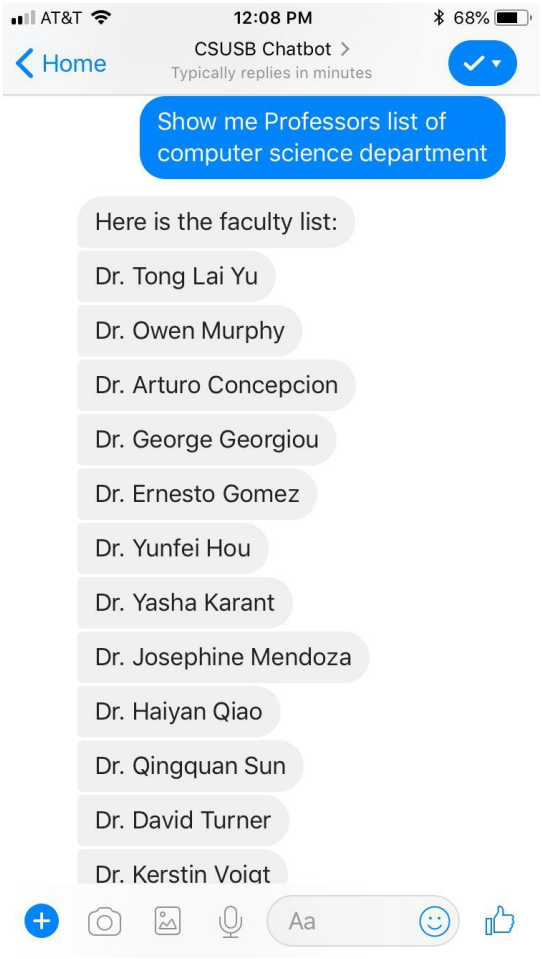

AT&T
11:53 AM
70%
Home
CSUSB Chatbot
Typically replies in minutes
Right now, it is partly cloudy in San Bernardino, United States. It is moderately cold at 17 degrees celsius.
What is the dinner for today
I don't know what do you mean. Type "Help" for more help.
Help
This is the personal assistant bot for CSUSB. This bot serves you informations about courses and professors of Computer Science and Engineering department only. Please type "Sample Questions" for more idea.
Aa

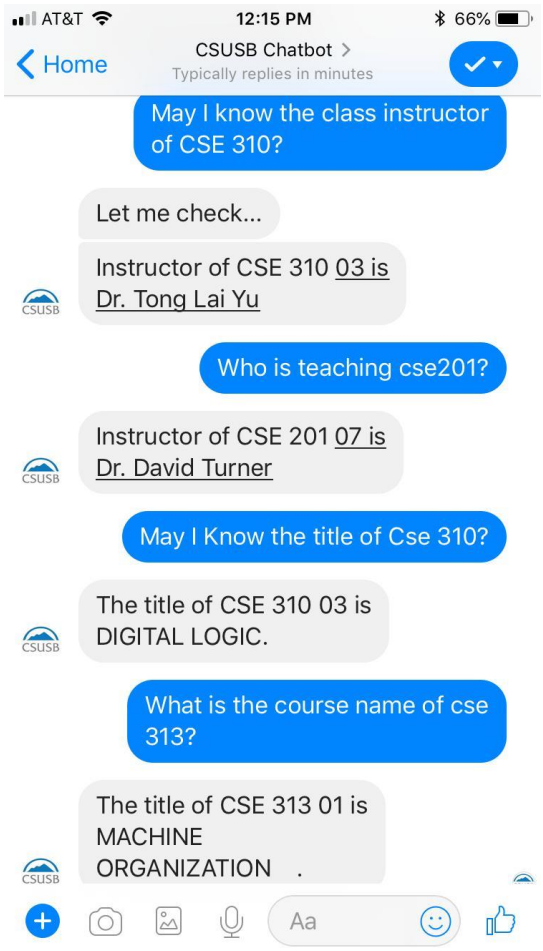

AT&T
11:53 AM
70%
Home
CSUSB Chatbot
Typically replies in minutes
Science and Engineering department only. Please type "Sample Questions" for more idea.
Sample questions
You may ask "Who is the instructor of cse 202?" "May I know office hours of Dr. Yu?" You may ask questions about professors(by last name) to check their office hours, email, phone, office location. You may ask questions about computer science courses to check course title, class location, class/lab hours or timing, no. of units and course instructor. You may type "Faculty List" or "Course Catalog" to get more details about faculties and courses.
Aa

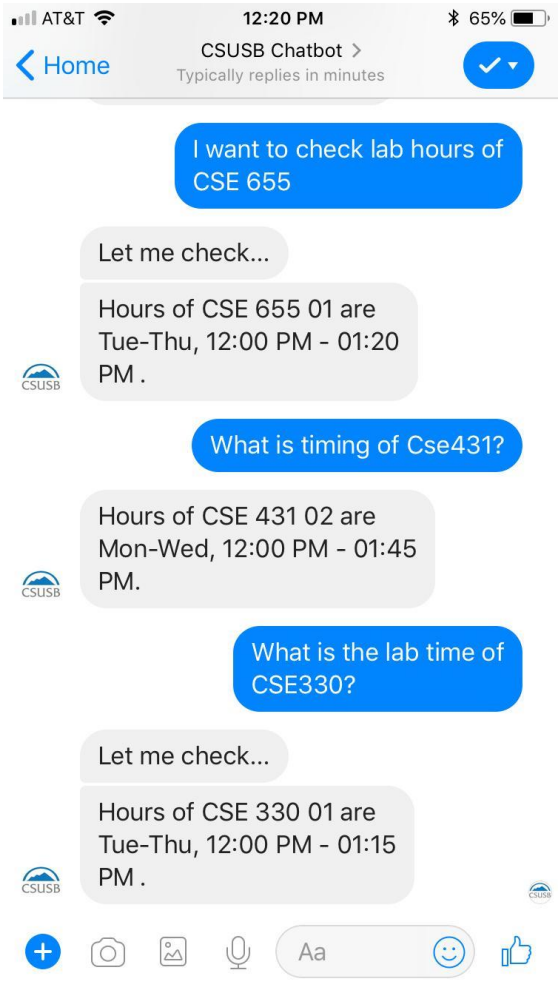

AT&T
11:58 AM
69%
Home
CSUSB Chatbot
Typically replies in minutes
What is the office hours of Dr. Murphy?
Office hours of Dr. Murphy are Mon - Wed, 1:30PM to 3:30PM.
May I know email of Dr. Yu?
Email of Dr. Yu is tyu@csusb.edu.
What is the email address of Dr. Gomez?
Email of Dr. Gomez is ernesto@csusb.edu.
I wanna check email id of Dr. Murphy.
Email of Dr. Murphy is murphy@csusb.edu.
Aa

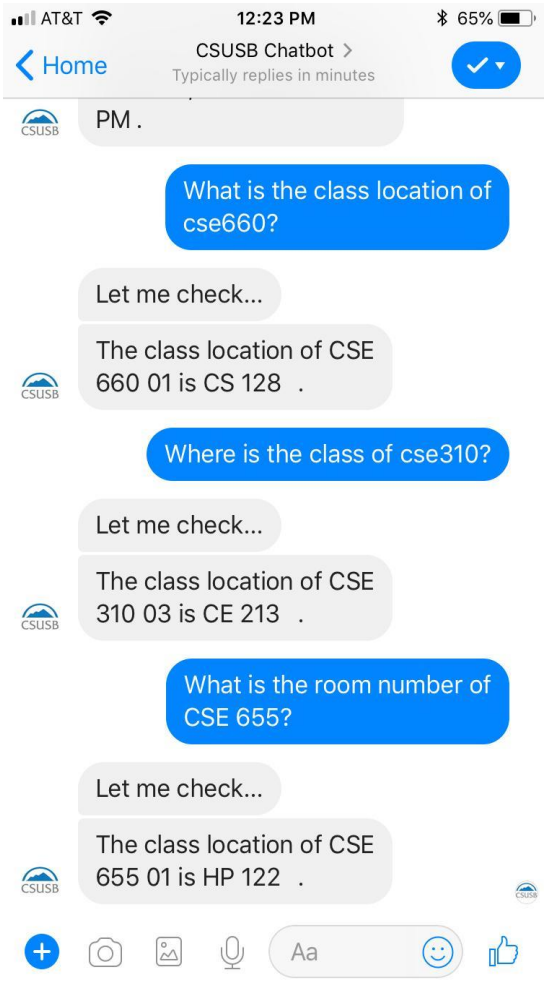

AT&T
12:05 PM
68%
Home
CSUSB Chatbot
Typically replies in minutes
Where is the office of Dr. Voigt?
Let me check...
Office location of Dr. Voigt is JB 349.
I wanna see office location of Dr. Yu
Office location of Dr. Yu is JB 346.
What is the phone number of Dr. Gomez?
Phone number of Dr. Gomez is (909) 537-5429.
May I know contact number of Dr. Turner?
Phone number of Dr. Turner is (909) 537-5326.
Aa

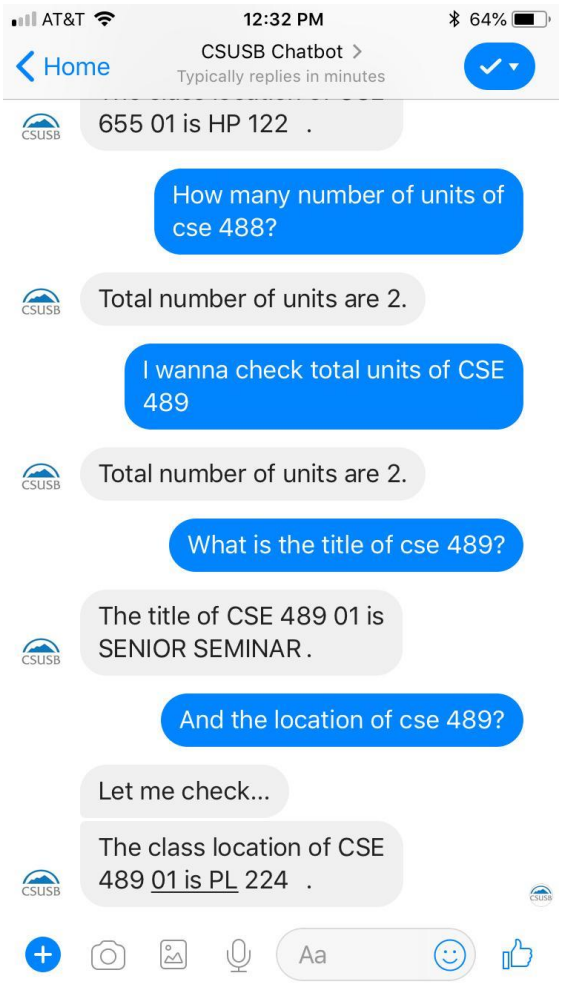

AT&T
12:06 PM
68%
Home
CSUSB Chatbot
Typically replies in minutes
What is the phone number of Dr. Gomez?
Phone number of Dr. Gomez is (909) 537-5429.
May I know contact number of Dr. Turner?
Phone number of Dr. Turner is (909) 537-5326.
I wanna see the contact information of Dr. Murphy.
Phone number of Dr. Murphy is (909) 537-5408.
Phone no. of Dr. Yu
Phone number of Dr. Yu is (909) 537-5334.
Aa

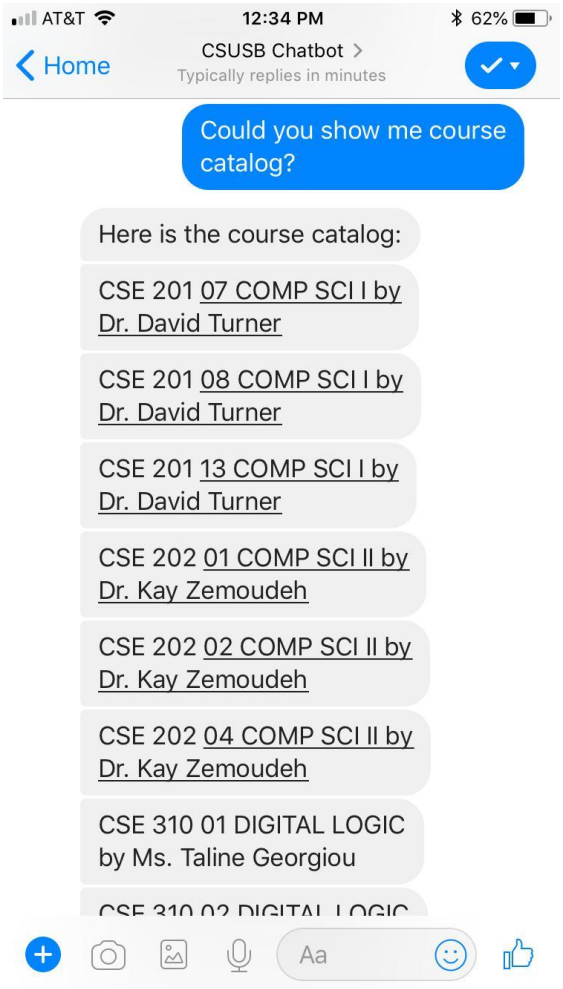

AT&T
12:08 PM
68%
Home
CSUSB Chatbot
Typically replies in minutes
Show me Professors list of computer science department
Here is the faculty list:
Dr. Tong Lai Yu
Dr. Owen Murphy
Dr. Arturo Concepcion
Dr. George Georgiou
Dr. Ernesto Gomez
Dr. Yunfei Hou
Dr. Yasha Karant
Dr. Josephine Mendoza
Dr. Haiyan Qiao
Dr. Qingquan Sun
Dr. David Turner
Dr. Kerstin Voigt
Aa

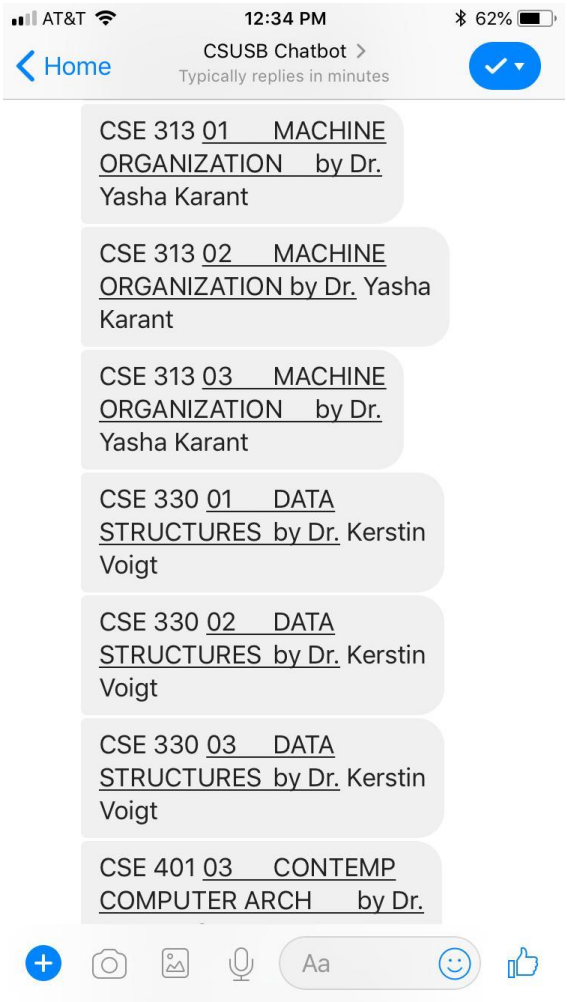

AT&T
12:15 PM
66%
Home
CSUSB Chatbot
Typically replies in minutes
May I know the class instructor of CSE 310?
Let me check...
Instructor of CSE 310 03 is Dr. Tong Lai Yu
Who is teaching cse201?
Instructor of CSE 201 07 is Dr. David Turner
May I Know the title of Cse 310?
The title of CSE 310 03 is DIGITAL LOGIC.
What is the course name of cse 313?
The title of CSE 313 01 is MACHINE ORGANIZATION .
Aa

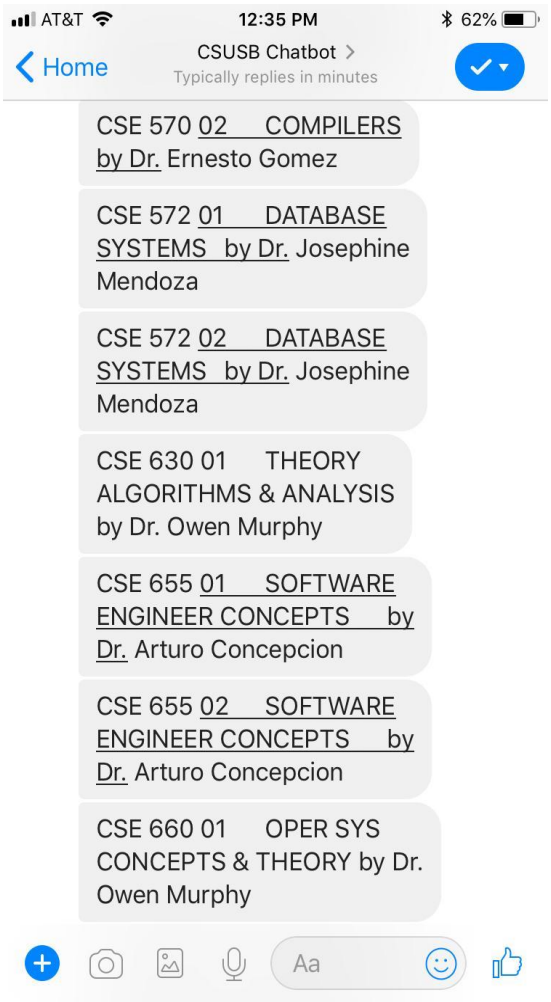

AT&T
12:20 PM
65%
Home
CSUSB Chatbot
Typically replies in minutes
I want to check lab hours of CSE 655
Let me check...
Hours of CSE 655 01 are Tue-Thu, 12:00 PM - 01:20 PM .
What is timing of Cse431?
Hours of CSE 431 02 are Mon-Wed, 12:00 PM - 01:45 PM.
What is the lab time of CSE330?
Let me check...
Hours of CSE 330 01 are Tue-Thu, 12:00 PM - 01:15 PM .
Aa


AT&T
12:23 PM
65%
Home
CSUSB Chatbot
Typically replies in minutes
PM .
What is the class location of cse660?
Let me check...
The class location of CSE 660 01 is CS 128 .
Where is the class of cse310?
Let me check...
The class location of CSE 310 03 is CE 213 .
What is the room number of CSE 655?
Let me check...
The class location of CSE 655 01 is HP 122 .
Aa


AT&T
12:32 PM
64%
Home
CSUSB Chatbot
Typically replies in minutes
655 01 is HP 122 .
How many number of units of cse 488?
Total number of units are 2.
I wanna check total units of CSE 489
Total number of units are 2.
What is the title of cse 489?
The title of CSE 489 01 is SENIOR SEMINAR .
And the location of cse 489?
Let me check...
The class location of CSE 489 01 is PL 224 .
Aa


AT&T
12:34 PM
62%
Home
CSUSB Chatbot
Typically replies in minutes
Could you show me course catalog?
Here is the course catalog:
CSE 201 07 COMP SCI I by Dr. David Turner
CSE 201 08 COMP SCI I by Dr. David Turner
CSE 201 13 COMP SCI I by Dr. David Turner
CSE 202 01 COMP SCI II by Dr. Kay Zemoudeh
CSE 202 02 COMP SCI II by Dr. Kay Zemoudeh
CSE 202 04 COMP SCI II by Dr. Kay Zemoudeh
CSE 310 01 DIGITAL LOGIC by Ms. Taline Georgiou
Aa


AT&T
12:34 PM
62%
Home
CSUSB Chatbot
Typically replies in minutes
CSE 313 01 MACHINE ORGANIZATION by Dr. Yasha Karant
CSE 313 02 MACHINE ORGANIZATION by Dr. Yasha Karant
CSE 313 03 MACHINE ORGANIZATION by Dr. Yasha Karant
CSE 330 01 DATA STRUCTURES by Dr. Kerstin Voigt
CSE 330 02 DATA STRUCTURES by Dr. Kerstin Voigt
CSE 330 03 DATA STRUCTURES by Dr. Kerstin Voigt
CSE 401 03 CONTEMP COMPUTER ARCH by Dr.
Aa


AT&T
12:35 PM
62%
Home
CSUSB Chatbot
Typically replies in minutes
CSE 570 02 COMPILERS by Dr. Ernesto Gomez
CSE 572 01 DATABASE SYSTEMS by Dr. Josephine Mendoza
CSE 572 02 DATABASE SYSTEMS by Dr. Josephine Mendoza
CSE 630 01 THEORY ALGORITHMS & ANALYSIS by Dr. Owen Murphy
CSE 655 01 SOFTWARE ENGINEER CONCEPTS by Dr. Arturo Concepcion
CSE 655 02 SOFTWARE ENGINEER CONCEPTS by Dr. Arturo Concepcion
CSE 660 01 OPER SYS CONCEPTS & THEORY by Dr. Owen Murphy
Aa

APPENDIX B

CODE OF CRITICAL PARTS

Server.js

```
const express = require('express');
const bodyParser = require('body-parser');

const config = require('./config');
const FBeamer = require('./fbeamer');

const matcher = require('./matcher');
//Get APIs
const weather = require('./weather');
const professorApi = require('./api/professorApi');
const courseApi = require('./api/courseApi');
const professorAll = require('./api/professorAllApi');
const courseAll = require('./api/courseAllApi');
//Weather Parser
const {currentWeather} = require('./parser');
//Professor Parser
const {officeHr} = require('./parser/professorsParse');
const {emailId} = require('./parser/professorsParse');
const {phoneNo} = require('./parser/professorsParse');
const {officeLocation} = require('./parser/professorsParse');
const {allProfessors} = require('./parser/professorsParse');
//Course Parser
const {courseTitle} = require('./parser/courseParse');
const {courseTime} = require('./parser/courseParse');
const {classLocation} = require('./parser/courseParse');
const {courseUnits} = require('./parser/courseParse');
const {courseByProfessor} = require('./parser/courseParse');

const server = express();
const PORT = process.env.PORT || 3000;
const f = new FBeamer(config.fb);

server.get('/', (req, res) => f.registerHook(req,res));
server.post('/', bodyParser.json({
   verify: f.verifySignature
}));

let mapCourse = getMapCourses();

server.post('/', (req, res, next) => {
   return f.incoming(req, res, data => {
      try{
         if(data.type === 'text'){
            matcher(data.content, async resp => {
               switch(resp.intent){
                  case 'Hello':
                     await f.txt(data.sender, `${resp.entities.greeting} How can I help you?`);
                     break;
                  case 'Greetings':
                     await f.txt(data.sender, 'I am doing great! How can I help you today?');
                     break;
                  case 'CurrentWeather':
                     await f.txt(data.sender, 'Let me check...');
                     let weatherData = await weather(resp.entities.city, 'current');
```

```
                    let cwResult = currentWeather(weatherData);
                    await f.txt(data.sender, cwResult);
                    break;
                  //Show Professors:
                 case 'AllProfessors':
                    await f.txt(data.sender, 'Here is the faculty list:');
                    let response = await professorAll(resp.entities.professor);
                    for(let i = 0; i<response.length; i++) {
                        if(response[i].lname){
                            await f.txt(data.sender, response[i].fname +' '+ response[i].lname);
                        }
                    }
                    break;
                 case 'OfficeHours':
                    await f.txt(data.sender, 'Let me check...');
                    let cwData1 = await
professorApi(mapProfessor.get((resp.entities.professor).toLowerCase()));
                    let office = officeHr(cwData1);
                    await f.txt(data.sender, office);
                    break;
                 case 'Email':
                   // await f.txt(data.sender, 'Let me check...');
                    let cwData2 = await
professorApi(mapProfessor.get((resp.entities.professor).toLowerCase()));
                    let email = emailId(cwData2);
                    await f.txt(data.sender, email);
                    break;
                 case 'Phone':
                   //  await f.txt(data.sender, 'Let me check...');
                    let cwData3 = await
professorApi(mapProfessor.get((resp.entities.professor).toLowerCase()));
                    let phone = phoneNo(cwData3);
                    await f.txt(data.sender, phone);
                    break;
                 case 'OfficeLocation':
                    await f.txt(data.sender, 'Let me check...');
                    let cwData4 = await
professorApi(mapProfessor.get((resp.entities.professor).toLowerCase()));
                    let loc = officeLocation(cwData4);
                    await f.txt(data.sender, loc);
                    break;
                // Show Courses:
                 case 'AllCourses':
                    await f.txt(data.sender, 'Here is the course catalog:');
                    let courses = await courseAll(resp.entities.course);
                    for(let i = 0; i<courses.length; i++) {
                        if(courses[i].id){
                            await f.txt(data.sender, courses[i].number +' '+ courses[i].name+' by
'+courses[i].professors.fname+' '+courses[i].professors.lname);
                        }
                    }
                    break;
                 case 'CourseName':
                  //   await f.txt(data.sender, 'Let me check...');
                    let courseData1 = await courseApi(mapCourse.get((resp.entities.course).replace(/\s+/,
"").toLowerCase()));
                    let title = courseTitle(courseData1);
                    await f.txt(data.sender, title);
                    break;
```

```
                    case 'CourseTime':
                        await f.txt(data.sender, 'Let me check...');
                        let courseData2 = await courseApi(mapCourse.get((resp.entities.course).replace(/\s+/,
"").toLowerCase()));
                        let time = courseTime(courseData2);
                        await f.txt(data.sender, time);
                        break;
                    case 'ClassLocation':
                        await f.txt(data.sender, 'Let me check...');
                        let courseData3 = await courseApi(mapCourse.get((resp.entities.course).replace(/\s+/,
"").toLowerCase()));
                        let location = classLocation(courseData3);
                        await f.txt(data.sender, location);
                        break;
                    case 'CourseUnits':
                      //  await f.txt(data.sender, 'Let me check...');
                        let courseData4 = await courseApi(mapCourse.get((resp.entities.course).replace(/\s+/,
"").toLowerCase()));
                        let units = courseUnits(courseData4);
                        await f.txt(data.sender, units);
                        break;
                    case 'CourseByProfessor':
                        await f.txt(data.sender, 'Let me check...');
                        let courseData5 = await courseApi(mapCourse.get((resp.entities.course).replace(/\s+/,
"").toLowerCase()));
                        let course = courseByProfessor(courseData5);
                        await f.txt(data.sender, course);
                        break;
                    case 'Help':
                        await f.txt(data.sender,`This is the personal assistant bot for CSUSB. This bot serves
you informations about courses and professors of Computer Science and Engineering department
only. Please type "Sample Questions" for more idea.`);
                        break;
                    case 'SampleQuestions':
                        await f.txt(data.sender,`You may ask "Who is the instructor of cse 202?"  "May I
know office hours of Dr. Yu?"  You may ask questions about professors(by last name) to check their
office hours, email, phone, office location. You may ask questions about computer science courses
to check course title, class location, class/lab hours or timing, no. of units and course instructor.
You may type "Faculty List" or "Course Catalog" to get more details about faculties and courses.`);
                        break;
                    case 'Exit':
                        await f.txt(data.sender, `${resp.entities.greeting} Krutarth!`);
                        await f.txt(data.sender, `Have a great day :)`);
                        break;
                    default: {
                    await f.txt(data.sender, `I don't know what do you mean. Type "Help" for more help.`);
                    }
                }
            });
        }
    } catch(e){
        console.log(e);
    }
  });
});
server.listen(PORT, () => console.log(`CSUSB ChatBot Service running on Port ${PORT}`));
```

Pattern.js

```
const patternDict = [{
    pattern : '\\b(?<greeting>Hi|Hello|Hey)\\b',
    intent : 'Hello'
},{
    pattern: '\\b(How are you|How are you doing|How you doing)',
    intent: 'Greetings'
},{
    pattern: 'like\\sin\\s\\b(?<city>.+)',
    intent: 'CurrentWeather'
},{
    pattern: '\\b(?<professor>.+)\\slist',
    intent: 'AllProfessors'
},{
    pattern: '(office ?hours?)\\sof\\s?(d|D)?r?.?\\s\\b(?<professor>.+)\\b',
    intent: 'OfficeHours'
},{
    pattern: '(email ?id|email|email ?address)\\sof\\s?(d|D)?r?.?\\s\\b(?<professor>.+)\\b',
    intent: 'Email'
},{
    pattern: '(phone no.?|phone number|phone|contact|contact number|contact
information)\\sof\\s?(d|D)?r?.?\\s\\b(?<professor>.+)\\b',
    intent: 'Phone'
},{
    pattern: '(office number|office no.?|office location|office)\\sof\\s?(d|D)?r?.?\\s\\b(?<professor>.+)\\b',
    intent: 'OfficeLocation'
},{
    pattern: '(name|title)\\sof\\s\\b(?<course>.+)\\b',
    intent: 'CourseName'
},{
    pattern: '(time?i?n?g?|duration|hours?|lab hours?|lab time?i?n?g?|class
hours?)\\sof\\s\\b(?<course>.+)\\b',
    intent: 'CourseTime'
},{
    pattern: '(location|room number|room no.?|lab ?n?u?m?b?e?r?|class ?n?u?m?b?e?r?|class
location)\\sof\\s\\b(?<course>.+)\\b',
    intent: 'ClassLocation'
},{
    pattern: '(units?|credits?)\\s(of|have|has)\\s\\b(?<course>.+)\\b',
    intent: 'CourseUnits'
},{
    pattern: '(instructor|faculty|professor|teacher)\\sof\\s\\b(?<course>.+)\\b',
    intent: 'CourseByProfessor'
},{
    pattern: '(teache?s?i?n?g?)\\s\\b(?<course>.+)\\b',
    intent: 'CourseByProfessor'
},{
    pattern: '(list of ?a?l?l?|all|all the|catalogu?e? of)\\s\\b(?<course>.+)\\b',
    intent: 'AllCourses'
},{
    pattern: '\\b(?<course>.+)\\b\\sare\\s?c?a?t?a?l?o?g?o?u?e?',
    intent: 'AllCourses'
},{
    pattern: '\\b(?<course>.+)\\b\\scatalogo?u?e?',
    intent: 'AllCourses'
},{
    pattern: '\\b(help|menu|info)\\b',
    intent: 'Help'
```

```
},{
    pattern: '\\b(sample questions?)\\b',
    intent: 'SampleQuestions'
},{
    pattern : '\\b(?<greeting>bye|see you)',
    intent : 'Exit'
}];
```

Matcher.js

```
'use strict';
const patterns = require('../patterns');
const XRegExp = require('xregexp');

let createEntities = (str, pattern) => {
    return XRegExp.exec(str, XRegExp(pattern,"i"));
}

let matchPattern = (str, cb) =>{
    let getResult = patterns.find(item =>{
        if(XRegExp.test(str, XRegExp(item.pattern, "i"))){
            return true;
        }
    });

    if(getResult){
        return cb({
            intent: getResult.intent,
            entities: createEntities(str, getResult.pattern)
        });
    } else {
        return cb({});
    }
}

module.exports = matchPattern;
```

CourseApi.js

```
var Request = require("request");

COURSE_URL = "https://chatbot2018.azurewebsites.net/api/courses/";

const getCourseService = (id, callback) => {
    return new Promise((resolve, reject) => {
        Request.get(`${COURSE_URL}${id}`, {json:true}, (err,res,body) => {
            if(err){
                reject(err);
            }
            resolve(body);
        });
    });
}
module.exports = getCourseService;
```

ProfessorApi.js

```
var Request = require("request");

PROFESSOR_URL = "https://chatbot2018.azurewebsites.net/api/professors/";

const getProfessorService = (id, callback) => {
    return new Promise((resolve, reject) => {
        Request.get(`${PROFESSOR_URL}${id}`, {json:true}, (err,res,body) => {
            if(err){
                reject(err);
            }
            resolve(body);
        });
    });
}

module.exports = getProfessorService;
```

ProfessorParse.js

```
let officeHr = response => {
    if(response.lname){
        return `Office hours of Dr. ${response.lname} are ${response.officeHours}.`
    }
}

let emailId = response => {
    if(response.lname){
        return `Email of Dr. ${response.lname} is ${response.email}.`
    }
}

let phoneNo = response => {
    if(response.lname){
        return `Phone number of Dr. ${response.lname} is ${response.phone}.`
    }
}

let officeLocation = response => {
    if(response.lname){
        return `Office location of Dr. ${response.lname} is ${response.location}.`
    }
}

let allProfessors = response => {
    console.log("Here is the faculty list: ");
    for(let i = 0; i<response.length; i++) {
        if(response[i].lname){
            console.log( `${response[i].fname} ${response[i].lname},`);
        }
    }
}

module.exports = {
    officeHr,
    emailId,
    phoneNo,
    officeLocation,
    allProfessors
};
```

CourseParse.js

```
let courseTitle = response => {
    if(response.id){
        let course = response.number;
        return `The title of ${course.trim()} is ${response.name}.`
    }
}

let courseTime = response => {
    if(response.id){
        let courseId = response.number;
        return `Hours of ${courseId.trim()} are ${response.time}.`
    }
}

let classLocation = response => {
    if(response.id){
        let courseNo = response.number;
        return `The class location of ${courseNo.trim()} is ${response.location}.`
    }
}

let courseUnits = response => {
    if(response.id){
        let units = response.noOfUnits;
        return `Total number of units are ${units.toString()}.`
    }
}
let courseByProfessor = response => {
    if(response.id){
        let courseName = response.number.trim();
        return `Instructor of ${courseName} is ${response.professors.fname} ${response.professors.lname}`
    }
}

module.exports = {
    courseTitle,
    courseTime,
    classLocation,
    courseUnits,
    courseByProfessor
};
```

Fbeamer.js

```
'use strict';

const crypto = require('crypto');
const request = require('request');
const apiVersion = 'v2.8';

class FBeamer {
  constructor({ pageAccessToken, verifyToken, appSecret}) {
    try {
```

```
    if (pageAccessToken && verifyToken) {
      this.pageAccessToken = pageAccessToken;
      this.verifyToken = verifyToken;
      this.appSecret = appSecret;
    } else {
      throw "One or more tokens/credentials are missing!";
    }
  } catch (e) {
    console.log(e);
  }
}

registerHook(req, res) {
  const params = req.query;
  const mode = params['hub.mode'],
    token = params['hub.verify_token'],
    challenge = params['hub.challenge'];
    // if mode === 'subscribe' and token === verifytoken, then send back challenge
    try {
      if ((mode && token) && (mode === 'subscribe' && token === this.verifyToken)) {
        console.log("Webhook registered!");
        return res.send(challenge);
      } else {
        throw "Could not register webhook!";
        return res.sendStatus(200);
      }
    } catch(e) {
      console.log(e);
    }
}

verifySignature(req, res, buf) {
  return (req, res, buf) => {
    if(req.method === 'POST') {
      try {
        let signature = req.headers['x-hub-signature'];
        if(!signature) {
          throw "Signature not received";
        } else {
          let hash = crypto.createHmac('sha1', this.appSecret).update(buf, 'utf-8');
          if(hash.digest('hex') !== signature.split("=")[1]) {
            throw "Invalid signature!";
          }
        }
      } catch (e) {
        console.log(e);
      }
    }
  }
}

incoming(req, res, cb) {
  res.sendStatus(200);
  if(req.body.object === 'page' && req.body.entry) {
    let data = req.body;
    data.entry.forEach(pageObj => {
      if(pageObj.messaging) {
        pageObj.messaging.forEach(messageObj => {
          console.log(messageObj);
```

```
          if(messageObj.postback) {
            // Handle postbacks
          } else {
            // Handle messages
            return cb(this.messageHandler(messageObj));
          }
        });
      }
    });
  }
}

messageHandler(obj) {
  let sender = obj.sender.id;
  let message = obj.message;

  if(message.text) {
    let obj = {
      sender,
      type: 'text',
      content: message.text
    }

    return obj;
  }
}
sendMessage(payload) {
  return new Promise((resolve, reject) => {
    request({
      uri: `https://graph.facebook.com/${apiVersion}/me/messages`,
      qs: {
        access_token: this.pageAccessToken
      },
      method: 'POST',
      json: payload
    }, (error, response, body) => {
      if (!error && response.statusCode === 200) {
        resolve({
          mid: body.message_id
        });
      } else {
        reject(error);
      }
    });
  });
}

txt(id, text, messaging_type = 'RESPONSE') {
  let obj = {
    messaging_type,
    recipient: {
      id
    },
    message: {
      text
    }
  }

  return this.sendMessage(obj);
```

```
  }

  img(id, url, messaging_type = 'RESPONSE') {
    let obj = {
      messaging_type,
      recipient: {
        id
      },
      message: {
        attachment: {
          type: 'image',
          payload: {
            url
          }
        }
      }
    }

    return this.sendMessage(obj);
  }
}

module.exports = FBeamer;
```

Config.js

```
if(process.env.NODE_ENV === 'production') {
  module.exports = {
    fb: {
      pageAccessToken: process.env.pageAccessToken,
      verifyToken: process.env.verifyToken,
      appSecret: process.env.appSecret
    }
}
```

REFERENCES

[1] Goodwin, R., Lee, J., and Stanoi, G. “Relational Database Systems for Large Scale Ontology Management” in *Proceeding of CIDR Conference.* 2005 pp.150-195

[2] Abu Shawar, B. and Atwell, E. “Chatbots: Are They Really Useful?” *in Proceedings of LDV-Forum* 2007. pp.31-50.

[3] ALICE, (2011). “A.L.I.C.E. Artificial Intelligence Foundation [online]. Available from: http://alice.pandorabots.com. 2007

[4] Freese, E. “Enhancing AIML Bots Using Semantic Web Technologies,” *in Proceeding of Extreme Markup Languages,* 2007

[5] Falbo, R., Menezes, C., and Rocha, A. “ A Systematic Approach for Building Ontologies,” *in Proceedings of 6th IberoAmerican Conference on AI, number LNCS1484* Lecture *Notes in Artificial Intelligence* Lisbon, Portugal, 2005 pp.349-360.

[6] Falbo, R., Guizzardi, G., and Duarte, K., “An Ontological Approach to Domain Engineering,” *in Proceedings of 14th International Conference on Software Engineering and Knowledge Engineering (SEKE’02)* Ischia, Italy, 2002, pp.351-358.

[7] Liao, L., Qu, Y., and Leung, H. “A Software Process Ontology and Its Application,” *in Proceedings of Workshop on Sematic Web Enable Software Engineering (SWESE)* Galway, Ireland, 2005.

[8] Z. Zhang, J. Jiang, X. Liu, R. Lau, H. Wang, R. Zhang. “A Real Time Binary Pattern Matching Scheme for Stock Time Series,” *Australian Computer Society, Inc*. 2010, vol. 104, pp. 161– 170

[9] Corradini, F., Grabmayer, C. “A Characterization of Regular Expressions under Bisimulation” *Journal of the ACM*, 2010, vol. 54, pp. 61– 100

[10] Yang, X., Qiu, T., Wang, B. “Improving Regular-Expression Matching in Strings,” *ACM Transactions on Database System,* 2016, vol. 40, pp. 151–170

[11] Jain, M., Kumar, P., Kota, R., Patel, S. “Evaluating and Informing the Design of Chatbots,” *Interacting with Conversational Agents,* 2018, vol. 12, pp. 895–906

[12] O’ Brien Chris. 2016. “Facebook Messenger chief says platform’s 34,000 chatbots are finally improving user experience. (2016).” From http://venturebeat.com/2016/11/11/facebookmessenger-chief-saysplatforms-34000/-chatbots-are-finally-improving-user-experience/