**Project 2: Customer Support System**

**UML Design of the Application**



**Interdependencies of the Objects:**

(Depends on = “->”)

ExecutionFlow -> FlowMonitor – Using FlowMonitor object we are executing the program

Ticket (PhoneCall, Inperson, Email) -> FlowMonitor – using FlowMonitor object we are creating tickets.

Ticket (PhoneCall, Inperson, Email) ->Transcript – In order to create transcript we need ticket.

**Flow of the program:**

* The Console application developed for managing customer tickets starts with ExecutionFlow (main method) which will prompt the user to choose between customer representative, Customer service analyst or to the exit the program.
* Customer Service Representative:

1. The ExecutionFlow method hits the FlowMonitor(Object) which ultimately does the following:
2. The First step of FlowMonitor object if we choose the Customer Service Representative (option 1) calls the CreateTicket method which in turn asks the users to select among the three below options:

* PhoneCall
* Inperson
* Email

1. When the user selects any one of the above mode of communication, series of customer and service information are provided to the system which are stored for analysis purpose - a Transcript(Object) saves all the ticket and conversation for each ticket in an Arraylist.

* Customer Service Analyst:

1. The ExecutionFlow method hits the FlowMonitor(Object) which ultimately does the following:
2. The First step of FlowMonitor object if we choose the Customer Service Analyst (option 2) prompts the user with 3 more options.

* Update Suspend Ticket

To update the suspended ticket by checking the top 3 frequent words and matching it with all the tickets and then assigning it the department id of the most similar ticket.

* Print Sentiment Analysis

To calculate and print the below stats.

* + Average Sentiment Score of all tickets.
  + Average Sentiment Score of each customer representative.
  + Average Sentiment Score of each service.
* Print Flow Status of All tickets

Prints all the required details of all the tickets.

**Java Code:**

**Ticket.Java**

**import** java.util.\*;

**public** **class** Ticket {

**public** Ticket() {

// **TODO** Auto-generated constructor stub

}

String ticketID; //basic ticket details

String CustomerRepID; //basic ticket details

String DateCreated; //basic ticket details

String CustomerserviceRepID; //basic ticket details

String ID; //customer details

String Name; //customer details

**int** phnNumber; //customer details

String Address; //customer details

String ServiceID; //service details

String ServiceStartDate; //service details

String ServiceEndDate; //service details

**boolean** Isresolved; //service details

**int** DepartmentID; //service details

Transcript Trans; //service details

FlowMonitor Fl; //service details

Scanner sc=**new** Scanner(System.***in***);

}

**class** PhoneCall **extends** Ticket{ //contact handler of phone call.

**public** Ticket contactHandler(){

String transcript="";

String line;

String input;

System.***out***.println("Customer Calls Company..."); //Assigning details.

**while**(**true**){

System.***out***.println("Press I for ID Number \nPress N for Name \nPress P for Phone Number \nPress A for Address \nPress SI for ServiceID \nPress SD for ServiceStartDate \nPress ED for ServiceEndDate \nPress T to add conversation into Transcript \nPress E to exit");

input=sc.nextLine();

**if**(input.equalsIgnoreCase("I")){

System.***out***.println("Entering and Updating the ID");

**this**.ID=sc.nextLine() ;

}

**else** **if**(input.equalsIgnoreCase("N")){

System.***out***.println("Entering and Updating Name");

**this**.Name=sc.nextLine();

}

**else** **if**(input.equalsIgnoreCase("P")){

System.***out***.println("Entering and Updating phone number");

**this**.phnNumber=sc.nextInt();

}

**else** **if**(input.equalsIgnoreCase("A")){

System.***out***.println("Entering and Updating address");

**this**.Address=sc.nextLine();

}

**else** **if**(input.equalsIgnoreCase("SI")){

System.***out***.println("Entering and Updating ServiceID");

**this**.ServiceID=sc.nextLine();

}

**else** **if**(input.equalsIgnoreCase("SD")){

System.***out***.println("Entering and Updating Service Start Date");

**this**.ServiceStartDate=sc.nextLine();

}

**else** **if**(input.equalsIgnoreCase("ED")){

System.***out***.println("Entering and Updating Service End Date");

**this**.ServiceEndDate=sc.nextLine();

}

**else** **if**(input.equalsIgnoreCase("E"))

{**break**;}

**else** **if**(input.equalsIgnoreCase("T")){ //Updating transcript.

System.***out***.println("Enter and Update Transcript \n");

**while**(**true**){

line=sc.nextLine();

transcript+=line;

**if**(line.equalsIgnoreCase("")){

**break**;}

}

}

}

Trans=**new** Transcript(transcript); //Providing decision to the ticket.

Trans.List();

System.***out***.println(Trans.wordList);

System.***out***.println("Please provide the Decision on this ticket\n Should ticket must be resolved?\n");

Isresolved=sc.nextBoolean(); //Checking if it is resolved or not,accordingly rerouted or suspended.

{

**if**(**this**.Isresolved==**true**){ // Representative enters whether the ticket is resolved or not , based on user input.

**this**.DepartmentID=0;

System.***out***.println("Ticket is Resolved");

}

**else**{ // Please enter specific department to route the ticket

System.***out***.println("Please provide the department ID to route this ticket or enter -1 if this needs to be suspended");

**this**.DepartmentID=sc.nextInt();{

**if**(DepartmentID== -1){

System.***out***.println("The ticket is suspended \n");}

**else**{

System.***out***.println("The ticket is rerouted to department: "+DepartmentID);}

}

}

}

**return** **this**;

}

}

**class** InPerson **extends** Ticket{ //contact handler of inperson.

**public** Ticket contactHandler(){

String input;

System.***out***.println(" Customers Enters Company...\n "); //Entering details.

**while**(**true**){

System.***out***.println("Press I for ID Number \nPress N for Name \nPress P for Phone Number \nPress A for Address \nPress SI for ServiceID \nPress SD for ServiceStartDate \nPress ED for ServiceEndDate \nPress E to exit");

input=sc.nextLine();

**if**(input.equalsIgnoreCase("I")){

System.***out***.println("Entering and Updating the ID");

**this**.ID=sc.nextLine() ;

}

**else** **if**(input.equalsIgnoreCase("N")){

System.***out***.println("Entering and Updating Name");

**this**.Name=sc.nextLine();

}

**else** **if**(input.equalsIgnoreCase("P")){

System.***out***.println("Entering and Updating phone number");

**this**.phnNumber=sc.nextInt();

}

**else** **if**(input.equalsIgnoreCase("A")){

System.***out***.println("Entering and Updating address");

**this**.Address=sc.nextLine();

}

**else** **if**(input.equalsIgnoreCase("SI")){

System.***out***.println("Entering and Updating ServiceID");

**this**.ServiceID=sc.nextLine();

}

**else** **if**(input.equalsIgnoreCase("SD")){

System.***out***.println("Entering and Updating Service Start Date");

**this**.ServiceStartDate=sc.nextLine();

}

**else** **if**(input.equalsIgnoreCase("ED")){

System.***out***.println("Entering and Updating Service End Date");

**this**.ServiceEndDate=sc.nextLine();

}

**else** **if**(input.equalsIgnoreCase("E"))

{**break**;}

}

System.***out***.println("Customer Leaves\n");

System.***out***.println("Please provide the Decision on this ticket\n Should ticket must be resolved?\n"); //Providing decision about resolving.

Isresolved=sc.nextBoolean();

{

**if**(**this**.Isresolved==**true**){ // Representative enters whether the ticket is resolved or not , based on user input.

**this**.DepartmentID=0;

System.***out***.println("Ticket is Resolved");

}

**else**{ // Please enter specific department to route the ticket

System.***out***.println("Please provide the department ID to route this ticket or enter -1 if this needs to be suspended");

**this**.DepartmentID=sc.nextInt();{

**if**(DepartmentID== -1){

System.***out***.println("The ticket is suspended \n");}

**else**{

System.***out***.println("The ticket is rerouted to department: "+DepartmentID);}

}

}

}

System.***out***.println("Representative Enters what ever he remembers \n"); //Adding comments after customer leaves.

sc.nextLine(); //

String transcript=sc.nextLine(); // Enter everything in one line using "." as Line Separator.

Trans=**new** Transcript(transcript);

Trans.List();

**return** **this**;

}

}

**class** Email **extends** Ticket{ //contact handler of email.

**public** Email(String iD, String name, **int** phnNumber, String address, String serviceID, String serviceStartDate,

String serviceEndDate) {

// **TODO** Auto-generated constructor stub

**this**.ID=iD; //Getting details from the customer.

**this**.Name=name;

**this**.phnNumber=phnNumber;

**this**.Address=address;

**this**.ServiceID=serviceID;

**this**.ServiceStartDate=serviceStartDate;

**this**.ServiceEndDate=serviceEndDate;

//String Email=email;

}

**public** Ticket contactHandler(String email){

System.***out***.println("Printing the Email from Customer\n"+email);

System.***out***.println("Please provide the Decision on this ticket whether resolved(TRUE or FALSE)"); //Proving decision about resolving the issue.

Isresolved=sc.nextBoolean();

**if**(**this**.Isresolved==**true**){ // Representative enters whether the ticket is resolved or not , based on user input.

**this**.DepartmentID=0;

}

**else**{ // Please enter specific department to route the ticket

System.***out***.println("Please provide the department ID to route this ticket or enter -1 if this needs to be suspended");

**this**.DepartmentID=sc.nextInt();

**if**(DepartmentID== -1){

System.***out***.println("The ticket is suspended");}

**else**{

System.***out***.println("The ticket is rerouted to department: "+DepartmentID);}

}

Trans=**new** Transcript(email);

Trans.List();

//this.sentimentScore=Fl.getSentiment(Trans.wordList);

/\*

this.Trans.sentenceList=new ArrayList<String>(Arrays.asList(email.split("\\."))); // Spillting the transcripts into sentences

String t1=email.replaceAll("\\W"," "); // For removing Punctuations

String t2=t1.replaceAll("\\s+"," ");//For removing Spaces

ArrayList<String> removeWords=new ArrayList<String>(Arrays.asList(Trans.stopWords)); //For converting stopWords to Arraylist so that we can remove easily

this.Trans.wordList=new ArrayList<String>(Arrays.asList(t2.split("\\s+"))); // For Splitting into words

this.Trans.wordList.removeAll(removeWords);

\*/

**return** **this**;

}

}

**Transcript.java**

**import** java.util.\*;

**public** **class** Transcript {

String transcript;

**public** Transcript(String transcript) {

**this**.transcript=transcript;

// **TODO** Auto-generated constructor stub

}

**public** Transcript() {

// **TODO** Auto-generated constructor stub

}

**public** **final** String[] stopWords={"A","about","above","after","again","against","all","am","an","and","any","are","aren't","as","at","be","because","been","before","being","below","between","both","but","by","can't","cannot","could","couldn't","did","didnt","do","does","doesn't","doing","don't","down","during","each","few","for","from","further","had","hadn't","has","hasn't","have","haven't","having","he","he'd","he'll","he's","her","here","here's","hers","herself","him","himself","his","how","how's","I","I'd","I'll","I'm","I've","if","in","into","is","isn't","it","it's","its","itself","let's","me","more","most","mustn't","my","myself","no","nor","not","of","off","on","once","only","or","other","ouht","our","ours","ourselves","out","over","own","same","shan't","she","she'd","she'll","she's","should","shouldn't","so","some","such","than","that","that's","the","their","theirs","them","themselves","then","there","here's","these","they","they'd","they'll","they're","they've","this","those","through","to","too","under","until","up","very","was","wasn't","we","we'd","we'll","we're","we've","were","weren't","what","what's","when","when's","whre","where's","which","while","who","who's","whom","why","why's","with","won't","would","wouldn't","you","you'd","you'll","you're","you've","your","yours","yourself","yourselves","exit","EXIT","EOM"};

**public** **final** String[] positiveLexicons={"don't","down","during","each","few","for","from","further","had","hadn't","has","hasn't","have","haven't","having","he","he'd","he'll","he's","her","here","here's","hers","herself","him","himself","his","how","how's","I","I'd","I'll","I'm","I've","if","in","into","is","isn't","it","it's","its","itself","let's","me","more"};

**public** **final** String[] negativeLexicons={"bad","worse","worst","crazy","confuse","confused","confusing","disturb","disturbance","disturbed","angry","annoy","annoyed","annoying","annoyingly","refuse","refsed","regret","reject","rejected","repetitive","repetitively","cancel","discontinue","terminate","mad","unhappy","unhelpful","unlucky"};

ArrayList<String> sentenceList= **new** ArrayList<String>();

ArrayList<String> wordList=**new** ArrayList<String>();

//ArrayList<ArrayList<String>> wordlist=new ArrayList<ArrayList<String>>();

Object[][] totalWorldCount=**new** Object[1][2];

**double** averageSentenceLength;

String[] topFreqWords= **new** String[3];

**public** String[] getTopFreqWords() { //Getting top 3 frequency words.

**return** topFreqWords;

}

**public** **void** setTopFreqWords(String[] topFreqWords) { //setting top 3 frequency words.

**this**.topFreqWords = topFreqWords;

}

**public** **void** List(){

System.***out***.println("Transcript Creation");

**this**.sentenceList=**new** ArrayList<String>(Arrays.*asList*(transcript.split("\\."))); // Spillting the transcripts into sentences

System.***out***.println(sentenceList+"\n");

String t1=transcript.replaceAll("\\W"," "); // For removing Punctuations

System.***out***.println(t1+"\n");

String t2=t1.replaceAll("\\s+"," ");//For removing Spaces

System.***out***.println(t2+"\n");

ArrayList<String> removeWords=**new** ArrayList<String>(Arrays.*asList*(**this**.stopWords));//For converting stopWords to Arraylist so that we can remove easily

//System.out.println("StopWords"+removeWords);

**this**.wordList=**new** ArrayList<String>(Arrays.*asList*(t2.split("\\s+"))); // For Splitting into words

**this**.wordList.removeAll(removeWords); //Removing words.

}

**public** String[] freqCount(){

ArrayList<String> words= **new** ArrayList<String>();

ArrayList<Double> freq= **new** ArrayList<Double>();

String[] words1=wordList.toArray(**new** String[words.size()]);

Map<String, Double> map = **new** HashMap<>(); //Using hashmaps to find the top 3 frequencies.

**for** (String w : words1) {

Double n = map.get(w);

n = (n == **null**) ? 1 : ++n; //Counter to count the frequency of each unique word.

map.put(w, n);

// String[] targetArray = values.toArray(new String[values.size()]);

}

System.***out***.println(Arrays.*asList*(map));

**for**(String key : map.keySet()){

**double** val = map.get(key);

words.add(key); //Creating arrayList of words.

freq.add(val); // Creating arraylist of frequencies

}

System.***out***.println(words); //Creating an array of top 3 frequency words of every ticket.

System.***out***.println(freq);

**int** a = -1,b=-1,c=-1;

**for**(**int** i=0; i<freq.size();i++){

**if**(a==-1 || freq.get(i)>=freq.get(a)){

c=b;

b=a;

a=i;

}**else** **if**(b==-1 ||freq.get(i)>=freq.get(b)){

c=b;

b=i;

}

**else** **if**(c==-1 ||freq.get(i)>=freq.get(c)){

c=i;

}

}

System.***out***.println("top 3 words are: " + words.get(a)+"," + words.get(b)+","+words.get(c));

String[] words2={ words.get(a) ,words.get(b) ,words.get(c)};

System.***out***.println("\n" +words2[0]+","+words2[1]+","+words2[2]+"\n");

String[] topFreqWords={words2[0],words2[1],words2[2]};

**return** topFreqWords;

// These to TopFreqWords using setTopFreqWords

//setTopFreqWords(String[] words2);

}

**public** **double** getaverageSentenceLength(){ // Getting average sentence length.

**return** averageSentenceLength;

}

**public** **void** setaverageSentenceLength(){ // setting average sentence length.

**double** averageSentenceLength;

averageSentenceLength= wordList.size() / sentenceList.size(); //average sentence length.

**this**.averageSentenceLength = averageSentenceLength;

}

**public** **int** getSimilarity(Transcript trans ){ //Checking similarity between current ticket and other tickets.

/\*Set<String> firstSet = new HashSet<String>(Arrays.asList(topFreqWords2));

for(int i=0;i<ttFW.size();i++){

String[] ttFWS=(String[]) ttFW.get(i).toArray();

Set<String> secondSet = new HashSet<String>(Arrays.asList(ttFWS));

}\*/

**int** count=0;

**for**(**int** i=0;i<3;i++){

**for**(**int** j=0;j<3;j++){

**if**(topFreqWords[i]==trans.topFreqWords[j]){

count++;

//continue;

}

}

}

**return** count;

}

}

FlowMonitor.java

import java.text.\*;

import java.util.\*;

public class FlowMonitor {

public FlowMonitor() {

// TODO Auto-generated constructor stub

}

public static String[] arrayRepresentativeIDs={"1","2","3"}; //array for representative IDs

public static int[] arrayServiceIDs={1,2,3,4,5,6,7,8,9}; //array for representative service IDs

static ArrayList<Ticket> T= new ArrayList<Ticket>();

static Scanner Sc=new Scanner(System.in);

static int ticketID=1000; //initialising the ticket id

public static void assignTicket(Ticket ticket,String toT){

Date dateCreated=new Date(); //getting the current date

DateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss");

//int ticketID=1000;

while(true){

ticket.ticketID=Integer.toString(ticketID); //converting the ticket id to string and assigning it to ticket id.

ticketID++;

break;

}

if(toT.equalsIgnoreCase("Phone")){ //if type of ticket is phone

ticket.CustomerRepID=arrayRepresentativeIDs[0]; //assigning the ticket to representative 0

}

else if(toT.equalsIgnoreCase("InPerson")){ //if type of ticket is inperson

ticket.CustomerRepID=arrayRepresentativeIDs[1]; //assigning the ticket to representative 1

}

else if(toT.equalsIgnoreCase("Email")){ //if type of ticket is email

ticket.CustomerRepID=arrayRepresentativeIDs[2]; //assigning the ticket to representative 2

}

ticket.CustomerserviceRepID=Integer.toString(new Random().nextInt(arrayServiceIDs.length)); //randomly assigning the service representative id

ticket.DateCreated=dateFormat.format(dateCreated);

}

public static void createTicket(String toT){ //creation of createTicket method

Ticket total=new Ticket();

if (toT.equalsIgnoreCase("Phone")){ //type of ticket is phone

PhoneCall pc = new PhoneCall(); //Creating a phone object

System.out.println("\n Assign Ticket Details\n");

assignTicket(pc,toT); //assigning the basic ticket details

total=pc.contactHandler(); //calling the contact handler method

T.add(total); //adding the details in transcript

//System.out.println("Testing");

return;

}

else if (toT.equalsIgnoreCase("inperson")){ //type of ticket is inperson

InPerson ip=new InPerson(); //Creating a inperson object

assignTicket(ip,toT); //assigning the basic ticket details

total=ip.contactHandler(); //calling the contact handler method

T.add(total); //adding the details in transcript

return;

}

else if (toT.equalsIgnoreCase("email"));{ //type of ticket is email

String email="";

String line="";

System.out.println("Enter the email Provided\n");

System.out.println("Please enter your ID, name,phoneNumber,Address,serviceID,ServiceStartDate,ServiceEndDate , rest of email\n");

while(true){ // For taking input.Each sentence should end with "." and Representative goes to next line by pressing enter. "Exit" is used to get out of loop

/\*

\* Please enter the Email in ID,Name,PhnNumber,Address,ServiceID,ServiceStartDate,ServiceEndDate and rest of mail with "." after each line

\*/

line=Sc.nextLine(); // Infinite loop till "exit" is pressed

email+=line;

if(line.equalsIgnoreCase(""))

break;

}

ArrayList<String>sentenceList=new ArrayList<String>(Arrays.asList(email.split("\\."))); //splitting the email based on "."

String ID=sentenceList.get(0); //get first element from Sentencelist

sentenceList.remove(0); //remove the first element

String Name=sentenceList.get(0);

sentenceList.remove(0);

int PhnNumber=Integer.parseInt(sentenceList.get(0));

sentenceList.remove(0);

String Address=sentenceList.get(0);

sentenceList.remove(0);

String ServiceID=sentenceList.get(0);

sentenceList.remove(0);

String ServiceStartDate=sentenceList.get(0);

sentenceList.remove(0);

String ServiceEndDate=sentenceList.get(0);

sentenceList.remove(0);

String Email=String.join(". ",sentenceList);

Email em=new Email(ID,Name,PhnNumber,Address,ServiceID,ServiceStartDate,ServiceEndDate); //creating the email object

assignTicket(em,toT); //assigning the ticket

total=em.contactHandler(Email); //calling the email contacthandler

T.add(total); //adding the ticket to arraylist

}

return ;

//updateTicket();

}

public static void updateTicket( ){ //updateTicket method

/\*ArrayList<ArrayList<String>> totaltopFreqWords=new ArrayList<ArrayList<String>>();

for(int i=0;i<T.size();i++)

{

ArrayList<String> FreqWords=new ArrayList<String>();

for(int j=0;j<3;j++){

FreqWords.add(T.get(i).Trans.topFreqWords[j]);}

totaltopFreqWords.add(FreqWords);

FreqWords.clear();

}\*/

Transcript trans=new Transcript(); //creating an object of type transcript

//int[] simNum = null;

// ArrayList<ArrayList<Integer>> simNum=new ArrayList<ArrayList<Integer>>();

ArrayList<Integer> simNum=new ArrayList<Integer>(); //creation of simnum array list

for(int j=0;j<T.size();j++){ //loop for fetching the suspended ticket

if(T.get(j).DepartmentID==-1){ //check condition.

for(int i=0;i<T.size();i++){

if(T.get(i).DepartmentID!=-1){

Transcript Trans=T.get(i).Trans; //fetching data from transcript.

// ArrayList<Integer>simnum=new ArrayList<Integer>(Arrays.asList(T.get(j).Trans.getSimilarity(Trans)));

int num =Trans.getSimilarity(Trans); //checking for similarity by calling the getsimilarity method.

simNum.add(num); //Adding the result to simnum arraylist to check the similarity.

}else

simNum.add(0);

}

int max = 0;

for(int p =0 ; p< simNum.size();p++){ //loop to check the highest number of similarity.

if(simNum.get(p)>simNum.get(max))

max = p;

}

if(simNum.get(max)>0) //assigning the department ID of the most similar ticket.

T.get(j).DepartmentID = T.get(max).DepartmentID;

else

System.out.println("No similar ticket found");

//int n=trans.getSimilarity(T.get(j).Trans.topFreqWords,totaltopFreqWords);

//T.get(j).DepartmentID=n ;

}

}

}

public static int getSentiment(Ticket T){ //Calling the ticket and checking the wordlist with positive and negative lexicons.

int sentimentScore=0;

Transcript Trans=new Transcript();

int positive=0;

int negative=0;

//System.out.println("Entering Sentiment Score Calculator");

//System.out.println("WordList Size"+T.Trans.wordList.size()+"\n");

for(int i =0;i <T.Trans.wordList.size();i++){

for(int j =0;j< Trans.positiveLexicons.length;j++){

if(T.Trans.wordList.get(i)==Trans.positiveLexicons[j]){

positive++;

System.out.println(T.Trans.wordList.get(i));

}

}

for(int k=0;k < Trans.negativeLexicons.length;k++){

if(T.Trans.wordList.get(i)==Trans.negativeLexicons[k]){

negative++;

System.out.println(T.Trans.wordList.get(i));

}

}

}

if(positive > negative){ //Assigning sentimentScore according to the number of positive and negative lexicons.

sentimentScore=3;

}

else if(positive < negative){

sentimentScore=1;

}

else{

sentimentScore=2;

}

return sentimentScore;

}

public static ArrayList<Object> sentimentAnalysis(){

ArrayList<Object> analysis=new ArrayList<Object>();

double sum=0;

double[] Rsum=new double[arrayRepresentativeIDs.length];

int[] Rcount=new int[arrayRepresentativeIDs.length];

double[] Ssum= new double[arrayServiceIDs.length];

int[] Scount= new int[arrayServiceIDs.length];

for(int i=0;i<T.size();i++){ //Calculating the total sentimentScore

int sentiscore=getSentiment(T.get(i));

sum+=sentiscore;

}

analysis.add(sum/T.size()); //Finding the average sentiment score of all the tickets.

for(int j=0;j<T.size();j++){ //Getting the sentiment Score from the customer representative.

for(int m=0;m<arrayRepresentativeIDs.length-1;m++){

if(T.get(j).CustomerRepID.equalsIgnoreCase(arrayRepresentativeIDs[m])){

Rsum[m]+=getSentiment(T.get(j));

Rcount[m]++;

}

}

}

double[] Ranalysis=new double[arrayRepresentativeIDs.length]; //Average of sentimentScore based on customer Representative.

for(int a=0;a<arrayRepresentativeIDs.length-1;a++){

Ranalysis[a]=Rsum[a]/Rcount[a];

analysis.add(Ranalysis[a]);

}

for(int k=0;k<T.size();k++){ //Total sentimentScore based on customer service.

for(int l=0;l<arrayServiceIDs.length-1;l++){

if(T.get(k).CustomerserviceRepID.equalsIgnoreCase(Integer.toString(arrayServiceIDs[l]))){

Ssum[l]+=getSentiment(T.get(k));

Scount[l]++;

}

}

}

double[] Sanalysis=new double[arrayServiceIDs.length]; //Average sentimentScore based on customer service.

for(int a=0;a<arrayServiceIDs.length-1;a++){

Sanalysis[a]=Ssum[a]/Scount[a];

analysis.add(Sanalysis[a]);

}

return analysis ;

}

public static void printAnalysis( ){

System.out.println("Enter the type of Analysis \n Press 0 for the Overall Sentiment Scores of the Tickets \n Press 1 for the Average Sentiment Scores of the ticket for Each Customer Representative \n Press 2 for Average Sentiment Scores for the Services \n");

//int n=toA;

int n=Sc.nextInt();

int flag=1;

int count1=arrayRepresentativeIDs.length;

int count2=count1+arrayServiceIDs.length;

ArrayList<Object> Analysis=sentimentAnalysis();

if(n==0){ //Printing analysis according to user input.

System.out.println("Overall Sentiment Scores of the Tickets : "+Analysis.get(0)+ "\n");

}

else if(n==1){

for(int i=1;i<arrayRepresentativeIDs.length+1;i++){

System.out.println("Average Sentiment Scores of the tickets of Customer Representative-"+i+ ":" +Analysis.get(i)+ "\n");

}

}

else if(n==2){

for(int i=count1+1;i<count2+1;i++)

System.out.println("Average Sentiment Scores for the Service-"+ flag+ " : "+Analysis.get(i)+ "\n");

flag++;

}

System.out.println("\n \n ");

}

public static void printFlowStatus(){

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");

System.out.println("Printing All the tickets \n \n \n");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");

for(int i=0;i<T.size();i++){

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");

System.out.println("Ticket Details : \n");

System.out.println("Ticket ID : " + T.get(i).ticketID);

System.out.println("Date Created : " + T.get(i).DateCreated);

System.out.println("Customer's Representative ID : " + T.get(i).CustomerRepID);

System.out.println("Customer's Service ID : " + T.get(i).CustomerserviceRepID+"\n");

System.out.println("Customer's Details \n");

System.out.println("Customer's ID " + T.get(i).ID);

System.out.println("Customer's Name " + T.get(i).Name);

System.out.println("Customer's Phone Number" + T.get(i).phnNumber);

System.out.println("Customer's Address" + T.get(i).Address+"\n");

System.out.println("Customer's Service Details \n");

System.out.println("Service ID" + T.get(i).ServiceID);

System.out.println("Start Date of Service" + T.get(i).ServiceStartDate);

System.out.println("End Date of Service" + T.get(i).ServiceEndDate);

System.out.println("Department ID"+ T.get(i).DepartmentID);

System.out.println("Status of Ticket"+T.get(i).Isresolved);

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");

System.out.println("\n \n ");

}

}

}

**ExecutionFlow.java**

/\*

\* GROUP 5

\*

\* Venkata Rama Harsha Dasari - vdasar3@uic.edu

\* Antara Gupta - agupt30@uic.edu

\* Shreayash Shukla - sshukl6@uic.edu

\*

\*/

**import** java.util.\*;

**public** **class** ExecutionFlow {

**public** **static** **void** main(String[] args){

FlowMonitor Fl=**new** FlowMonitor();

Scanner sc=**new** Scanner(System.***in***); //Creating scanner object to take user input

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");

System.***out***.println(" Welcome to Customer Support System \n");

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");

**while**(**true**){

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");

System.***out***.println(" Please Select the user \n 1.Customer Service Representative \n 2.Customer Services Analyst \n 3.Exit \n"); //Menu for choosing the representative.

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");

**int** typeofUser=sc.nextInt();

**if**(typeofUser==1){ //If the user is customer representative.

System.***out***.println("Please enter the type of ticket");

String toT=sc.next(); //Take the type of ticket

Fl.*createTicket*(toT); //Creation of a ticket

**continue**;

}

**else** **if**(typeofUser==2){ //if customer service analyst

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");

System.***out***.println("Please Select the Options: \n\n 1.Update Suspened Ticket \n 2.Print Sentiment Analysis \n 3.Print Flow Status of All tickets ");

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n"); //Select the above option for performing the analysis.

**int** options=sc.nextInt();

**if**(options==1){

Fl.*updateTicket*(); //selecting option 1

}

**else** **if**(options==2){ //selecting option 2

Fl.*printAnalysis*();

}

**else** **if**(options==3){ //selecting option 1

Fl.*printFlowStatus*();

}

**continue**;

}

**else** **if**(typeofUser==3){ //for exiting the application

**break**;

}

}

}

}