



Department of Computer Science Engineering

Symbiosis Institute of Technology

(A constituent member of Symbiosis International University)

A PROJECT REPORT ON

“ CHATBOT : CYRUS ”

Submitted by:

NAME	PRN
Himank Jain	18070122027

Faculty Mentor: Shubham Londhe

TABLE OF CONTENTS

1. Abstract	4
2. Imports and Libraries used	5
3. Database and connection	7
4. Flowchart and Class Diagram.....	8-9
5. Future Scope of project	9
6. Understanding of project	11
7. Project Highlights	12

YOU: Open the Final Report on Chatbot.

CYRUS: Will do!



ABSTRACT

Chatbots, or conversational interfaces as they are also known, present a new way for individuals to interact with computer systems. Traditionally, to get a question answered by a software program involved using a search engine, or filling out a form. A chatbot allows a user to simply ask questions in the same manner that they would address a human. The most well known chatbots currently are voice chatbots: Alexa and Siri. However, chatbots are currently being adopted at a high rate on computer chat platforms. A simple chatbot can be created by loading an FAQ (frequently asked questions) into chatbot software. The functionality of the chatbot can be improved by integrating it into the organization's enterprise software, allowing more personal questions to be answered, like "What is my balance?", or "What is the status of my order?".

Chat bot can be segregated into 2 broad categories.

- ❖ Using AI to implement multiple chatbot functionalities
- ❖ Using mapping technique to correlate with what the user has given as input to find out the apt answer from the pre-existing set of database in the form of a Dynamic memory allocation like linked list or even in static data structures like arrays.

In our project we have implemented the second approach to make our chatbot run successfully and answer the users query. We have used String Array to store all the possible inputs that user can give to us and mapped it into another array that holds the answer to those questions. We use a simple analogous equation code to find out the length match comparison of the query given as input, match it with our question sets array on the basis of length, keywords used and then find out whether we have the question feeded into our database or not. If we do, then with another analogous code equation we find the answer associated with the query entered by the user and display.

Our chat bot also has multiple functionalities such as:

- ❖ The user can access the home windows applications like calculator and even search for anything on the internet by just writing phrases like: "open calc" or "search testla".
- ❖ We have also given our chatbot a voice to make the user feel that the chatbot is more alive than just lines of code.
- ❖ We have also include a contingency for when the user's query cannot be answered by the chatbot, in such cases we store the query given as an input by the user to us in a text file and ask for their permission to save their entries so that we can diversify the range of responsiveness of the chat bot.

IMPORTS AND LIBRARIES USED:

1.Freetts

FreeTTS is an open source speech synthesis system written entirely in the Java programming language. It is based upon Flite. FreeTTS is an implementation of Sun's Java Speech API

import com.sun.speech.freetts contains the implementation of the FreeTTS synthesis engine.

FreeTTSSpeakable is an interface. Anything that is a source of text that needs to be spoken with FreeTTS is first converted into a FreeTTSSpeakable.

import com.sun.speech.freetts.Voice - The Voice is the central processing point for FreeTTS. The Voice takes as input a FreeTTSSpeakable, translates the text associated with the FreeTTSSpeakable into speech and generates audio output corresponding to that speech. A Voice will accept a FreeTTSSpeakable via the Voice.speak method and process it as follows:

- The Voice converts a FreeTTSSpeakable into a series of Utterances. The rules for breaking a FreeTTSSpeakable into an Utterance is generally language dependent. For instance, an English Voice may chose to break a FreeTTSSpeakable into Utterances based upon sentence breaks.
- As the Voice generates each Utterance, a series of UtteranceProcessors processes the Utterance. Each Voice defines its own set of UtteranceProcessors. This is the primary method of customizing Voice behavior.
- Once all Utterance processing has been applied, the Voice sends the Utterance to the AudioOutput UtteranceProcessor. The AudioOutput processor may run in a separate thread to allow Utterance processing to overlap with audio output, ensuring the lowest sound latency possible.

import com.sun.speech.freetts.VoiceManager - The VoiceManager is the central repository of voices available to FreeTTS. Provides access to voices for all of FreeTTS. There is only one instance of the VoiceManager. Each call to getVoices() creates a new instance of each voice.

2.Awt

import java.awt.* - Contains all of the classes for creating user interfaces and for painting graphics and images.

java.awt.event - Provides interfaces and classes for dealing with different types of events fired by AWT components.

import java.awt.event.ActionEvent - A semantic event which indicates that a component-defined action occurred. This high-level event is generated by a component (such as a Button) when the component-specific action occurs (such as being pressed). The event is passed to every every ActionListener object that registered to receive such events using the component's addActionListener method.

import java.awt.event.ActionListener - The listener interface for receiving action events. The class that is interested in processing an action event implements this interface, and the object created with that class is registered with a component, using the component's addActionListener method. When the action event occurs, that object's actionPerformed method is invoked.

import java.awt.event.KeyListener - The listener interface for receiving keyboard events (keystrokes). The class that is interested in processing a keyboard event either implements this interface (and all the methods it contains) or extends the abstract KeyAdapter class (overriding only the methods of interest). A keyboard event is generated when a key is pressed, released, or typed.

import java.awt.event.KeyEvent - An event which indicates that a keystroke occurred in a component. This low-level event is generated by a component object (such as a text field) when a key is pressed, released, or typed (pressed and released). The event is passed to every KeyListener or KeyAdapter object which registered to receive such events using the component's addKeyListener method.

3.Swing

import javax.swing.* - Provides a set of "lightweight" (all-Java language) components that, to the maximum degree possible, work the same on all platforms.

import javax.swing.JFrame – provides an extended version of java.awt.Frame that adds support for the JFC/Swing component architecture.

import javax.swing.JPanel – provides JPanel which is a generic lightweight container.

import javax.swing.JTextArea - provides a JTextArea which is a multi-line area that displays plain text.

import javax.swing.JScrollPane - Provides a scrollable view of a lightweight component.

4. Java Standard libraries

import java.lang.Math - The class Math contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions. In our project we have used Math.random() method to generate random answer from matching set.

import java.util.* - It contains the collections framework, legacy collection classes, event model, date and time facilities, internationalization, and miscellaneous utility classes (a string tokenizer, a random-number generator, and a bit array).

import java.io.* - This package provides for system input and output through data streams, serialization and the file system. Unless otherwise noted, passing a null argument to a constructor or method in any class or interface in this package will cause a NullPointerException.

Database and Connections

1. Database

A multidimensional array is used as the database for the ChatBot.

This ChatBot uses a two-dimensional String Array to store possible inputs and outputs.

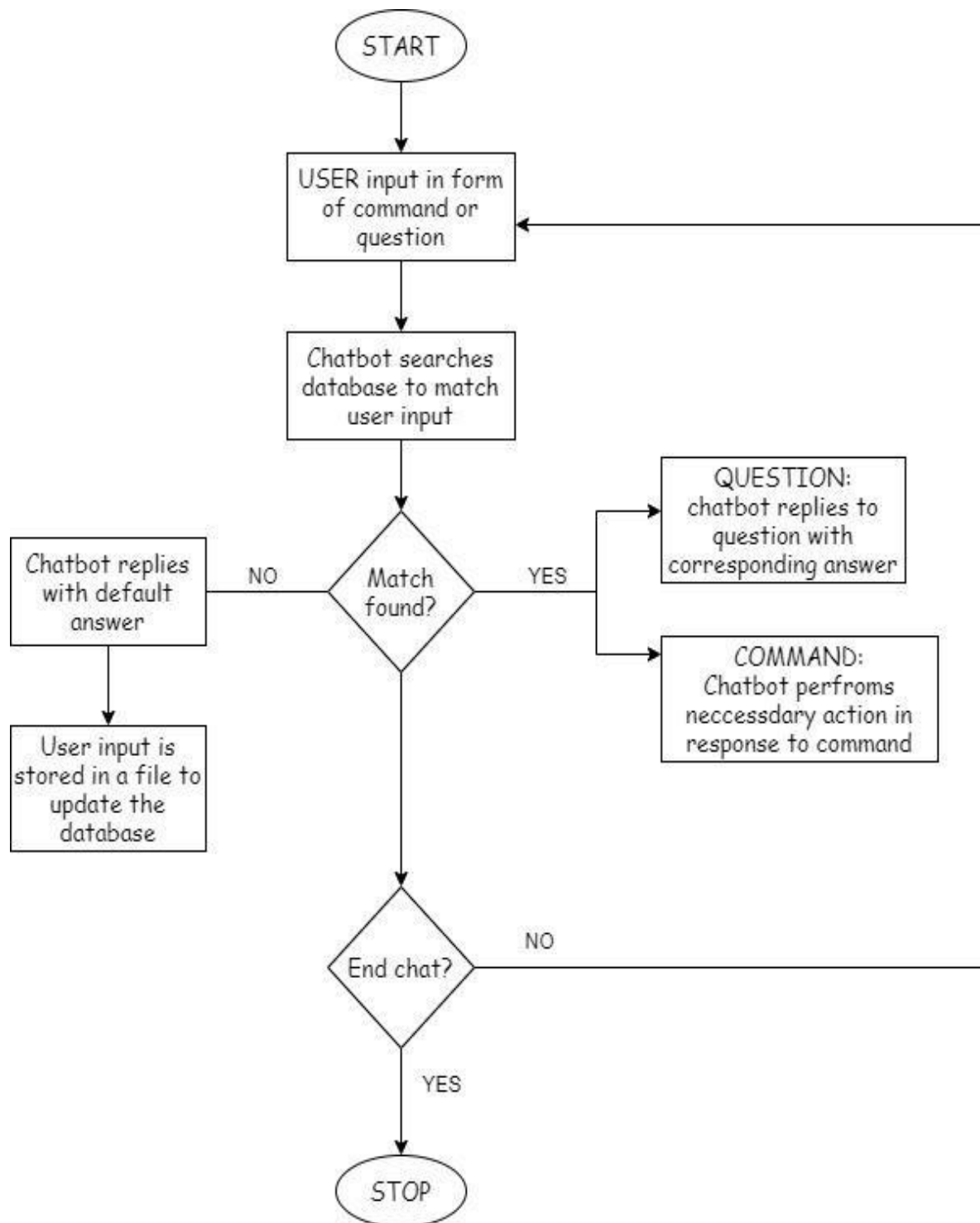
The two-dimensional array or matrix acts as a spreadsheet, containing a set of questions and their corresponding answers, serving as the brain of the ChatBot. The advantage of using a 2D array is that you can structure your knowledge using an ontology of your own choosing, and you don't have to license a database software. This also makes the code run faster. Another advantage includes the security of database. Using a 2D array, we don't have to worry about SQL injections which make the application vulnerable. The program is secured using a simple virtue of JAVA which makes it platform independent. A class file can be executed on any OS with JVM installed. The database cannot be corrupted because the user will only have access to the class file in order to execute the application. Only the admin will have access to the JAVA file containing database. The only disadvantage of using 2D array, is that whenever the database is updated the program need to recompile.

2. Connections

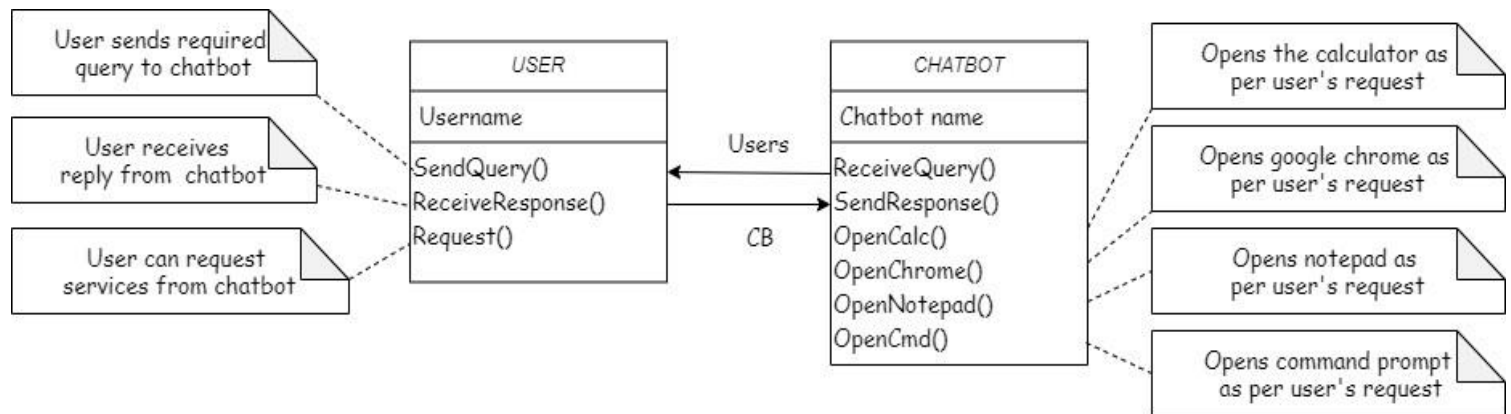
This database is linked to the Program using Input and Output Streams implemented in the method keyPressed. The method keyPressed receives the user input and is supposed to return the correct output. When the user enters a question, the input stream comes into play. The entered question is then normalised by using the method trim which removes the leading and trailing spaces from the question. This is followed by removing punctuation marks from the end of the string. By using a while loop, the question is verified, to check if the content of the field matches with the database. Once the question entered is marked valid, the output stream helps the ChatBot to find the answer to the asked question. It uses the method Math.random to select an answer from a set of possible answers. If the loop reaches the end of the array, it is supposed to return the default answer.

In this way, the database is connected to the program which provides functionality to the Bot.

FLOWCHART



CLASS DIAGRAM



FUTURE SCOPE

- ❖ With our program being currently implemented on java using the swing UI, we plan to expand it to make our chatbot compatible with:
 - Android / iOS devices.
 - Execute it on a working website.
- ❖ We currently use array data structures to store our queries and their respective responses. We plan on rebuilding that segment of the code by using RDBMS to store both the queries and the replies so that we can easily access the database and also make the code immutable to the user.
- ❖ Our project can be incorporated with latest technology like Machine learning or Artificial intelligence to introduce several creative features like sentimental analysis where chatbot will reply according to the mood of user as interpreted from his/her input

UNDERSTANDING OF PROJECT

Before designing the structure of the chatbot, it is important to decide a goal. The goal of the bot could be service based assistance, performing certain tasks or simply leisure, based on the goal we can decide the database and the APIs to be used by the bot.

We can categorize interactions offered by a bot into two types, namely structured and unstructured.

A structured interaction consists of menus, forms and options, that lead the chat forward. But this chatbot employs an unstructured form of interaction which follows freestyle plain text. This unstructured type is more suited for informal conversations which leads to personalisation of the chatbot i.e. user can type commands/questions in the form of plain text which makes the application more modular. A chatbot consists of a core and an interface accessing the core, this is similar to a server client relation used in several chat applications, in this case the core acts as the server and the client is provided access through interface.

CORE

The core lies in the database, in this case, the 2D array. The database consists of a matrix to store questions with their corresponding answers, whereas the rest of the core consists of methods required to interpret as well as match the input to the database in order to provide accurate results.

A chatbot should not only answer general questions but also perform service-based tasks. In order to perform a certain task, the bot should have access to required libraries and methods to implement the task. These libraries and methods are embedded in the core of the chatbot and are accessed whenever the user requests for them.

For Example, if the user asks the bot to open a web browser, the bot identifies the keywords “web” and “open” and employs Runtime Classes in order to interact with the OS to launch the application.

Another part of the core consists of the Text-to-Speech API, which uses Speech Recognition and Speech Synthesis, to get audio feedback from the bot, whenever the user commands the bot to speak, the text from the output field is fed to system and read out loud by the bot.

INTERFACE

The interface here is a standalone console application that can be employed by the user for chatting or conversation. The application in the interface side interacts with the core, and initializes the bot i.e. makes it operational. The interface can be expanded more over as the user needs it.

The interface also plays a crucial role in the enhancement of the database. If a user enters a question which requires complex thinking or requests a task which is unknown to the bot, it can be stored by creating a log in the form of a file, which can later be embedded into the database to increase the functionality of the bot.

In this way, the core and the interface combine together in perfect harmony to provide a functional and efficient chatbot for personalised use.

PROJECT HIGHLIGHTS

CODE SNIPPETS

1) Components of frame

```
user.java x
class ChatBot extends JFrame implements KeyListener, ActionListener{

    JPanel p=new JPanel();
    JButton b = new JButton("SPEAK");
    JTextArea dialog=new JTextArea(30,50);
    JLabel label=new JLabel("Hello I am Cyrus the Chatbot !!");
    JTextArea input=new JTextArea(("Enter text here"),1,50);
    JTextArea output=new JTextArea(1,50);
    JScrollPane scroll=new JScrollPane(
        dialog,
        JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED,
        JScrollPane.HORIZONTAL_SCROLLBAR_NEVER
    );

    String[][] chatBot={

        {"hi","hello","hey"},
        {"Hi","Hello","Hey"},
        {"how are you","how r you","how are u","how are u"},
        {"good","doing well"},
        {"are you cyrus","will you answer my questions","will you help me"},
        {"yes","yes!!"},
        {"tell me about coronavirus","what is the current topic","what is happening around the world"},
        {"Coronavirus disease (COVID-19) is an infectious disease caused by a "+"+"\n"+"t"+"newly discovered"},
        {"can you give me stats of covid-19 around the world","how many cases of coronavirus","how many deaths"},
        {"YES! sure:"+"n"+"t"+"World:"+"n"+"t"+"Total cases:1,203,428"+"t"+"Total deaths:64,754"},
        {"n"+"t"+"USA:"+"n"+"t"+"Total cases:311,637"+"t"+"Total deaths:8,454"},
        {"n"+"t"+"Spain:"+"n"+"t"+"Total cases:126,168"+"t"+"Total deaths:11,947"},
        {"n"+"t"+"Italy:"+"n"+"t"+"Total cases:124,632"+"t"+"Total deaths:15,362"},
        {"n"+"t"+"Germany:"+"n"+"t"+"Total cases:96,092"+"t"+"Total deaths:1,444"},
        {"n"+"t"+"China:"+"n"+"t"+"Total cases:81,669"+"t"+"Total deaths:3,329"},
        {"n"+"t"+"India:"+"n"+"t"+"Total cases:3,588"+"t"+"Total deaths:99 "}
```

```
user.java x
public ChatBot() {
    super("Chat Bot");
    setSize(600, 595);
    add(p, BorderLayout.CENTER);
    setResizable(false);
    setDefaultCloseOperation(EXIT_ON_CLOSE);

    Image icon = Toolkit.getDefaultToolkit().getImage("F:\\\\Chatbot.jpg");
    setIconImage(icon);

    dialog.setEditable(false);
    input.addKeyListener(this);
    output.addKeyListener(this);

    p.add(label);
    label.setForeground(Color.white);

    p.add(scroll);
    p.add(b);
    p.add(input);
    p.add(output);
    p.setBackground(new Color(0,0,153));
    add(p);

    setVisible(true);
    b.addActionListener(this);
}
```

2) Use of Freetts by speak button

```
user.java x
@Override
public void actionPerformed(ActionEvent e) {

    if (e.getSource() == b) {
        Voice voice;
        voice = VoiceManager.getInstance().getVoice("kevin");
        if (voice != null) {
            voice.allocate();
        }
        try {
            voice.setRate(130);
            voice.setPitch(120);
            voice.setVolume(7);
            voice.speak(output.getText());

        } catch (Exception e1) {
            e1.printStackTrace();
        }
    }
}

public void keyPressed(KeyEvent e) {
    if (e.getKeyCode() == KeyEvent.VK_ENTER) {
        input.setEditable(false);
        //-----grab quote-----
        String quote = input.getText();
        input.setText("");
        addText("<-->You:\t"+quote);
        quote = quote.trim();
        while (
```

3) Key pressed method

```
user.java x
while(
    quote.charAt(quote.length()-1)=='!' ||
    quote.charAt(quote.length()-1)=='.' ||
    quote.charAt(quote.length()-1)=='?'
){
    quote=quote.substring(0,quote.length()-1);
}
quote=quote.trim();
byte response=0;
int j=0;
while(response==0){
    if(inArray(quote.toLowerCase(), chatBot[j*2])){
        response=2;
        int r=(int)Math.floor(Math.random()*chatBot[(j*2)+1].length);
        addText("\n-->Cyrus:\t"+chatBot[(j*2)+1][r]);
        output.setText(chatBot[(j*2)+1][r]);
        try{
            if(quote.toLowerCase().contains("calc")){
                Runtime.getRuntime().exec("calc");
            }
            if(quote.toLowerCase().contains("note")){
                Runtime.getRuntime().exec("notepad");
            }

            if(quote.toLowerCase().contains("chrome") || quote.toLowerCase().contains("google") || quote
            quote.toLowerCase().contains("web") || quote.toLowerCase().contains("internet")){

                Runtime rt = Runtime.getRuntime();
                rt.exec("C:\\Program Files (x86)\\Google\\Chrome\\Application\\chrome.exe");
            }
            if(quote.toLowerCase().contains("command prompt") || quote.toLowerCase().contains("cmd")){
                Runtime.getRuntime().exec("cmd");
            }
        }
    }
}
```

```
user.java x
    }
    j++;
    if(j*2==chatBot.length-1 && response==0){
        response=1;
    }
}

//-----default-----
if(response==1){
    int r=(int)Math.floor(Math.random()*chatBot[chatBot.length-1].length);
    addText("\n-->Cyrus:\t"+chatBot[chatBot.length-1][r]);
}
addText("\n");
addText("\n");
}

}

public void keyReleased(KeyEvent e){
    if(e.getKeyCode()==KeyEvent.VK_ENTER){
        input.setEditable(true);
    }
}
```

4. Key typed and inArray method

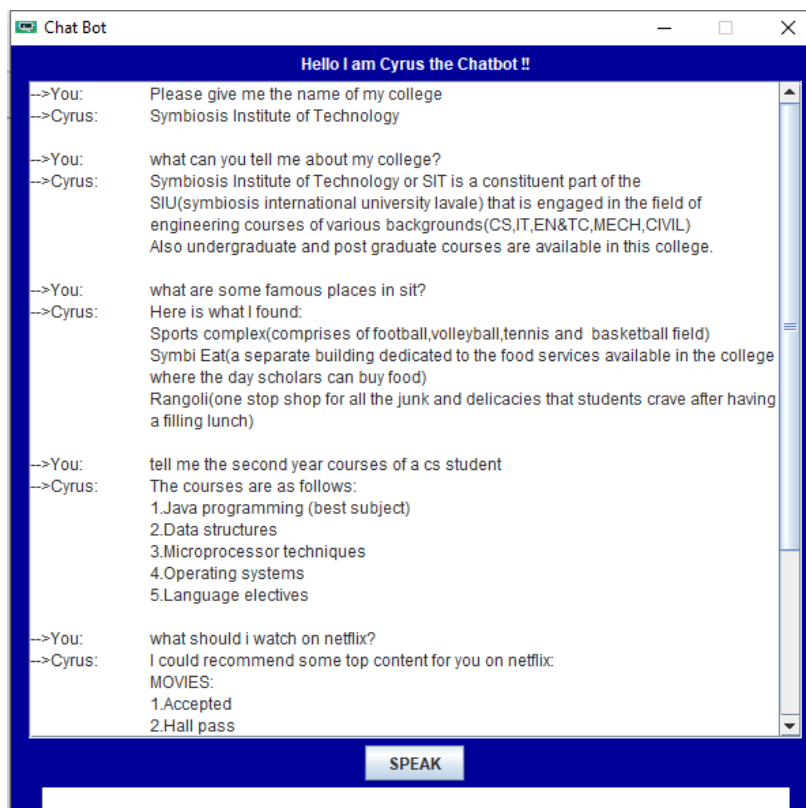
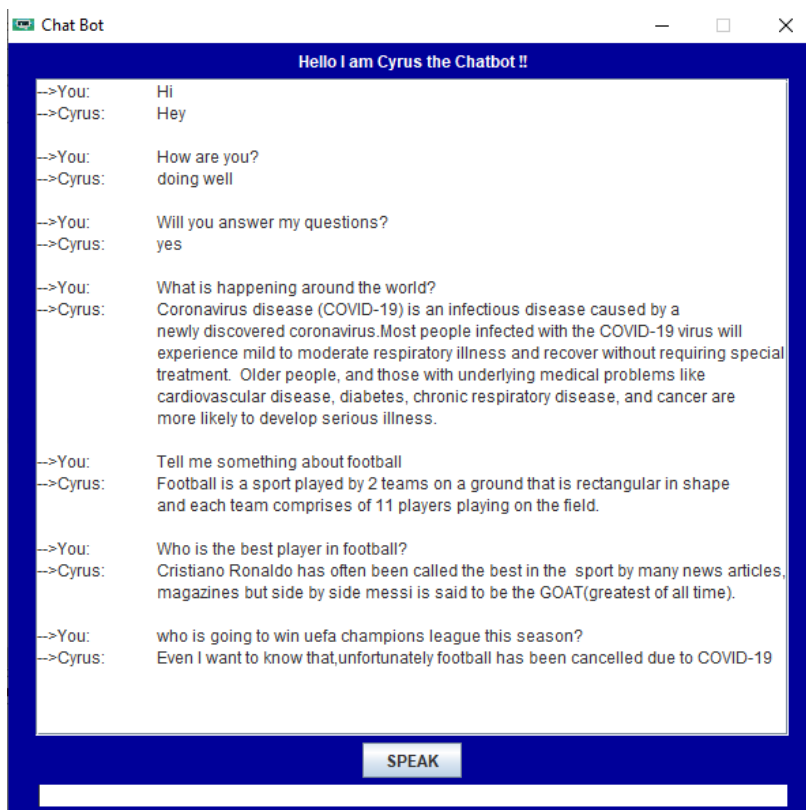
```
public void keyTyped(KeyEvent e){}

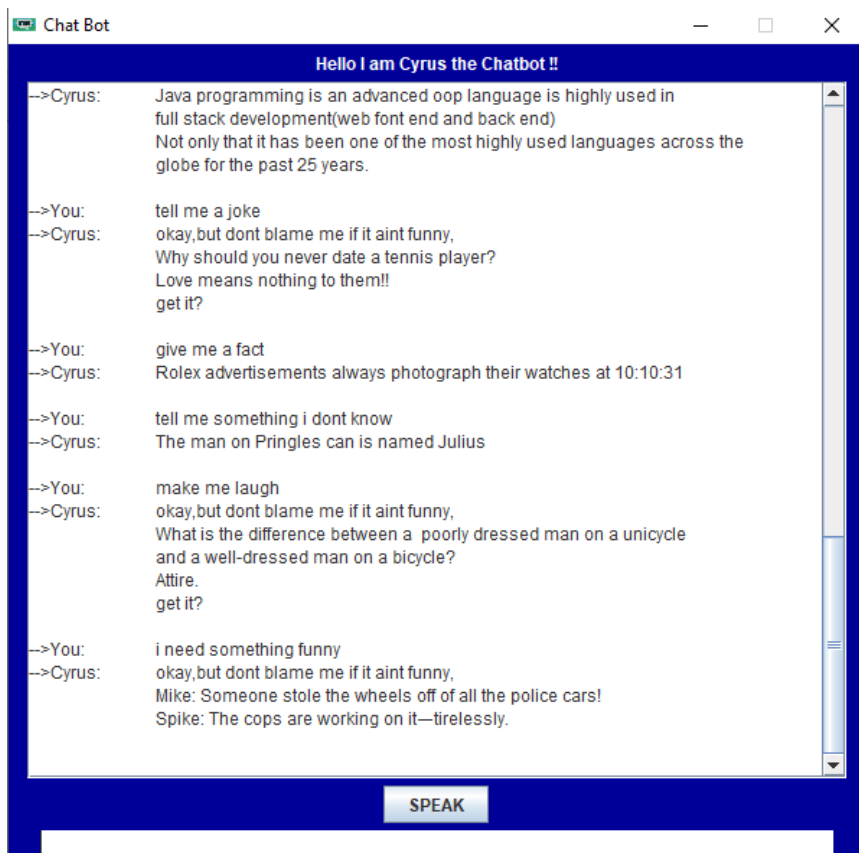
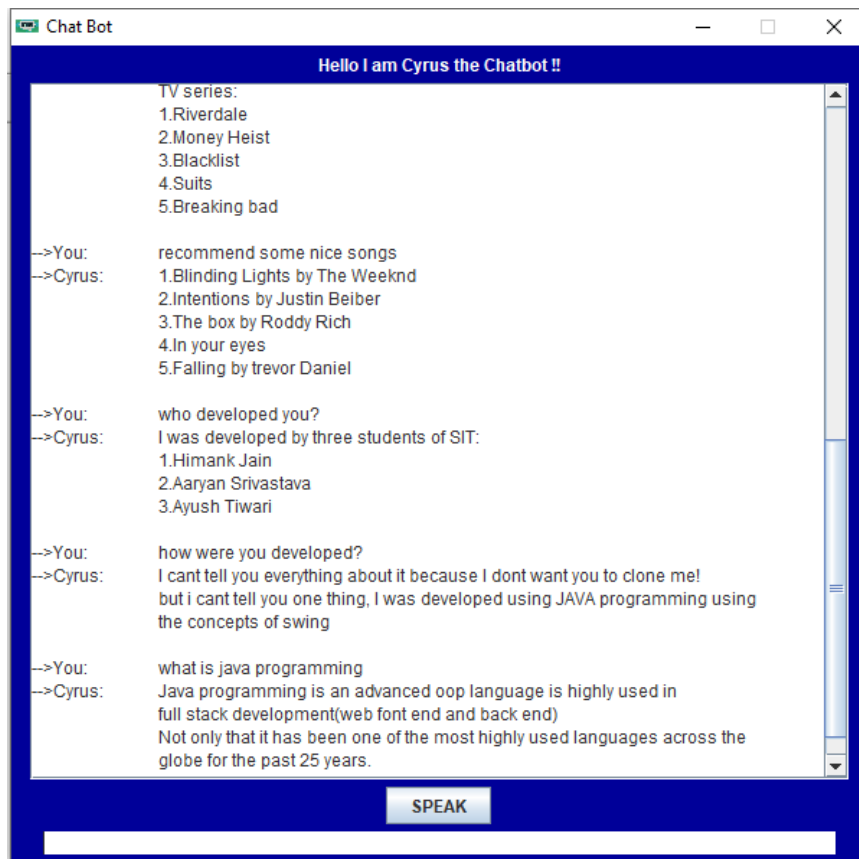
public void addText(String str){
    dialog.setText(dialog.getText()+str);
}

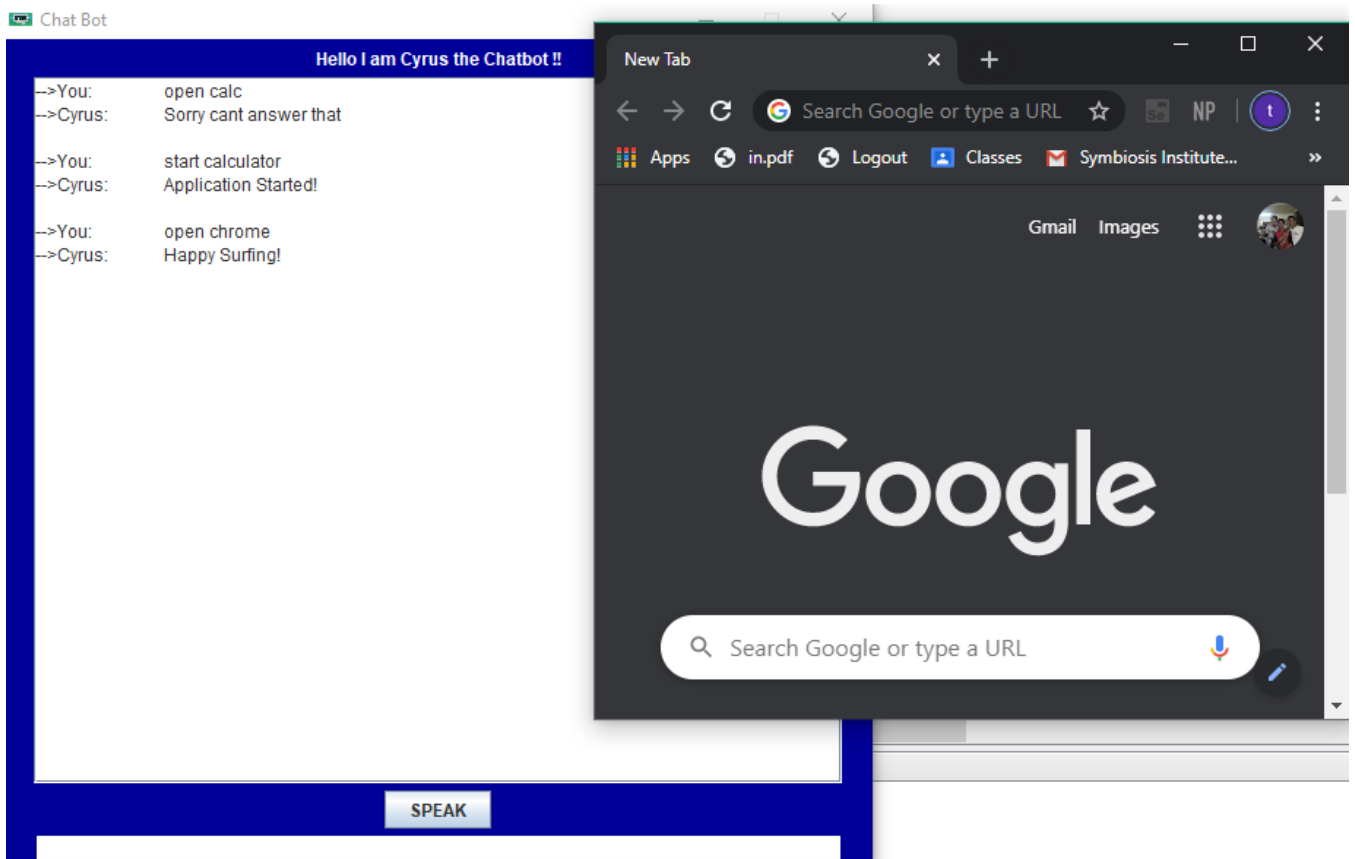
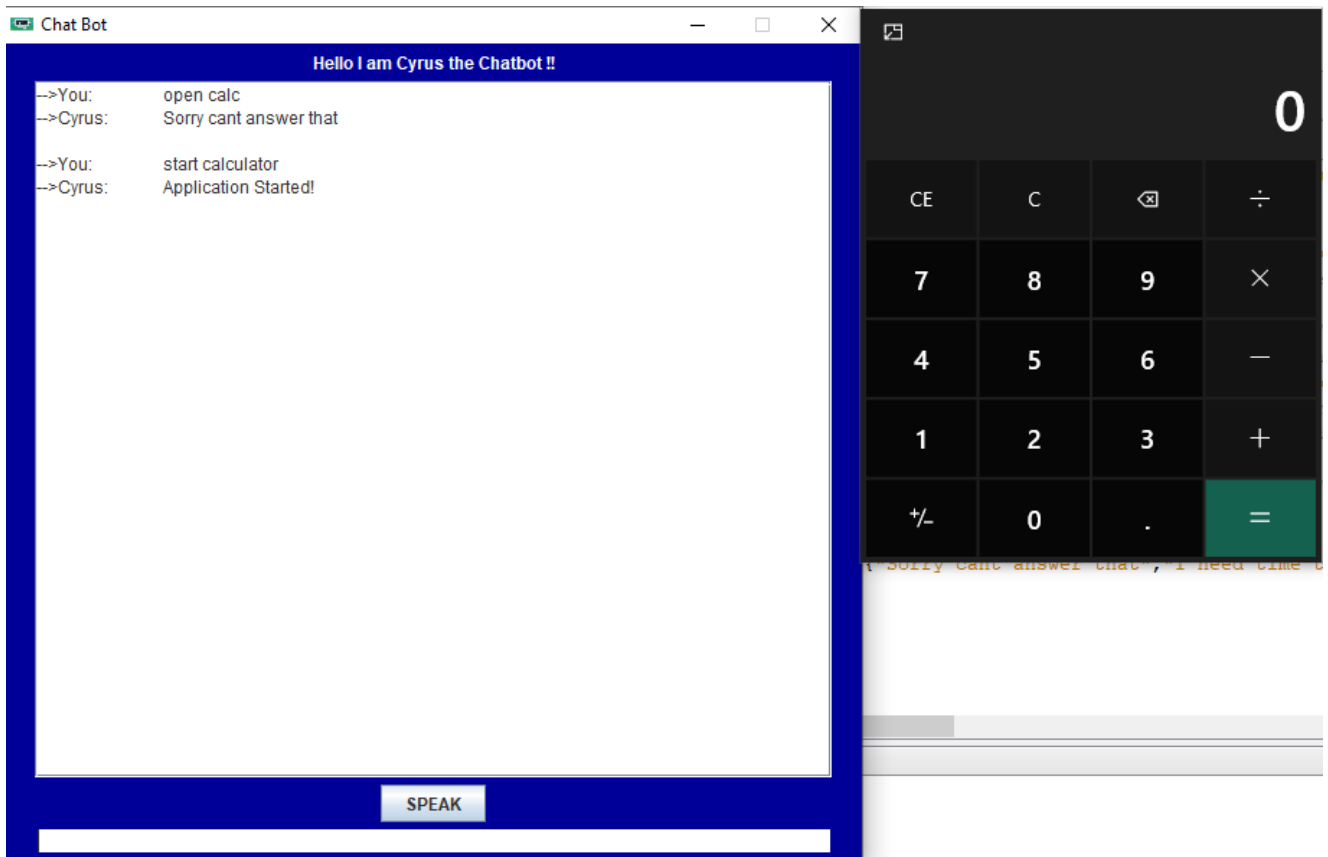
public boolean inArray(String in,String[] str){
    boolean match=false;
    for(int i=0;i<str.length;i++){
        if(str[i].equals(in)){
            match=true;
        }
    }
    return match;
}

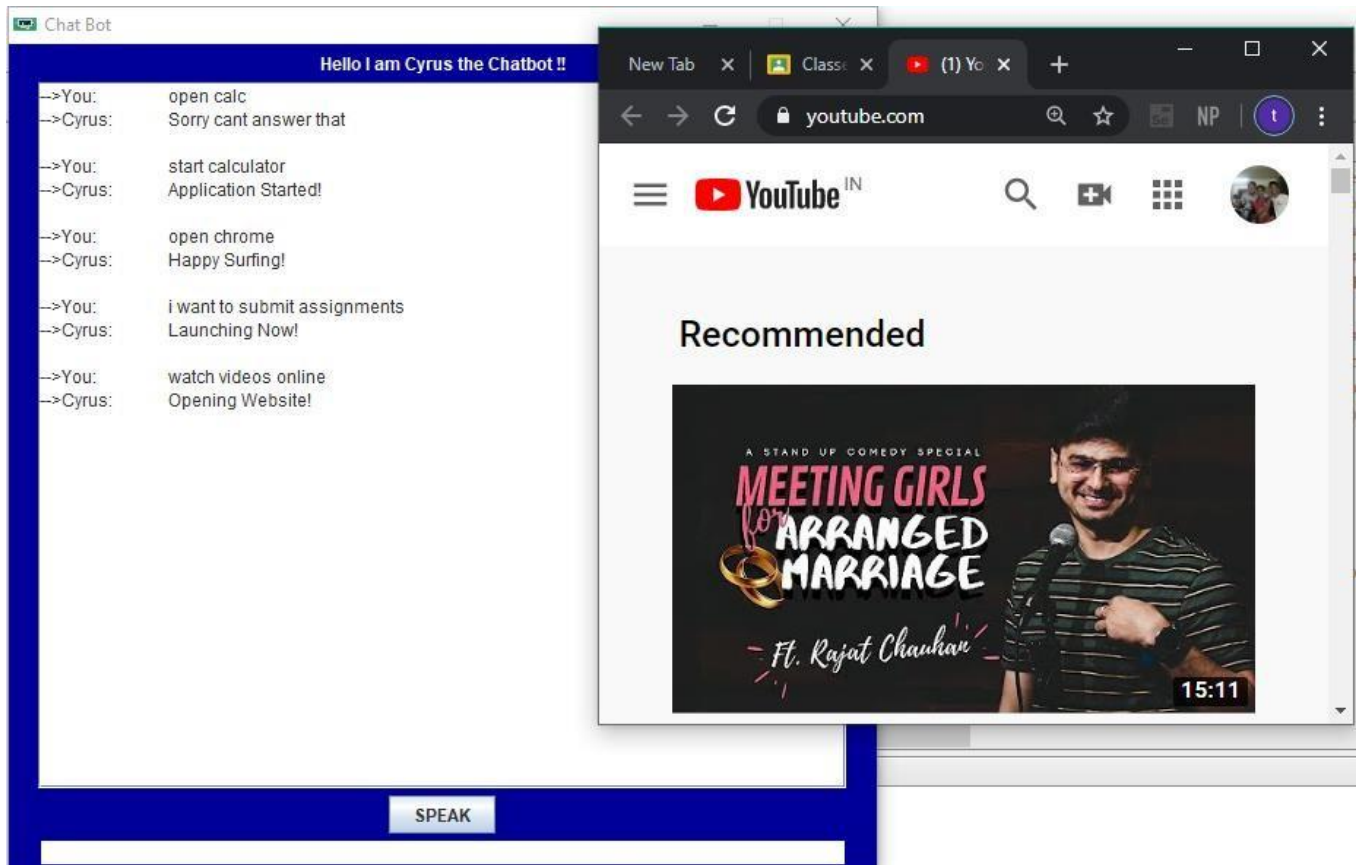
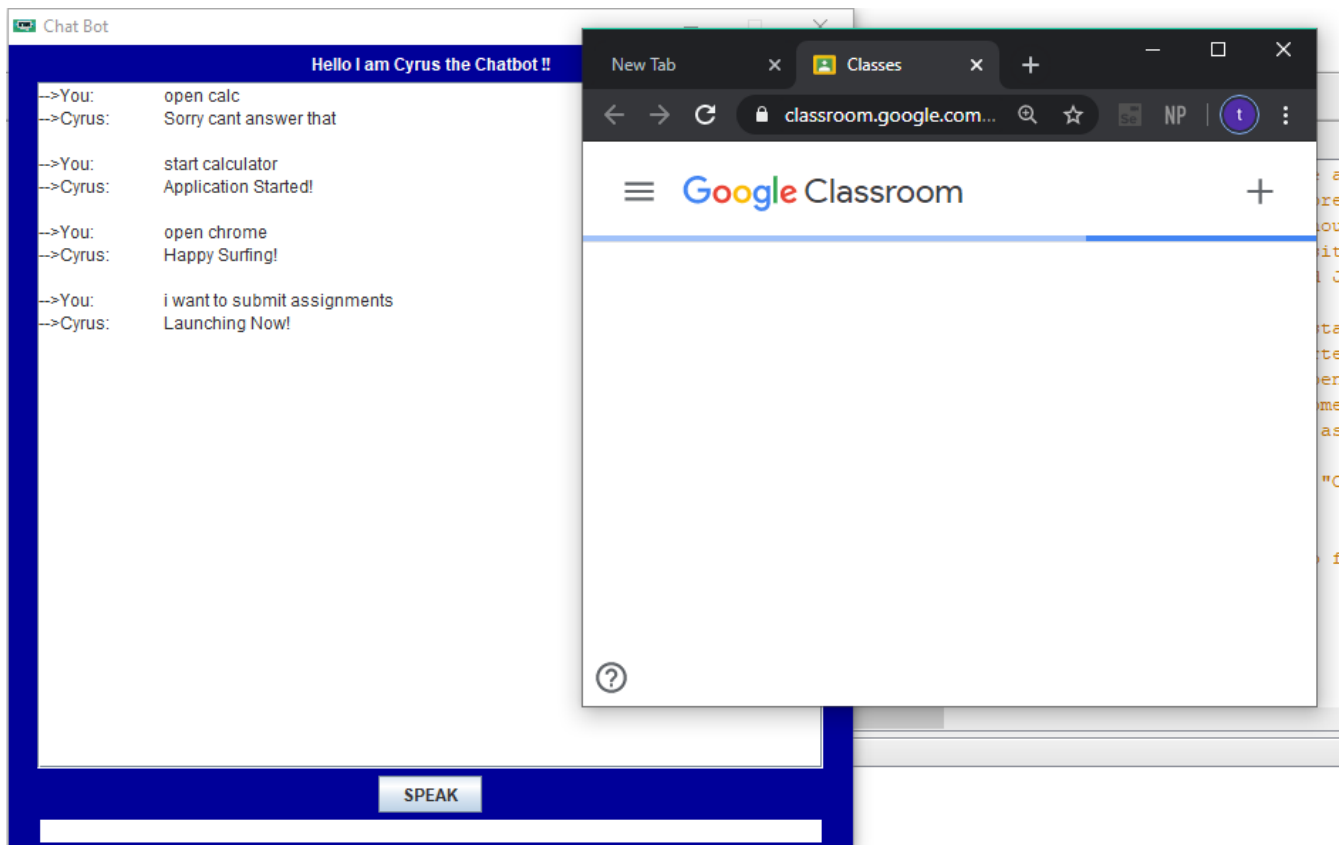
class user
{
    public static void main(String[] args){
        new ChatBot();
    }
}
```

SCREESHOTS

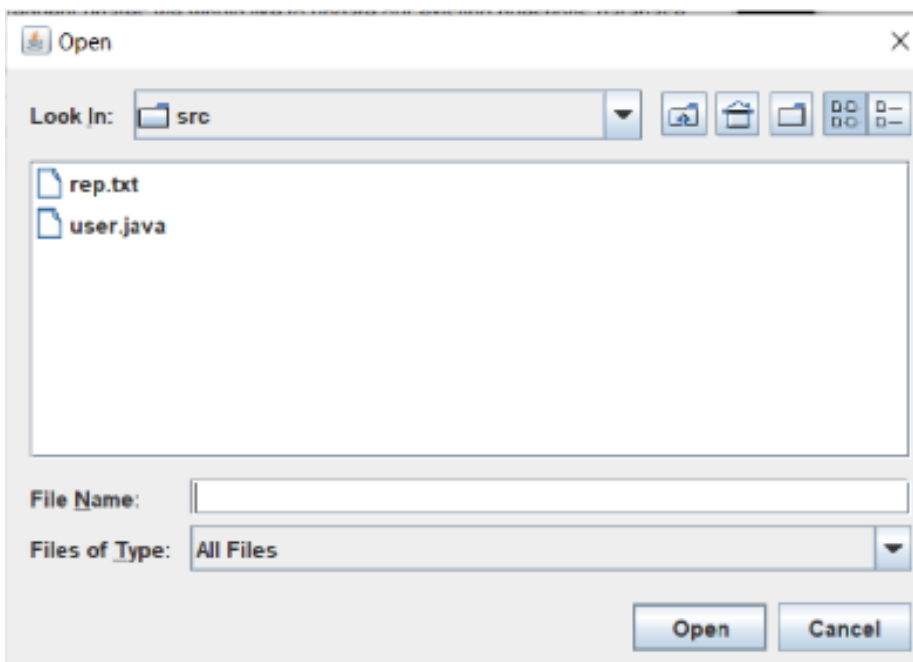




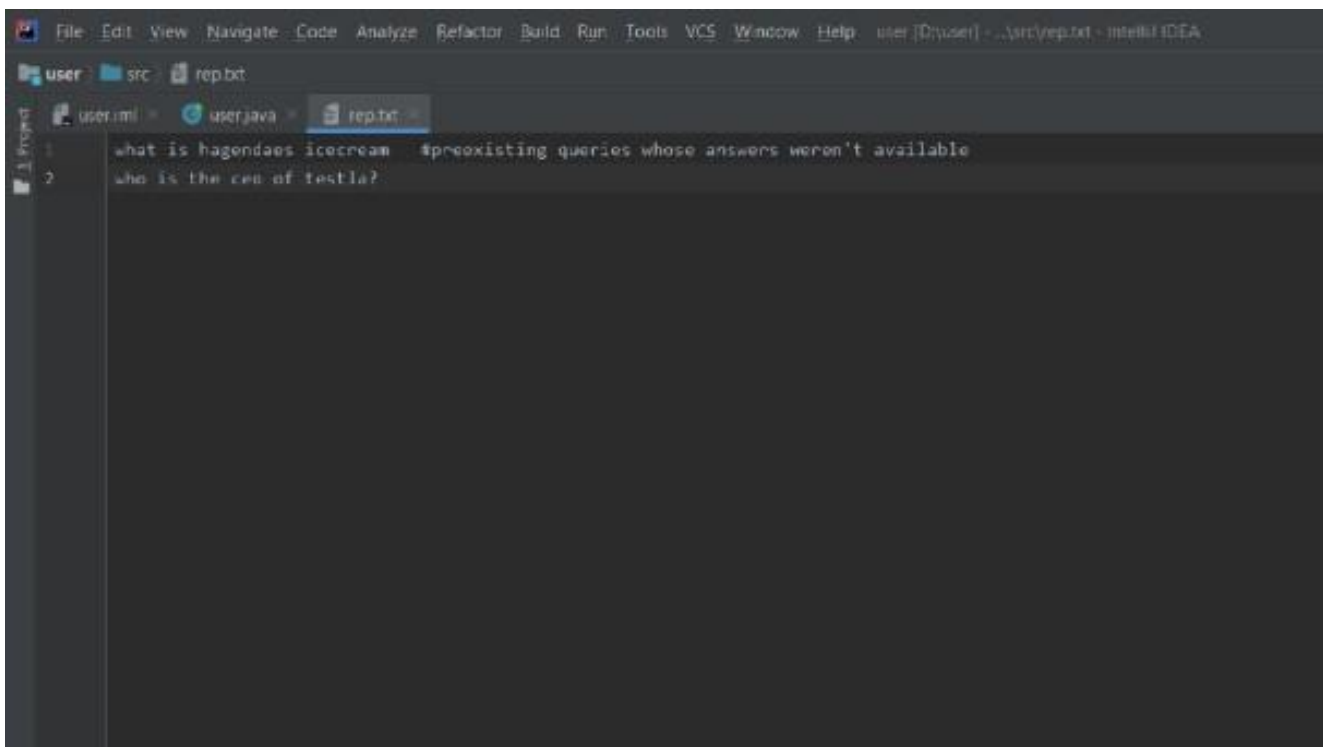




-->You: tell me a joke
-->Cyrus: okay,but dont blame me if it aint funny,
Doctor:I'm sorry but you suffer from a terminal illness and have only 10 to live.
Patient:What do you mean, 10? 10 what? Months? Weeks?!
Doctor: Nine.
-->You: fastest car on the planet
-->cyrus sorry but i could not find any answers in my repositories
-->cyrus for frequent updates we would like to update our existing questions database
and therefore we will saving your query in our database
writing successfull,we will get back to you shortly!



FILE



The screenshot shows the IntelliJ IDEA interface. The top menu bar includes File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, and Help. The title bar indicates the current file is user [D:\user] - \src\rep.txt - IntelliJ IDEA. The breadcrumb navigation shows user > src > rep.txt. The file explorer on the left shows a project structure with a file named rep.txt. The editor window displays the content of rep.txt, which contains two lines of text: "what is hagendas icecream #preexisting queries whose answers weren't available" on line 1 and "who is the ceo of tesla?" on line 2.

```
1 what is hagendas icecream #preexisting queries whose answers weren't available
2 who is the ceo of tesla?
```