SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE

A PRELIMINARY PROJECT REPORT ON

ELATE- AN INTELLIGENT THERAPIST CHATTERBOT

SUBMITTED TOWARDS THE

PARTIAL FULFILLMENT OF THE REQUIREMENTS OF

BACHELOR OF ENGINEERING

(COMPUTER ENGINEERING)

BY

Akanksha Patil	B120394334
Kshitija Pandit	B120394332
Aishwarya Nerlekar	B120394326
Anuja Tatpuje	B120394398

UNDER GUIDANCE OF
Prof. L. A. Bewoor

DEPARTMENT OF COMPUTER ENGINEERING

VISHWAKARMA INSTITUTE OF INFORMATION TECHNOLOGY, PUNE

VISHWAKARMA INSTITUTE OF INFORMATION TECHNOLOGY, PUNE DEPARTMENT OF COMPUTER ENGINEERING

CERTIFICATE

This is to certify that the Project entitled

ELATE- AN INTELLIGENT THERAPIST CHATTERBOT

Submitted by

Akanksha Patil	B120394334
Kshitija Pandit	B120394332
Aishwarya Nerlekar	B120394326
Anuja Tatpuje	B120394398

is a bonafide work carried out by Students under the supervision of Prof. L.A.Bewoor and it is submitted towards the partial fulfilment of the requirement of Bachelor of Engineering (Computer Engineering) Project.

Prof. L. A. Bewoor	Dr S.R.SAKHARE
Internal Guide	H.O.D.
Dept. of Computer Engg.	Dept. of Computer Engg.

PROJECT APPROVAL SHEET

A Project Title

ELATE- AN INTELLIGENT THERAPIST CHATTERBOT

Is successfully completed by

Akanksha Patil	B120394334
Kshitija Pandit	B120394332
Aishwarya Nerlekar	B120394326
Anuja Tatpuje	B120394398

at

DEPARTMENT OF COMPUTER ENGINEERING

VISHWAKARMA INSTITUTE OF INFORMATION TECHNOLOGY, PUNE DEPARTMENT OF COMPUTER ENGINEERING

SAVITRIBAI PHULE PUNE UNIVERSITY,PUNE

ACADEMIC YEAR 2017-2018

Prof. L. A. Bewoor	Dr S.R.SAKHARE
Internal Guide	H.O.D.
Dept. of Computer Engg.	Dept. of Computer Engg.

Abstract

"ELATE" is a self-initiating chatter bot which focuses on elating or rather motivating the user. Elate means making someone happy, thus our chatter bot chat system is designed in such a way which might help end user to feel good about itself. In the era of fierce competition, everyone feels the need of encouragement; ELATE might prove fruitful in encouraging oneself by constantly being available. There are many chatter bots in market which provide motivating happy-chats, but also come with limitations and drawbacks. Our main focus is to work on these limitations of already established chatter bot products, so as to create a human-like intelligent chat system. Also, unlike other chatter bots, ELATE will consists of sentiment analysis unit, on the basis of this unit, chat will be initialized. Sentiment analysis will be done on the basis of set of questionnaire to help our chatter-bot analyze user's mood. Further user's sentiments will be categorized in three groups viz. positive, neutral and negative. According to derived sentiments chats will be fired at user.

Acknowledgement

Example given as It gives us great pleasure in presenting the preliminary project report on
‘ELATE- AN INTELLIGENT THERAPIST CHATTER BOT’.

I would like to take this opportunity to thank my internal guide **Prof. L.A. BEWOOR** for giving me all the help and guidance I needed. I am really grateful to them for their kind support. Their valuable suggestions were very helpful.

I am also grateful to **Prof. S.R. Sakhre**, Head of Computer Engineering Department, Vishwakarma Institute of Information Technology for his indispensable support, suggestions.

In the end our special thanks to **Mrs. N.S. Bhirud** and **Mr. P.K. Kharade** for providing various resources such as laboratory with all needed software platforms, continuous Internet connection, for Our Project.

Akanksha Patil
Kshitija Pandit
Aishwarya Nerlekar
Anuja Tatpuje
(B.E. Computer Engg.)

Contents

List of Figures

List of Tables

CHAPTER 1
SYNOPSIS

1.1 PROJECT TITLE

ElATE : An Intelligent Therapist Chatter bot

1.2 PROJECT OPTION

Internal Project

1.3 INTERNAL GUIDE

Prof. L. A. Bewoor

1.4 SPONSORSHIP AND EXTERNAL GUIDE

None

1.5 TECHNICAL KEYWORDS (AS PER ACM KEYWORDS)

- Computing Methodologies
- ARTIFICIAL INTELLIGENCE
- Applications
- Sentiment Analysis
- Natural Language Processing
- Natural Language generation
- Miscellaneous

1.6 PROBLEM STATEMENT

Create an intelligent therapist chatter bot mobile application which will converse with user by first analysing the sentiment or mood of the user and thereby initializing a positive, negative or neutral chat format to motivate and elate the user'

1.7 ABSTRACT

"ELATE" is a self-initiating chatter bot which focuses on elating or rather motivating the user. Elate means making someone happy, thus the chatter bot chat system is designed in such a way which

might help end user to feel good about it. In the era of fierce competition, everyone feels the need of encouragement; ELATE might prove fruitful in encouraging one by constantly being available. There are many chatter bots in market which provide motivating happy-chats, but also come with limitations and drawbacks. Our main focus is to work on these limitations of already established chatter bot products, so as to create a human-like intelligent chat system. Also, unlike other chatter bots, ELATE will consists of sentiment analysis unit, on the basis of this unit, chat will be initialized. Sentiment analysis will be done on the basis of twitter account's tweet of user(if user has twitter account) or user will be asked a set of questionnaire to help our chatter bot analyse user's mood. Further user's sentiments will be categorized in three groups viz. positive, neutral and negative. According to derived sentiments chats will be fired at user.

1.8 GOALS AND OBJECTIVES

- To analyse user's mood and categorization into 3 types namely positive, negative and neutral
- To create a chatter bot which has an intelligent chat system and can converse with user to thereby motivate
- To reduce the redundancy in chat content

1.9 RELEVANT MATHEMATICS ASSOCIATED WITH THE PROJECT

System Description:

- S = {Input, Output, Constraint, Functions, Success, }
- Input = {User's textual reply in English language }
- Output = {ELATE's textual response on the basis of user's mood.}
- Constraint = {Chat system is not generalised and is designed for females from the age group 18 to 24}
- Function = {Tokenization, POS Tagging, Sentence Generation, Sentiment analyser}
- Success = {Appropriate response to user's reply which will make her happy}
- Failure = {Inappropriate response to user's reply which will instead annoy her }

1.10 NAMES OF CONFERENCES / JOURNALS WHERE PAPERS CAN BE PUBLISHED

- Vishwacon Paper presentation
- Vishwacon Project Competition

1.11 REVIEW OF CONFERENCE/JOURNAL PAPERS SUP-

PORTING PROJECT IDEA

[1] Prof.Nikita Hatwar, Ashwini Patil , Diksha Gondane," AI BASED CHATBOT", International Journal of Emerging Trends in Engineering and Basic Sciences (IJEEBS), Volume 3, Issue 2 (March-April 2016), PP.85-87

[2] JosEPh ~VEIZENBA UM," ELIZA A Computer Program For the Study of Natural Language Communication Between Man And Machine", Massachusets Institute of Technology,Cambridge, Mass., volume 9 / Number / / January., 1966

[3] John Zakos,Liesl Capper," CLIVE – An Artificially Intelligent Chat Robot for Conversational

[4] Language Practice", Griffith University, Gold Coast, Australia, SETN 2008, LNAI 5138, pp. 437–442, 2008

[5] AlbertGattand,EhudReiter," SimpleNLG : A realization engine for practical applications", Department of Computing Science University of Aberdeen Aberdeen AB24 3UE, UK, Proceedings of the 12th European Workshop on Natural Language Generation, pages 90–93, Athens, Greece, 30 – 31 March 2009.

[6] Bayan AbuShawar, Eric Atwell," Automatic Extraction of Chatbot Training Data from Natural Dialogue Corpora", IT department; School of Computing Arab Open University; University of Leeds Amman, Jordan; leeds, Uk

[7] Anirudh Khanna, Bishwajeet Pandey, Kushagra Vashishta, Kartik Kalia, Bhale Pradeepkumar and Teerath Das," A Study of Today's A.I. through Chatbots and Rediscovery of Machine Intelligence", Chitkara University, Punjab, India, International Journal of u- and e- Service, Science and Technology Vol.8, No. 7 (2015), pp.277-284

[8] Nicole Radziwill and Morgan Benton," Evaluating Quality of Chatbots and Intelligent Conversational Agents" ,2016

[9] M. A. Mioc." Client Server Chat Application", University Stefan cel Mare Suceava Integrated Center for research, development and innovation in Advanced Materials, Nanotechnologies, and Distributed Systems – MANSiD Suceava, Romania, Journal of Multidisciplinary Engineering Science and Technology (JMEST) ISSN: 2458-9403 Vol. 3 Issue 7, July - 2016

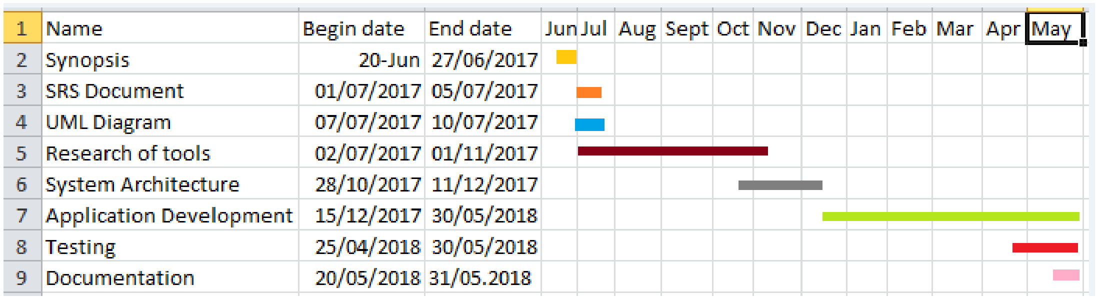
[10] Revati R. Dudhatra, Dr. Yogesh A Jogsan," Mental Health and Depression among Working and NonWorking Women", Department of Psychology, Saurashtra University, Rajkot, India, International Journal of Scientific and Research Publications, Volume 2, Issue 8, August 2012

[11] Holger Stenzhorn,” XtraGen — A Natural Language Generation System Using XML- and Java-Technologies” , XtraMind Technologies GmbH Stuhlsatzenhausweg 3 66123 Saarbr¨ucken, Germany, October 2002

[12] Sameena Thabassum,” Sentiment Analysis on Interactive Conversational Agent/Chatbots”, User. International Journal of Computer Applications 120(1):21-24, June 2015.

[13] A. TURING, "I.—COMPUTING MACHINERY AND INTELLIGENCE", *Mind*, vol., no. 236, pp. 433-460, 1950.

1.12 PLAN OF PROJECT EXECUTION



1	Name	Begin date	End date
2	Synopsis	20-Jun	27/06/2017
3	SRS Document	01/07/2017	05/07/2017
4	UML Diagram	07/07/2017	10/07/2017
5	Research of tools	02/07/2017	01/11/2017
6	System Architecture	28/10/2017	11/12/2017
7	Application Development	15/12/2017	30/05/2018
8	Testing	25/04/2018	30/05/2018
9	Documentation	20/05/2018	31/05.2018

Figure 1.1: Plan of the Project Execution

Chapter 2

TECHNICAL KEYWORD

2.1 AREA OF PROJECT

Artificial Intelligence, Natural Language Processing, Natural Language Generation, Sentiment Analysis

2.2 TECHNICAL KEYWORDS

- Computing Methodologies
- ARTIFICIAL INTELLIGENCE
- Applications and Expert Systems
- Natural language interfaces
- Knowledge Representation Formalisms and Methods
- Natural Language Processing
- Language parsing and understanding

Chapter 3

INTRODUCTION

3.1 PROJECT IDEA

Core idea for this project is to develop an intelligent chat system which will chat with user when user is not happy. Users will comprise of females from the age group 18 to 24. Users will have to answer set of questionnaire and result of the same will form the basis of user's mood categorization. Mood of user will be categorized into 3 groups viz. positive, negative and neutral. On the basis of the category positive, negative and neutral chat format is intialised. Further chat is intelligently conveyed on the basis of User's response.

3.2 MOTIVATION OF THE PROJECT

There are many chatter bots in market which provide motivating happy chats, but also come with limitations and drawbacks. Our main focus is to work on the prior limitation of already established chatter bot products so as to create a human-like intelligent chat system. Also, unlike other chatter bots, ELATE will consists of questionnaire and sentiment analysis unit on the basis of this unit chat will be initialized.

3.3 LITERATURE SURVEY

A chat bot (also known as a talk bot, chatter bot, Bot, IM bot, interactive agent, or Artificial Conversational Entity) is a computer program which conducts a conversation via auditory or textual methods. Such programs are often designed to convincingly simulate how a human would behave as a conversational partner, thereby passing the Turing test. Chat bots are typically used in dialog systems for various practical purposes including customer service or information acquisition. Some chatter bots use sophisticated natural language processing systems, but many simpler systems scan for keywords within the input, then pull a reply with the most matching keywords, or the most similar wording pattern, from a database.

The term "Chatter Bot" was originally coined by Michael Mauldin (creator of the first Verbot, Julia) in 1994 to describe these conversational programs. Today, chat bots are part of virtual assistants such as Google Assistant, and are accessed via many organizations' apps, websites, and on instant messaging platforms. Non-assistant applications include chat bots used for entertainment purposes, for research, and social bots which promote a particular product, candidate, or issue.

In the Literature survey; following articles have been researched extensively.

1. Prof.Nikita Hatwar, Ashwini Patil , Diksha Gondane," AI BASED CHATBOT", International Journal of Emerging Trends in Engineering and Basic Sciences (IJEEBS), Volume 3, Issue 2 (March-April 2016), PP.85-87

In this paper, the authors main goal was to show how their creation were to resemble a human being in the way they perform said interaction, trying to make the user think he/she is writing to another human being. This has been implemented with varying degrees of success. They have shown how to use parsing the training corpus and constructing the conditional frequency distribution which will aid the development of human-like chat.

Our motivation behind project that is to not take human-human conversation as the gold standard for conversational exchanges but if one had a perfect simulation of a human conversant, then it would be human-human conversation and not human-computer conversation with its sometimes odd but pertinent properties.

2. Joseph Weizenbaum , "Computational Linguistic",Massachusetts Institute of Technology, Cambridge, Mass,Volume 1/Number 1/January,1966

In this paper, detailed working and implementation of ELIZA chatter bot has been discussed. ELIZA being the first chatter bot to have implemented NLP and NLG and thus this paper proved very helpful to us to understand the basic concepts of conversation chatter bot which utilizes NLP and NLG. Input text from the user is analysed on the basis of decomposition rules which are triggered by keywords in the input text. Response generation use rule based approach. Identification of keywords and minimal context was the main basis of our project and this paper helped us to plan this project.

ELIZA uses pattern matching technique and substitution methodology to replicate human-like conversation that is it operates by identifying key words or phrases from the user's input and the formulate a response/reply on the basis of identified keywords from pre-programmed responses. For example, if a human says that 'I am feeling sad'. ELIZA would pick up the words 'feeling' and 'sad', and respond by asking an open- ended question 'How long have you been feeling sad'. This way ELIZA simulated the human-like conversation by understanding the intent of human replies and accordingly responding to them similarly the way human would have replied .This proved to be a noteworthy impact on natural language processing and artificial intelligence.

3. John Zakos,Liesl Capper,” CLIVE – An Artificially Intelligent Chat Robot for Conversational

In this paper, an artificially intelligent chat robot called CLIVE has been presented which aims at providing useful and engaging method for people learning a foreign language, to practice their conversational skills. Unlike other systems that focus on providing a limited or structured tutoring experience for language learning, CLIVE has the ability of holding open, natural human-like conversations with people on a wide range of topics. This provides users with a life-like experience that is a more natural way of learning a new language.

Experiments were conducted between CLIVE and real human users and an analysis of the conversations shows that CLIVE performs with accuracy and is an accepted method of language practice amongst users. This paper helped us understand as to how to design the components of intelligent chat system and what makes a chatter bot intelligent.

4. AlbertGattand, EhudReiter, “SimpleNLG : A realization engine for practical applications”, Department of Computing Science University of Aberdeen Aberdeen AB24 3UE, UK, Proceedings of the 12th European Workshop on Natural Language Generation, pages 90–93, Athens, Greece, 30 – 31 March 2009.

This paper describes SimpleNLG, a realisation engine for English which aims to provide simple and robust interfaces to generate syntactic structures and linearise them. The library is also flexible in allowing the use of mixed (canned and non-canned) representations. We were able to get an overview of SimpleNLG library, Lexical operations, Syntactic operations, Interaction of lexicon and syntax.

This paper has described SimpleNLG as a realisation engine which differs from most tactical generators in that it provides a transparent API to carry out low-level tasks such as inflection and syntactic combination, while making no commitments about input specifications or input-output mappings. This library was the main component of our NLG unit.

5. Revati R. Dudhatra, Dr. Yogesh A Jogsan,” Mental Health and Depression among Working and NonWorking Women”, Department of Psychology, Saurashtra University, Rajkot, India, International Journal of Scientific and Research Publications, Volume 2, Issue 8, August 2012

The main purpose of this research paper was to find out the mean difference between working and non-working women in mental health and depression. Results of the paper revealed that non-working women showed more signs of depression than non-working women. This paper helped us to decide an age group of women to focus on and thereby design a chat system style which will more relatable to decided age group.

Literature survey of the various method, techniques have been surveyed: -

1. NLG tools and library survey
2. Client Server architecture in chat bot application (Reference [9])
3. Market survey of various therapist chatter bots. (Table no.)
4. POS Tagging and Sentence Generation

3.3.1 Need of THERAPIST CHATBOTS

Mental health care is among the most concerned topics in today's era.[5] And it is often ignored because of our busy schedule and hectic lifestyle. So to help combat this, therapist chatter bots can prove very helpful. The three main reasons why chatter bots can prove fruitful in improving mental health are availability, anonymity and cost-efficiency .One of the most important characteristics of therapist chatter bots is time and location independent counseling as they are deployed as mobile applications which can be easily accessed via smart phones anytime and anywhere, aided with meager or sometimes free of cost aided with negligible waiting times. Chatter bots can be favored in terms of anonymity that avoids denunciation. A person might rather share private mental health issues with a machine than with their family doctor/psychologists/therapist, because they refrain from and are afraid of being judged or misinterpreted by another human being.

3.3.2 Current market survey

Below table shows the pros and cons of already established therapist chatter bots applications.

Sr.no	Product	Pros	Cons
1.	Wysa	1)Uses pictures and video during chat conversation. 2)Handle redundant chat well but not in efficient way.	1)Asks user multiple choice questions to analyse user's mood. Thus, chat doesn't feel realistic or less human-like. 2)Inefficient use of multimedia, this might annoy the end user.
2.	Woebot	1)Intelligent chat system.	1)Paid application and thus not available for everyone to use.

		2)Efficient use of multimedia.	
3.	Joybot	1)Chat system is good and entertaining. 2)Rich multimedia data usage.	1)Inefficient multimedia usage, which can be irritating sometime. 2)Redundant chat. 3)Paid application after some tenure .
4.	Eliza	1)Chat system is intelligent. 2)Prompt reply. 3) Free and open source software.	1)Redundant chats are not handled efficiently. 2) Chats are mundane and might bore the user.
5.	Siri	1)Assistant Chatter bot with good and entertaining chat system.	1)Being an assistant chatter bot, chat system of siri is not intelligent. 2)Available only for ios smart phone users.
6.	Hike's Natasha	1)Assistant Chatter bot with good and entertaining chat system.	1)Being an assistant chatter bot, chat system of Natasha is not intelligent and delivers annoying chats at times. 2)Available only for Android smart phone users. 3) Chat system is highly redundant.

3.3.3 User’s Input Pre-Processing

The first phase of language processing is essentially pre-processing of the input text. The input text is a simple English textual script. This text was analysed to extract the sentiment of user. This involved tokenization of text and thereby extracting keywords.
“Tokenization” can be defined as the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens. A token is an instance of a sequence of characters, which are grouped together as a useful semantic unit for processing in some particular document. Loosely referred to as words; tokens are used for further processing
These keywords are then matched against the set of words categorized into 3 groups viz. positive, negative and neutral. List of words is stored in text file format identified by the respective keyword and basic matching algorithm.

3.3.4 POS-Tagging

“POS Tagging” or “Part Of Speech Tagging” is the process of assigning tags to a word in given text as corresponding to a particular part of speech, based on both its definition and its context. A POS tagger is software that reads the text in some language and assigns appropriate tags to each word or other tokens. It is also called grammatical tagging or word-category disambiguation. POS Taggers use a pre-tagged word corpus to match words and then assign an appropriate tag. The pre-tagged word corpus is also called “POS Dictionary” or “WordNet”.

POS Tagging can be both supervised and unsupervised. In both cases, there are three broad and basic approaches. They are Rule-based approach, Statistical approach and Hybrid approach. The rule-based approach uses hand-crafted rules and contextual information to assign POS tags to words. The major drawback of the rule-based system is that it fails when the text is unknown or when a word is not present in the corpus. The statistical or stochastic approach uses word frequency, probability and statistics to assign correct tags. The most popular techniques are n-gram based or Hidden Markov Model. These statistical taggers are more efficient than simple rule-based taggers but this approach may result in tags which are not acceptable or appropriate according to the grammar rules of a language. The hybrid approach has higher accuracy than other two approaches as it uses both approaches. First, it assigns tags using a statistical approach and then if the tag is wrong then it tries to correct it using the rule-based approach.

Various POS tags applicable to a natural language are known as tag-sets. ELATE uses the statistical or stochastic approach for POS tagging. Thus the proposed system only uses five tags to categorize the words. These important categories are noun, pronoun, adjective, adverb and verb.

For example, if user's reply is "I am sad", following were the POS tagging results,

```
        TOKEN       TAG                 PROBABILITY

          I :        PRP     :        0.9766097166460526
        am       :        VBP    :        0.9401058199142288
        sad      :        JJ     :        0.8709539008315662
```

3.3.5 Response generation

Response generation by ELATE consists of two parts viz., sentence generation and generic reply. Sentence generation is done using POS tagger and SimpleNLG java library. Whereas generic replies are the hard coded replies which consists of descriptive replies and chats. The purpose of designing set of generic replies was to make chat more generalized and to minimize grammatical errors generated while generation of descriptive response.

These responses will be stored in different text files related music, books, motivational quotes, movies, hobbies, cooking, etc. By categorizing replies into different types will be essential to fire chats at user on the basis of their likings and interest. This will not only help user to engage into an interesting chat but also help her get distracted from current mood thereby elating her.

SimpleNLG is mainly used to create interrogative type generation and also aids in reducing redundancy in chat responses.

SimpleNLG is a Java library that provides interfaces offering direct control over the realisation process, that is, over the way phrases are built and combined, inflectional morphological operations, and linearisation. It defines a set of lexical and phrasal types, corresponding to the major grammatical categories, as well as ways of combining these and setting various feature values. In constructing a syntactic structure and linearizing it as text with SimpleNLG, the following steps are undertaken:

1. Initialisation of the basic constituents required, with the appropriate lexical items.
2. Using the operations provided in the API to set features of the constituents
3. Combining constituents into larger structures, again using the operations provided in the library.
4. Passing the resulting structure to the lineariser, which traverses the constituent structure, applying the correct inflections and linear ordering depending on the features, before returning the realised string.

Constituents in SimpleNLG can be a mixture of canned and non-canned representations. This is useful in applications where certain inputs can be mapped to an output string in a deterministic fashion, while others require a more flexible mapping to outputs depending, for example, on semantic features and context.

SimpleNLG tries to meet these needs by providing significant syntactic coverage with the added option of combining canned and non-canned strings. Another aim of the engine is robustness: structures which are in complete or not well-formed will not result in a crash, but typically will yield infelicitous, though comprehensible, output. This is a feature that SimpleNLG. A third design criterion was to achieve a clear separation between morphological and syntactic operations. The lexical component of the library, which includes a wide-coverage morphological generator, is distinct from the syntactic component. This makes it useful for applications which do not require complex syntactic operations, but which need output strings to be correctly inflected.

Lexical operations: The lexical component provides interfaces that define a Lexicon, a Morphological Rule, and a Lexical Item, with subtypes for different lexical classes (Noun, Preposition etc). Morphological rules, a re-implementation of those in MORPHG cover the full range of English inflection, including regular and irregular forms1. In addition to the range of morphological operations that apply to them, various features can be specified for lexical items.

Setting the INTERROGATIVE TYPE feature of sentence turns it into a question. Two examples, are shown below. While (1) exemplifies a simple yes/no question, in (2), a WH-constituent is specified as establishing a dependency with the direct object (the house).

(1) s1.setInterrogative(YES NO); (Did the boys leave home?)

(2) s1.setInterrogative(WHERE, OBJECT); (Where did the boys leave?)

In summary, building syntactic structures in SimpleNLG is largely a question of feature setting, with no restrictions on whether representations are partially or exclusively made up of canned strings.

Reference paper has described SimpleNLG, a realisation engine which differs from most tactical generators in that it provides a transparent library to carry out low-level tasks such as inflection and syntactic combination, while making no commitments about input specifications or input-output mappings. The simplicity of use of SimpleNLG is reflected in its community of users. The currently available public distribution, has been used by several groups for three main purposes: (a) as a front-end to NLG systems in projects where realisation is not the primary research focus; (b) as a simple natural language component in user interfaces for other kinds of systems, by researchers who do not work in NLG proper.

3.3.6 OpenNLP for sentiment analysis

We have trained and used opennlp.tools.doccat which comprise of Class DoccatModel.

The Apache OpenNLP Document Categorizer can be used to classify text into pre-defined categories. This is achieved by using the maximum entropy algorithm, also named MaxEnt. The algorithm constructs a model based on the same information as the naive Bayes algorithm, but uses a different

approach toward building the model. While naive Bayes assumes the feature independence, MaxEnt uses multinomial logistic regression to determine the right category for a given text.
The entropy is a term used in the context of information theory and measures the uncertainty of an information content. Let’s consider the example of a coin toss. When the coin is fair, that is, when the probability of heads is the same as the probability of tails, then the entropy of the coin toss is as high as it could be. This is because there is no way to predict the outcome of the coin toss ahead of time the best we can do is predict that the coin will come up heads, and our prediction will be correct with probability 1/2. Such a coin toss has one bit of entropy since there are two possible outcomes that occur with equal probability, and learning the actual outcome contains one bit of information. Contrarily, a coin toss with a coin that has two heads and no tails has zero entropy since the coin will always come up heads, and the outcome can be predicted perfectly.
The Maximum Entropy principle can be formulated as follows: given a collection of facts, choose a model which is consistent with all the facts, but otherwise as uniform as possible. The same principle is used also by this OpenNLP algorithm: from all the models that fit our training data, selects the one which has the largest entropy.

Chapter 4

PROBLEM DEFINITION AND SCOPE

4.1 PROBLEM STATEMENT

To create an intelligent therapist chatter bot mobile application which will converse with user by first analysing the sentiment or mood of the user and thereby initializing a positive, negative or neutral chat format to motivate and elate the user.

4.1.1 Goals and Objectives

- To initialize the chat by asking a set of questions
- To categorize mood of user according to obtained result in 3 categories namely positive, negative and neutral
- To create an intelligent chat system which will make user happy or will make user feel motivated to focus on other positive things
- To reduce the redundancy in chat content to make chat interesting

4.1.2 Statement of Scope

- Input : Textual sentence in English language
- Input Bound : Sentence end with either period or question mark
- Output : Textual chat based on user’s sentiment analysis in English language
- Major inputs : User’s reply , user’s response to ELATE’S reply
- Input dependency : ELATE’s reply depend on user’s input
- Elate chatter bot is created for women from age group 18 to 24
- Chat topics are not generalized it is limited to finite topics
- Sentiment analysis is limited to only 3 categories: positive, negative and neutral

4.2 MAJOR CONSTRAINTS

- Sentiment analysis may not be completely correct

- Chat history is not saved
- Chat is limited to certain topics and is mainly focuses on users who are negative or positive
- Redundancy is not reduced to 100%

4.3 METHODOLOGIES OF PROBLEM SOLVING AND EFFICIENCY ISSUES

To deliver a synchronous chat we used client-server architecture to reduce the size of application. All chat related data, files and codes are stored at server side. To minimize redundancy in chat we are using SimpleNLG library and OpenNLP for sentiment analysis.Set of multiple choice questions with sentiment wise scoring is designed to analyse the user’s mood and accordingly chat is triggered. Generic responses are used to make chat more generalised and less mundane.RNN neural network can be incorporated in future to bring dynamicity in chat.

4.4 Outcome

An intelligent therapist chatter bot mobile application.

4.4 SCENARIO IN WHICH MULTI-CORE, EMBEDDED AND DISTRIBUTED COMPUTING USED

4.5 APPLICATION

Applications includes

- Counselling
- Therapist chatter bot
- Intelligent chatter bot
- Online Chatting
- Motivation app
- Educational app

4.7 HARDWARE RESOURCES REQUIRED

Sr.No.	Parameter	Minimum Requirement	Justification
1	CPU Speed	Quad core 1.2 GHz	Java based program
2	RAM	2 GB	Java based program
3	Android SDK	VERSION 15	Application requirement
4	Android version	4.0.3 (Ice-cream Sandwich)	Application requirement

Table 4.1: Hardware Requirements

4.8 SOFTWARE RESOURCES REQUIRED

Platform:

1. Operating System : Windows 8
2. IDE : Eclipse oxygen, Intellij
3. ProgrammingLanguage:Java

Chapter 5

PROJECT PLAN

5.1 PROJECT ESTIMATES

This project is based on the waterfall model of software development which was the prevalent software development process.

The waterfall model is a sequential design process, used in software development processes, in which progress is seen as flowing steadily downwards through the phases of conception, initiation, analysis, design, construction, testing, implementation and maintenance.

In starting phase, we check the feasibility using concept of knowledge canvas and IDEA matrix, after that we identify the type of problem. This project belongs to the NP-complete problem. We designed our project using UML diagram. On the basis of this UML, we started the implementation.

5.1.1 Reconciled Estimates

5.1.1.1 Cost Estimates

COCOMO model - I has been used to estimate the cost and effort required for the project. COCOMO models depend upon the two main equations:

1. Development Effort : MM = a * KLOC* b

Which is based on MM - man-month / person month / staff-month is one month of effort by one person.

2. Efforts and Development Time (TDEV) : TDEV = 2.5 * MM

Note: The coefficients a, b and c depend on the mode of the development. BA-SIC COCOMO **MODEL** : Basic COCOMO Model is good for quick, early, rough order of magnitude estimate of software cost. It does not account for differences in hardware constraints, personal Quality and experience, use of modern tools and techniques, and other project attribute known to have a significant influence on software cost, which limits its accuracy. It gives an approximate estimate of the project parameters. KLOC is the estimated size of the software product expressed in Kilo Lines of Code. Tdev is the estimated time to develop the software, expressed in months. Effort is the total effort required to develop the software product, expressed in person months(PMs). The effort and the duration estimations obtained using the COCOMO model are called as nominal effort estimate and nominal duration estimate. The effort estimation is expressed in units of person-months (PM). It is the area under the person month plot.

Estimation of development effort: Semi-Detached:

Effort = 3.0(KLOC)**1.12 PM

PM: Person Months

Estimation of development time: Semi-detached:

Tdev = 2.5(Effort)**0.35 Months

Cost of project=Tdev*Monthy cost per person

IN CASE OF OUR PROJECT:

Monthly cost per person - Rs. 4000

KLOC = 21(Assumed at this time)

Aper the basic COCOMO estimation formula for semi-detached software:

Effort = 3.0 * (KLOC)**1.12 = 90.78 PM

Nominal development time(Tdev) = 2.5 * (Effort)**0.35 = 12.11 months

Cost required to develop the product = Tdev * 4000 = Rs. 48440

5.1.1.2 Time Estimates

Time has already been allocated as per University Guideline.

Time Estimation is the approximate time required for completion of project.

Time Estimation depends on two factors:

1.Project size (lines of code)

2.Type of project: Android application (java-based)

Activity	Resource	Optimistic	Most Likely	Pessimistic	Estimated Activity Duration	Reserve
Coding	All Members	40 hrs	43 hrs	50 hrs	45 hrs	+/- 10 hrs
Integrate	All Members	10 hrs	12 hrs	20 hrs	15 hrs	+/- 2 hrs
Testing	All Members	25 hrs	28 hrs	35 hrs	30 hrs	+/- 5 hrs
Debugging	All Members	25 hrs	28 hrs	35 hrs	30 hrs	+/- 5 hrs

Table 5.1: Time Estimates

5.1.2 Project Resources

In project management terminology, resources are required to carry out the project task. We divided our project among four persons in each and assigned different modules. This project requirements are:

1. Java IDE Eclipse
2. Programming Language : Java
3. Intellij IDE
4. Files directory structure
5. Android studio
6. Retrofit library
7. OpenNLP library
8. SimpleNLG library

5.2 RISK MANAGEMENT WITH RESPECT TO NP HARD

ANALYSIS

This project involves many NLP tasks which are very difficult to implement and time consuming. In order to manage this risk, we are using hard-coded generic responses to minimize time.

5.2.1 Risk Identification

The risks which we have identified till now are:

1. Risk in case a word does not exist in our dataset.
2. Risk in case the word exists, but appropriate reply is not generated.
3. Technology required to build this under research is new and less resources are available.
4. Some functional parameters like making chat generalized is difficult to achieve.
5. Minimizing redundancy cannot be achieved completely.
6. Application is designed for specific gender and age group.

5.2.2 Risk Analysis

ID	Risk Description	Probability	Impact		
			Schedule	Quality	Overall
1	Some functional parameters difficult to quantify	Medium	Medium	Low	Medium
2	Application running slow	Low	Low	Low	Low

Table 5.2: Risk Table

Probability	**Value**	**Description**
High	Probability of occurrence is	>75 %
Medium	Probability of occurrence is	26-75 %
Low	Probability of occurrence is	<25 %

Table 5.3: Risk Probability Definitions

Impact	**Value**	**Description**
Very High	>10%	Schedule Impact or Unacceptable Quality
High	5-10%	Schedule impact or some parts of project have low quality
Medium	<5%	Schedule Impact or barely noticeable degradation in quality Low Impact on schedule or Quality can be incorporated

Table 5.4: Risk Impact Definitions

5.2.3 Overview of Risk Mitigation, Monitoring, Management

Risk ID	1
Risk Description	Risk in case a word does not exist in our dataset.
Category	Dataset storage Limitation
Source	Software Requirement Specification document
Probability	Medium
Impact	Medium
Response	Mitigate
Strategy	Limit the requirements, Well-populated dataset
Risk Status	Identified

Risk ID	2
Risk Description	Risk in case the word exists, but appropriate reply is not generated.
Category	Program logic and sentence generation method
Source	Software Requirement Specification review
Probability	Medium
Impact	Medium
Response	Mitigate
Strategy	Better testing will resolve this issue
Risk Status	Identified

Risk ID	3
Risk Description	Technology required to build this under research is new and less resources are available
Category	Resource limitation
Source	Software Requirement Specification document
Probability	High
Impact	High
Response	Mitigate
Strategy	To limit the requirements
Risk Status	Identified

Risk ID	4
Risk Description	Some functional parameters like making chat generalized is difficult to achieve
Category	Program algorithm and storage limitation
Source	Software Requirement Specification review
Probability	Medium
Impact	Medium
Response	Mitigate
Strategy	To limit chat topics
Risk Status	Identified

Risk ID	5
Risk Description	Minimizing redundancy cannot be achieved completely.
Category	Program logic and sentence generation method
Source	Software Requirement Specification document
Probability	Medium
Impact	Medium
Response	Mitigate
Strategy	To use generic responses and RNN algorithm
Risk Status	Identified

Risk ID	6
Risk Description	Application is designed for specific gender and age group
Category	Deployment
Source	Software Requirement Specification review
Probability	High
Impact	High
Response	Mitigate
Strategy	Research regarding depression psychology of both genders
Risk Status	Identified

5.3 PROJECT SCHEDULE

5.3.1 Project task set

Major Tasks in the Project stages are:

1. **Requirement Gathering:** Gather relevant reference papers. Decide the parameters which need to be considered.
2. Analysis of gathered dataset and selecting appropriate dataset.
3. Designing the system.
4. Actual implementation of design.
5. Testing the system.
6. Debugging the system.
7. Creating statistical meta-data.

5.3.2 Task network

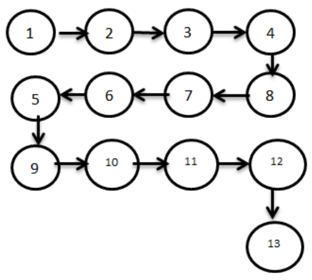
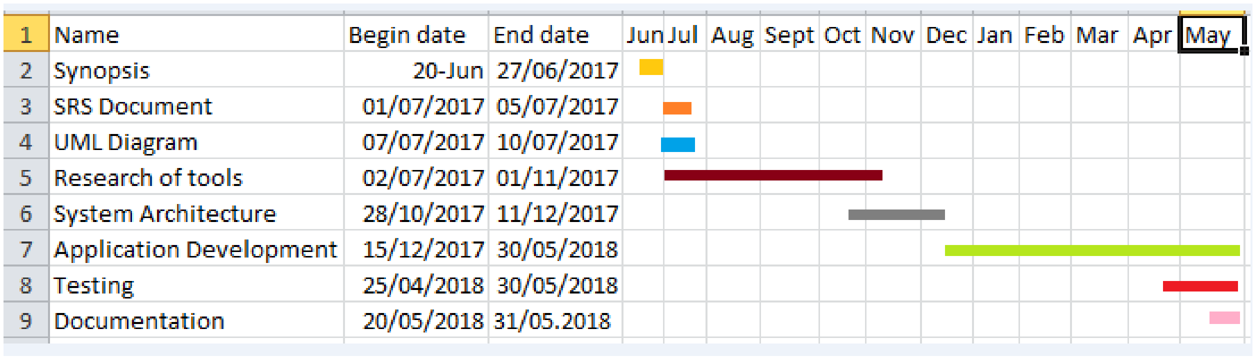


Figure 5.1: Dependencies between project tasks

Various functional modules are:

1. GUI
2. Questionnaire
3. Result
4. Initialization of chat on the basis result
5. User’s response
6. Sentiment analysis
7. Tokenization
8. Entity Recognition
9. Sentence Generation
10. Elate’s response
11. User’s response
12. Generic reply
13. Terminate

5.3.3 Timeline Chart



1	Name	Begin date	End date	Jun	Jul	Aug	Sept	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May
2	Synopsis	20-Jun	27/06/2017	🟨											
3	SRS Document	01/07/2017	05/07/2017		🟧										
4	UML Diagram	07/07/2017	10/07/2017		🟦										
5	Research of tools	02/07/2017	01/11/2017		🟥	🟥	🟥	🟥	🟥						
6	System Architecture	28/10/2017	11/12/2017					⬛	⬛	⬛					
7	Application Development	15/12/2017	30/05/2018							🟩	🟩	🟩	🟩	🟩	🟩
8	Testing	25/04/2018	30/05/2018											🟥	🟥
9	Documentation	20/05/2018	31/05.2018												🟪

Figure 5.2: Plan of the Project Execution

5.4 TEAM ORGANIZATION

The team is made of four members.

1. Akanksha Patil
2. Kshitija Pandit
3. Aishwarya Nerlekar
4. Anuja Tatpuje

Various tasks were divided among these members.

5.4.1 Team structure

Division of tasks is shown.

- Member 1 :- Research, Documentation ,Planning, Development
- Member 2 :- Development, Research ,Planning ,Testing, UI Development

- Member 3 :- Research, Documentation ,Planning ,UI Development
- Member 4 :- Development, Research ,Planning ,Testing

5.4.2 Management reporting and communication

A central could-based repository is used to backup and propagate files and data among team members and guides. Eg. Gmail and Google drive
Regular updates are conveyed using popular market tools. Eg. Gmail

Chapter 6

SOFTWARE REQUIREMENT SPECIFICATION

6.1 INTRODUCTION

6.1.1 Purpose and Scope of Document

Purpose of this intelligent chatter bot is to create a chatter bot which will cheer up user when user is not happy. This system is specifically designed for women from age group 18 to 24.

.

6.1.2 Overview of responsibilities of Developer

The following are the responsibilities of the developer:

1. Efficient use of hardware.
2. Use of licensed software.
3. To develop the product in stipulated time.

6.2 USAGE SCENARIO

A use case diagram at its simplest is a representation of a user interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of software engineering diagrams as well.

6.2.1 User profiles

The profiles of all user categories are described as below:

- User - The user of the system
- System - ELATE

6.2.2 Use-cases

All use-cases for the software are presented as below:

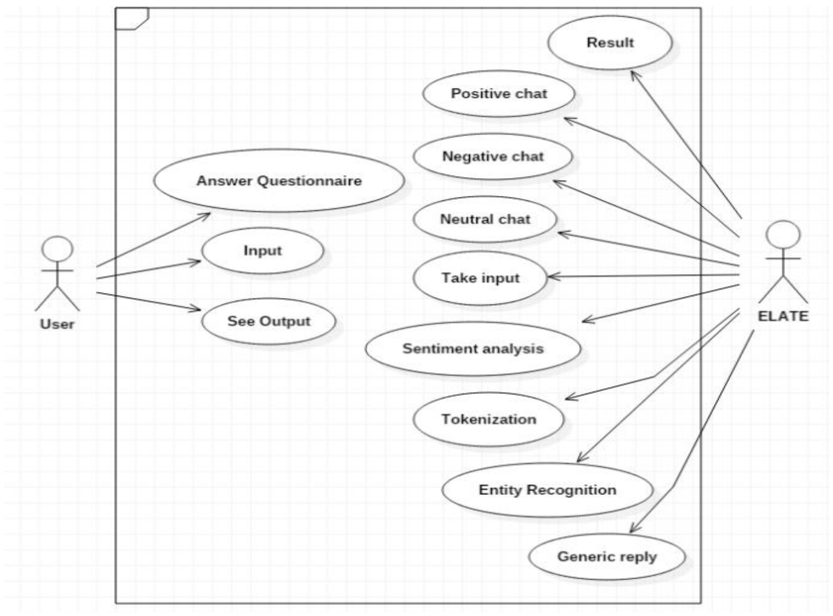


Figure 6.1: Use Case Diagram 1 - Functional Use-Case

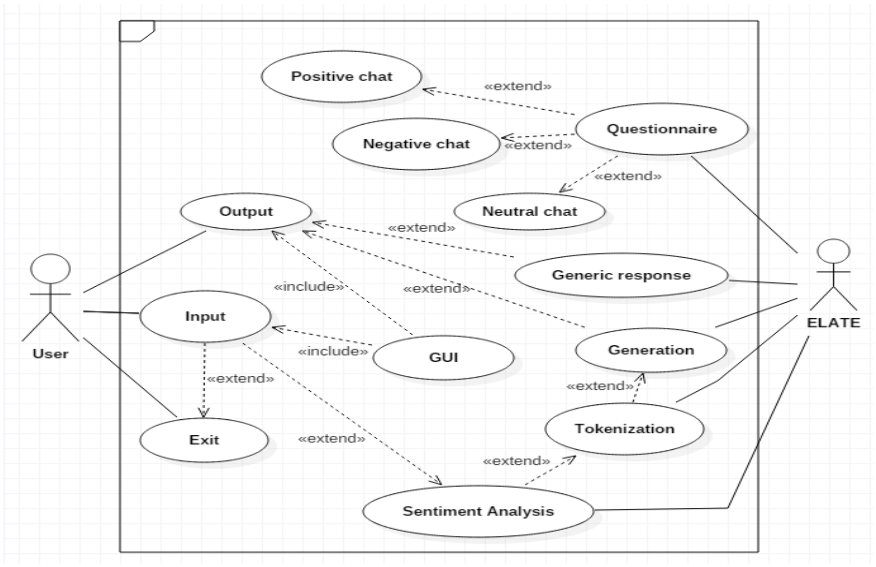


Figure 6.2: Use Case Diagram 2 - Structural Use-Case

6.2.3 Use Case View

Two types of views are shown.

- Functional View
- Structural View

6.3 DATA MODEL AND DESCRIPTION

Data Model Used: - Inverted File Model

Description: - A data store built with the inverted file structure is designed to facilitate fast full text searches. In this model, data content is indexed as a series of keys in a lookup table, with the values pointing to the location of the associated files. This structure can provide nearly instantaneous reporting in finite data and analytics, for instance.

6.3.1 Data Description

File Format: - Text Files (.txt)

Encoding: - UTF-8

Data is kept in a prescribed format in text files with UTF-8 encoding.

Each file has a different association and respectively the format templates are decided.

6.3.2 Data Objects and Relationships

A list of files and associated processes are described.

- story.txt - Chat file for negative mood
- quotes.txt - Chat file for negative mood
- travelling.txt - Chat file for negative mood
- books.txt - Chat file for negative mood
- video_links.txt - Chat file for negative mood
- movies_suggestion.txt - jokes.txt - Chat file for negative mood
- initiallization.txt - GUI
- sentiments.txt – Sentence generation

6.4 FUNCTIONAL MODEL AND DESCRIPTION

6.4.1 Data Flow Diagram

6.4.1.1 Level 0 Data Flow Diagram

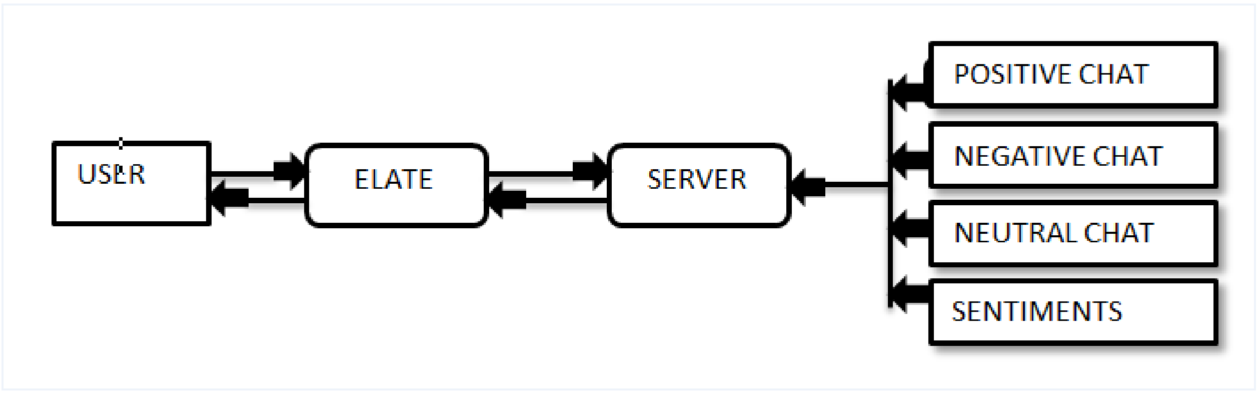


Figure 6.3: Level 0 Data Flow Diagram - Context Diagram

6.4.1.2 Level 1 Data Flow Diagram

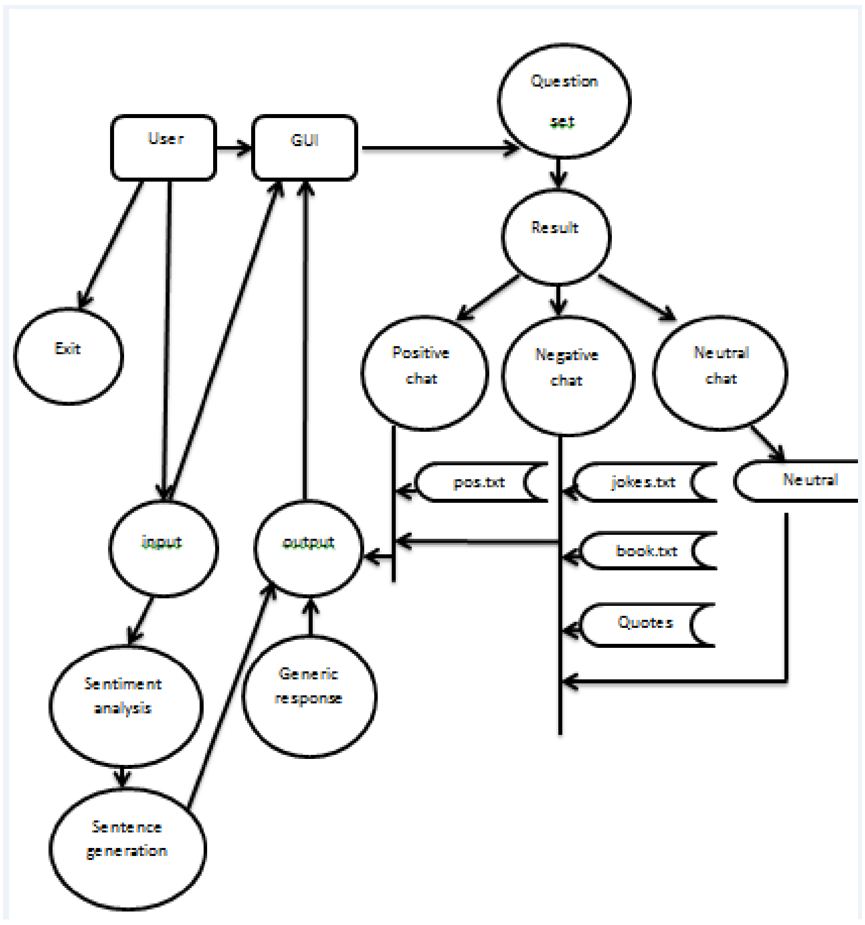
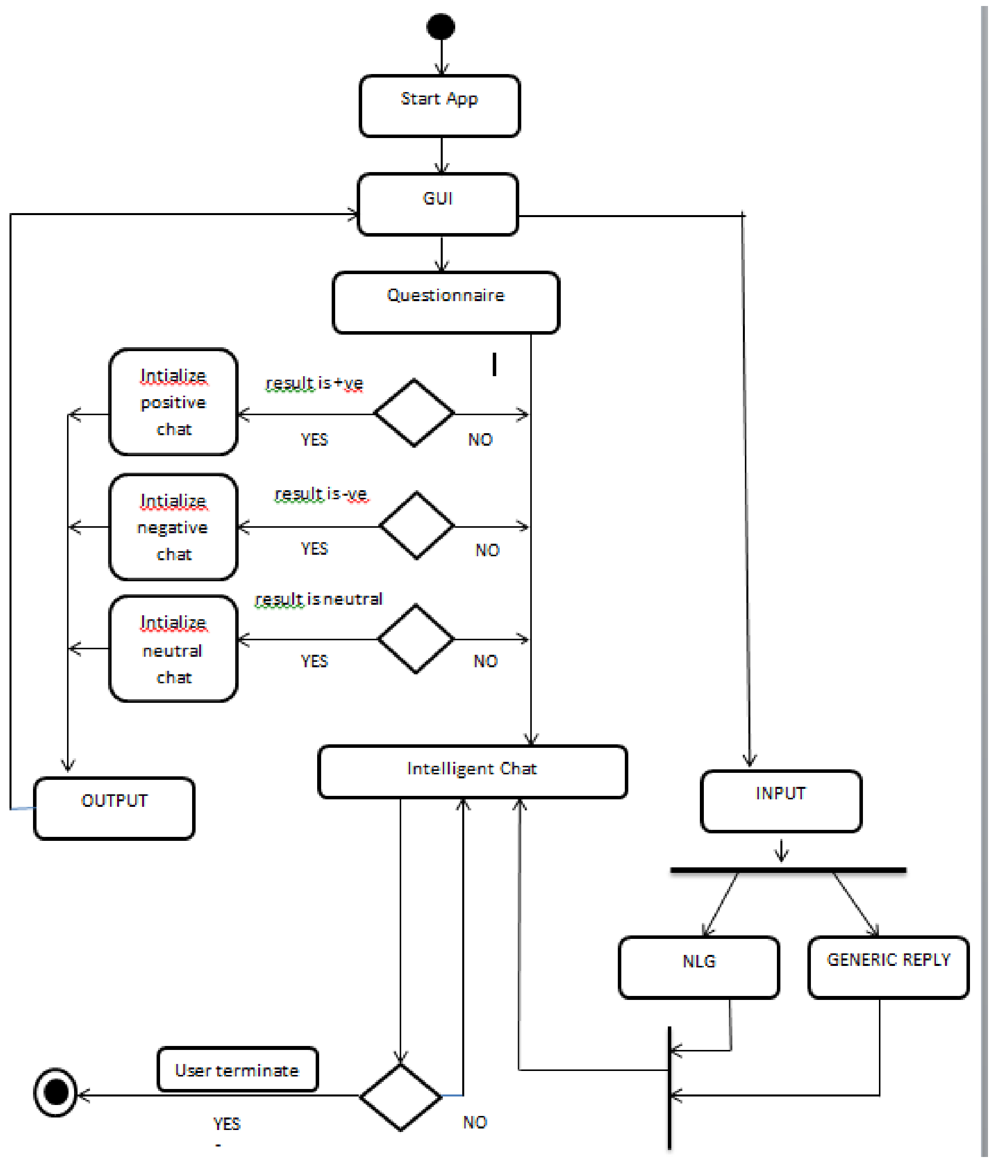


Figure 6.4: Level 1 Data Flow Diagram

6.4.2 Activity Diagram



6.4.3 Non Functional Requirements

6.4.3.1 Performance Requirements

- The system should not crash or show any lag in any case.
- The data i.e. the chat responses should be available in real time.
- Android SDK and version requirement should be fulfilled

6.4.3.2 Safety Requirements

Input should be in textual format and in English language.

Gibberish replies or reply containing emoticons or other symbols won't be responded by system.

6.4.3.3 Software Quality Attributes

- Adaptable: Different types of chat topics are analysed and thereby analysed and responded.
- Correctness: Input text is limited to usage of English language and non-usage of emoticons or any other type of symbols.
- Flexibility: Some exceptions like chat repetition or erroneous chat will be handled.
- Fault-tolerance: Application is highly responsive and gives prompt reply.

6.4.4 State Diagram

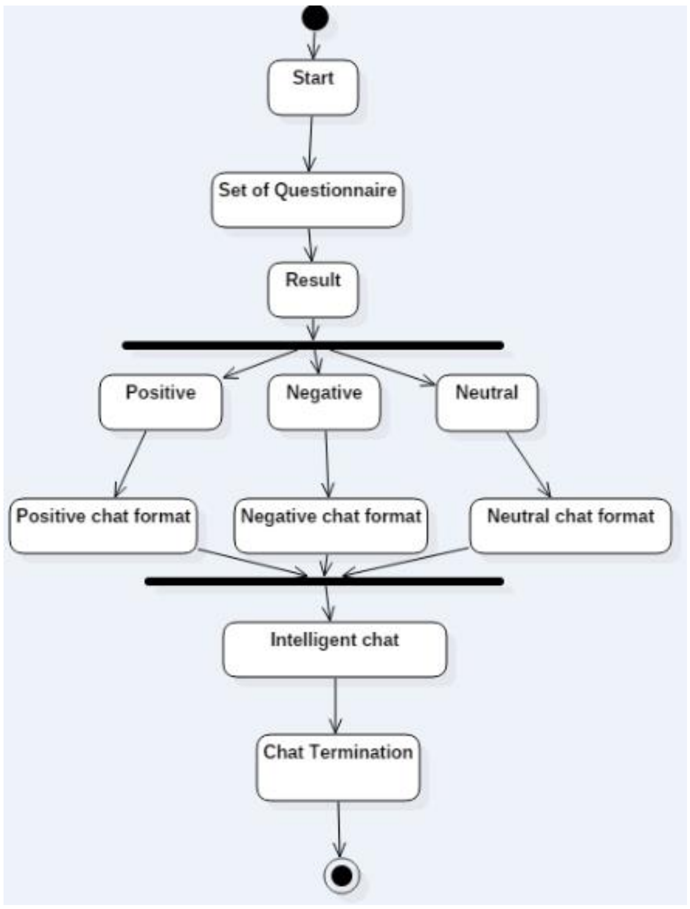


Figure 6.6: State Diagram

6.4.5 Design Constraints

1. Chat system is not generalized and covers only finite topics
2. Application should be able to handle huge data-sets in text file format
3. Some responses need to be hard-code
4. Application is gender and age specific

6.4.6 Software Interface Description

- **Interface Type :-** Android based Graphical UI
- **Interface Input Type :-** Text-based
- **Interface Output Type :-** Text-based

- **GUI Layout :-** Chat-like
- **Scrolling :-** Enabled and Automatic
- **External Entities Used:-** Intellij IDE and Retrofit library for server side development.

Chapter 7

DETAILED DESIGN DOCUMENT USING APPENDIX A AND B

7.1 INTRODUCTION

This document is used to solve the design related problem of the product.

7.2 ARCHITECTURAL DESIGN

A description of the program architecture is presented. Block diagram below roughly represents the flow of the working:

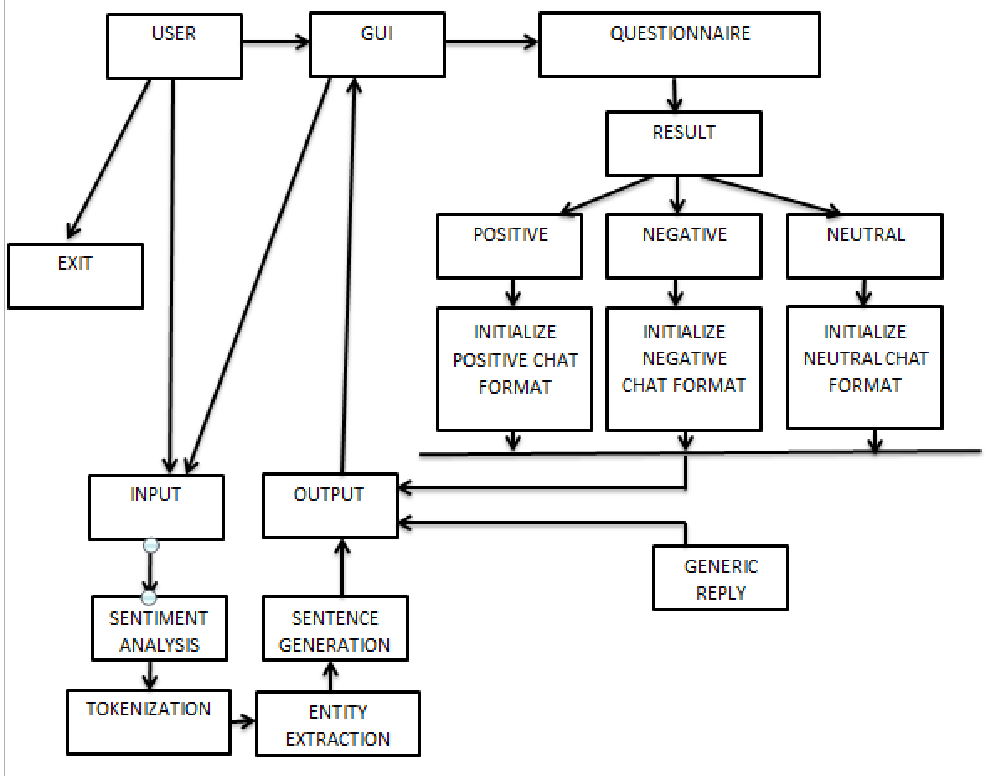


Figure 7.1: Architectural Design

7.3 DATA DESIGN (USING APPENDICES A AND B)

7.3.1 Internal Software Data Structure

String, Array, List

7.3.2 Global Data Structure

String, Integer, Array List, String Array, Boolean, Character, Array, File.

7.3.3 Temporary Data Structure

String, Integer, Array

7.3.4 Database Description

None needed.

7.4 COMPONENT DESIGN

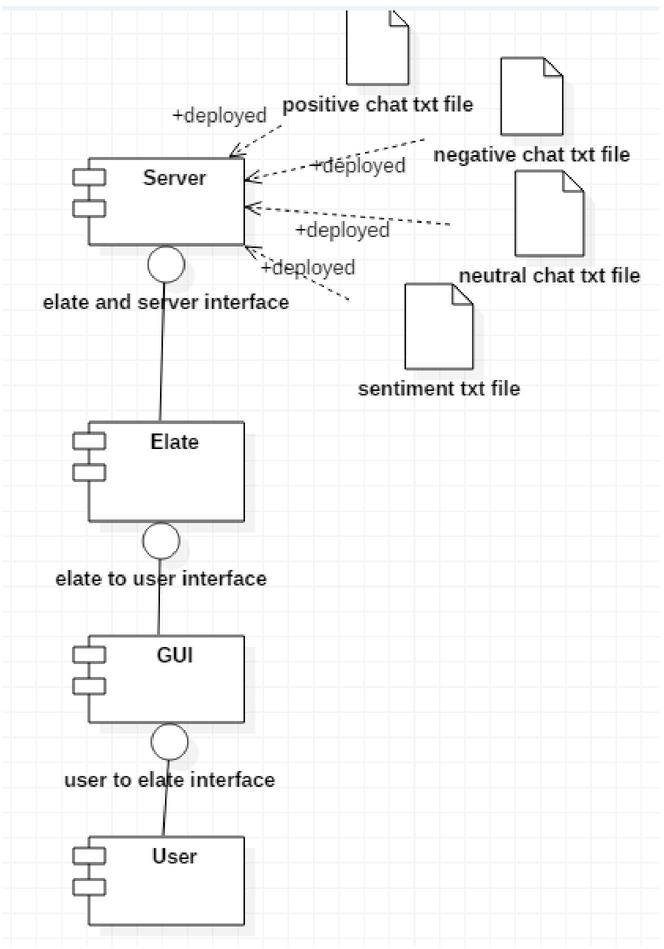


Figure 7.2: Component Design

7.4.1 Class Diagram

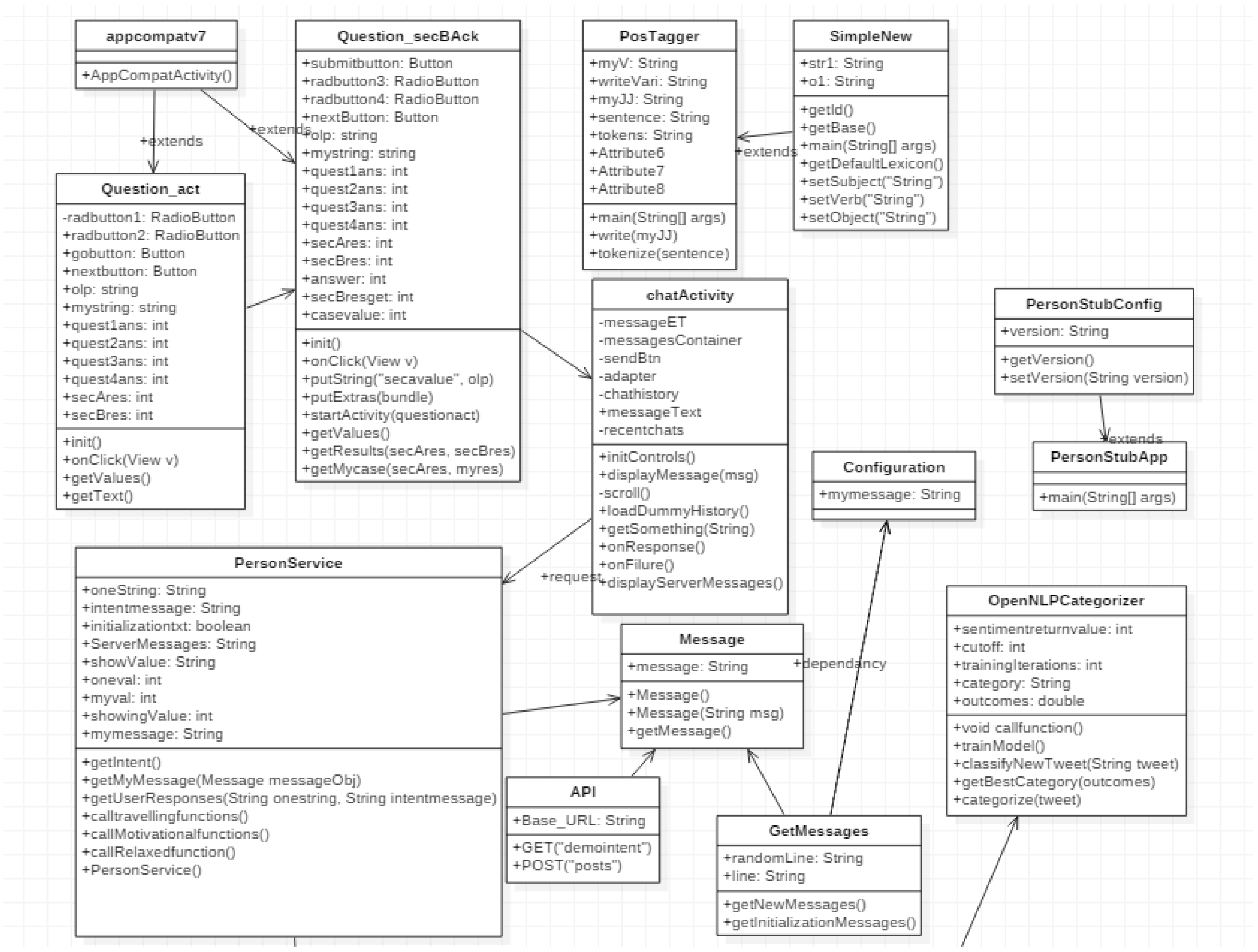


Figure 7.3: Class Diagram

Chapter 8

PROJECT IMPLEMENTATION

8.1 INTRODUCTION

The implementation is done using JAVA Eclipse IDE in programming language JAVA for sentence/reply analyzation and generation. Intellij IDE and Retrofit library is used for server side development. This implementation contains different modules in different files. It also requires implementation of android studio for application development and deployment.

8.2 TOOLS AND TECHNOLOGIES USED

- Tools:-
  - IDE :- Eclipse
  - IDE :- Intellij
  - IDE :- Android studio
  - Library :- Retrofit
  - Library :- OpenNLP
  - Library :- SimpleNLG

- Technology:-
  - Programming Language :- JAVA

8.3 METHODOLOGIES/ALGORITHM DETAILS

8.3.1 Algorithm

1. Present user with set of multiple choice questions.
2. Obtained result will categorize user's sentiment in 3 groups viz. positive, neutral and negative
3. If result is positive , positive chat will be initialized with user
4. If result is negative , positive chat will be initialized with user to motivate and make user happy

5. If result is neutral , neutral chat will be initialized with user

6. Further user’s response will be sent to sentiment analysis unit

7. Sentiment analysis unit will analyse the sentiment by extracting the keyword and matching against the dataset

8. Keyword extraction will be done by tokenization unit

9. Based on the derived sentiment of user ELATE will respond with an INTERROGATIVE question

10. Question will be generated in sentence generation unit

11. Descriptive user’s input will be responded by generic reply

12. Generic reply will based on user’s selected topic from the list of finite topics shown to her

13. User will terminate the chat with an exit message

14. ELATE will analyse it and will respond with an appropriate exit message

15. Chat will be terminated

8.3.2 Pseudo Code

```
1. User start the application
2. GUI displays set of question
3. User select the options
4. Result of Questionnaire is generated
5. IF (result = =4 or 6)
6. THEN generate Positive Chat
        i. Generate Positive Chat
7. IF (result = = 10 or 12)
8. THEN initialize Negative Chat
        i. Generate Negative Chat
9. ELSE start Generic Chat

10. User input is send to Sentiment Analyser
11. Sentiment Analyser analyse the sentiment keyword
12. Input is then send to Tokenizer
13. Tokens generated are send for POS TAGGING
14. Entity Recognition results are send to NLG unit
15. ELATE’s response to input is generated
16. IF Sentiment Analyser fail to analyse sentiment
17. THEN ELATE’S response is from Generic Chat file
```

18. IF user exit the chat window
19. THEN chat is terminated

8.4 VERIFICATION AND VALIDATION FOR ACCEPTANCE

User’s input is verified and validated before processing it.
Multiple cases are created for valid and invalid input

GenerationMainClass.java

```
package LanguageGeneration;

import java.util.ArrayList;
import java.util.List;

import simplenlg.features.Feature;
import simplenlg.features.InterrogativeType;
import simplenlg.framework.LexicalCategory;
import simplenlg.framework.NLGElement;
import simplenlg.framework.NLGFactory;
import simplenlg.framework.WordElement;
import simplenlg.lexicon.Lexicon;
import simplenlg.phrasespec.SPhraseSpec;
import simplenlg.realiser.english.Realiser;


public class GenerationMainClass extends PosTaggerNew
{
    Paragraph p=new Paragraph();
    static SPhraseSpec p1;
    static SPhraseSpec p2;
    Realiser r1;
    String output1;
    String output2;
    String message;
    GenerationMainClass gener=new GenerationMainClass();

    public GenerationMainClass()
    {

    }

    PosTaggerNew ps=new PosTaggerNew();
 //   String getVar=ps.getmyvariable();



    public List<String> getData()
```

```
{
   String[] array=new String[5];

   String demovar=ps.getmyvariable();
   System.out.println("Got verb=="+demovar);



   Lexicon lexicon = Lexicon.getDefaultLexicon();
   NLGFactory nlgFactory = new NLGFactory(lexicon);
   Realiser realiser = new Realiser(lexicon);



   p1= nlgFactory.createClause();
   p2= nlgFactory.createClause();









   WordElement word= new WordElement(demovar, LexicalCategory.ADJECTIVE);



   p1.setSubject("you");
   p1.setVerb("are");
   p1.addModifier(demovar);
   p1.setFeature(Feature.INTERROGATIVE_TYPE, InterrogativeType.WHY);

   Realiser r1=new Realiser();
   String output2=r1.realiseSentence(p1);
   //System.out.println(output2);






   SPhraseSpec p1 = nlgFactory.createClause();

   NLGElement s2=nlgFactory.createSentence("Don't worry");
   p1.addComplement("Everything will be alright.");
   p1.addComplement("Just be positive :) ");

   String ops=realiser.realiseSentence(s2);
   String myget=realiser.realiseSentence(p1);



   String oops=ops+myget;
   String message=p.getParagraph();

   SPhraseSpec p = nlgFactory.createClause();
```

```
        p.setSubject("I");
        p.setVerb("have");
        p.setObject("many relaxation techniques for you!!");




        List<String> list=new ArrayList<String>();

        Realiser r = new Realiser();


        array[0]=output2;
        array[1]=oops;
        array[2]=message;
        array[3]= r.realiseSentence(p);
        for(int i=0;i<array.length;i++)
        {
            if(array[i]!=null)
                list.add(array[i]);
        }



        return list;
    }
}


package LanguageGeneration;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.File;
import java.io.FileWriter;
import java.io.BufferedWriter;


import opennlp.tools.postag.POSModel;
import opennlp.tools.postag.POSTaggerME;
import opennlp.tools.tokenize.Tokenizer;
import opennlp.tools.tokenize.TokenizerME;
import opennlp.tools.tokenize.TokenizerModel;


public class PosTaggerNew {
    public String myV;
    public static String writeVari;
    public static String demonvariable;
    public String demovar;
    public PosTaggerNew()
    {

    }
    public PosTaggerNew(String message)
    {
        this.demonvariable=message;
        System.out.println("demon variable is "+demonvariable);
    }
```

PosTaggerNew.java

```
    public String getmyvariable() {

        String myJJ;
        InputStream tokenModelIn = null;
        InputStream posModelIn = null;

//      System.out.println("getMyMessage is null"+demonvariable);

        try {
            String sentence = demonvariable;
            // tokenize the sentence
            tokenModelIn = new FileInputStream("getInput/en-token.bin");
            TokenizerModel tokenModel = new TokenizerModel(tokenModelIn);
            Tokenizer tokenizer = new TokenizerME(tokenModel);
            //(tokenModel);
            String tokens[] = tokenizer.tokenize(sentence);

            // Parts-Of-Speech Tagging
            // reading parts-of-speech model to a stream
            posModelIn = new FileInputStream("getInput/en-pos-maxent.bin");
            // loading the parts-of-speech model from stream
            POSModel posModel = new POSModel(posModelIn);
            // initializing the parts-of-speech tagger with model
            POSTaggerME posTagger = new POSTaggerME(posModel);
            // Tagger tagging the tokens
            String tags[] = posTagger.tag(tokens);

            for (int i = 0; i < tokens.length; i++) {
                if (tags[i].equalsIgnoreCase("JJ")) {
                    myJJ = tokens[i];
                    writeVari = tokens[i];
                }
                if (tags[i].equalsIgnoreCase("VB")) {
                    myJJ = tokens[i];
                    writeVari = tokens[i];
                }
                if (tags[i].equalsIgnoreCase("VBD")) {
                    myJJ = tokens[i];
                    writeVari = tokens[i];
                }
                if (tags[i].equalsIgnoreCase("VBG")) {
                    myJJ = tokens[i];
                    writeVari = tokens[i];
                }

                if (tags[i].equalsIgnoreCase("VBN")) {
                    myJJ = tokens[i];
                    writeVari = tokens[i];
                }
                if (tags[i].equalsIgnoreCase("VBP")) {
                    myJJ = tokens[i];
                    writeVari = tokens[i];
                }
                if (tags[i].equalsIgnoreCase("VBZ")) {
                    myJJ = tokens[i];
                    writeVari = tokens[i];
                }


            }

        } catch (IOException e) {
            // Model loading failed, handle the error
            e.printStackTrace();
        } finally {
            if (tokenModelIn != null) {
                try {
                    tokenModelIn.close();
                } catch (IOException e) {
                }
            }
            if (posModelIn != null) {
                try {
                    posModelIn.close();
                } catch (IOException e) {
                }
            }
        }
        return writeVari;

    }
}
```

Paragraph.java

```
package LanguageGeneration;

import java.util.Arrays;


import simplenlg.features.Feature;
import simplenlg.features.Tense;
import simplenlg.framework.DocumentElement;
import simplenlg.framework.InflectedWordElement;
import simplenlg.framework.LexicalCategory;
import simplenlg.framework.NLGFactory;
import simplenlg.framework.WordElement;
import simplenlg.lexicon.Lexicon;
import simplenlg.lexicon.XMLLexicon;
import simplenlg.phrasespec.SPhraseSpec;
import simplenlg.realiser.english.Realiser;
import simplenlg.xmlrealiser.XMLRealiser;



public class Paragraph
{
    public String getParagraph()
    {
        Lexicon lexicon = Lexicon.getDefaultLexicon();
        NLGFactory nlgFactory = new NLGFactory(lexicon);
        Realiser realiser = new Realiser(lexicon);

        //SPhraseSpec p1 = NLGFactory.createClause("Mary", "chase", "the monkey");
        SPhraseSpec p2 = nlgFactory.createClause("I","can", "help you.");
        p2.addComplement(" You can tell me more about your feelings");

        //SPhraseSpec p1=nlgFactory.createClause("You can"," tell me more about your feelings");
        //p1.setFeature(Feature.TENSE, Tense.PAST);

        SPhraseSpec p3 = nlgFactory.createClause("I", "will", "try my best for you");

        // DocumentElement s1 = nlgFactory.createSentence(p1);
        DocumentElement s2 = nlgFactory.createSentence(p2);
        DocumentElement s3 = nlgFactory.createSentence(p3);

        DocumentElement par1 = nlgFactory.createParagraph(Arrays.asList(s2, s3));

        String output = realiser.realise(par1).getRealisation();
//                  System.out.println(output);

        return output;

    }
}
```

FetchingVaues.java

```
package LanguageGeneration;

import java.util.ArrayList;
import java.util.List;
import simplenlg.features.Feature;
import simplenlg.features.InterrogativeType;
import simplenlg.framework.LexicalCategory;
import simplenlg.framework.NLGElement;
import simplenlg.framework.NLGFactory;
import simplenlg.framework.WordElement;
import simplenlg.lexicon.Lexicon;
import simplenlg.phrasespec.SPhraseSpec;
import simplenlg.realiser.english.Realiser;

public class FetchingVaues extends PosTaggerNew
{
                Paragraph p=new Paragraph();
                static SPhraseSpec p1;
                static SPhraseSpec p2;
    Realiser r1;
    String output1;
    String output2;

    PosTaggerNew ps=new PosTaggerNew();

    public List<String> getData()
                {
                                String[] array=new String[5];

                    String demovar=ps.getmyvariable();
                            System.out.println("Got verb=="+demovar);



                                            Lexicon lexicon = Lexicon.getDefaultLexicon();
                                NLGFactory nlgFactory = new NLGFactory(lexicon);
                                Realiser realiser = new Realiser(lexicon);



                                p1= nlgFactory.createClause();


                                                p1.setSubject("you");
                                                p1.setVerb("are");
                                                p1.addModifier(demovar);
                                                p1.setFeature(Feature.INTERROGATIVE_TYPE, InterrogativeType.WHY);

                            Realiser r1=new Realiser();
                            String output2=r1.realiseSentence(p1);




                                System.out.println(output1);




                                WordElement word= new WordElement(demovar, LexicalCategory.ADJECTIVE);
```

```
                        SPhraseSpec p1 = nlgFactory.createClause();

                                        NLGElement s2=nlgFactory.createSentence("Don't worry");
                                        p1.addComplement("Everything will be alright.");
                                        p1.addComplement("Just be positive :) ");

                                        String ops=realiser.realiseSentence(s2);
                                        String myget=realiser.realiseSentence(p1);


                                                String oops=ops+myget;
                                                String message=p.getParagraph();

                                        SPhraseSpec p = nlgFactory.createClause();
                                        p.setSubject("I");
                                        p.setVerb("have");
                                        p.setObject("many relaxation techniques for you!!");



                                List<String> list=new ArrayList<String>();

                                Realiser r = new Realiser();

                                                array[0]=output2;
                                                array[1]=oops;
                                                array[2]=message;
                                                array[3]= r.realiseSentence(p);

                                                        for(int i=0;i<array.length;i++)
                                        {
                                                if(array[i]!=null)
                                                        list.add(array[i]);
                                        }



return list;
        }

}
```

IntentMessageGetter.java

```
package LanguageGeneration;
```

```
import javax.print.DocFlavor;

public class IntentMessageGetter
{
    String demoString;
    public void setmessage(String message)
    {
      this.demoString=message;
    }
    public String setDemoString()
    {
        return demoString;
    }
}
```

PositiveChatGeneration.java

```
package LanguageGeneration;

import javafx.geometry.Pos;
import simplenlg.features.Feature;
import simplenlg.features.InterrogativeType;
import simplenlg.framework.LexicalCategory;
import simplenlg.framework.NLGElement;
import simplenlg.framework.NLGFactory;
import simplenlg.framework.WordElement;
import simplenlg.lexicon.Lexicon;
import simplenlg.phrasespec.SPhraseSpec;
import simplenlg.realiser.english.Realiser;

import java.util.ArrayList;
import java.util.List;

public class PositiveChatGeneration extends PosTaggerNew {
//    Paragraph p = new Paragraph();
    static SPhraseSpec p1;
    static SPhraseSpec p2;
    static SPhraseSpec p3;

    Realiser r1;
    String output1;
    String output2;

    PosTaggerNew ps = new PosTaggerNew();

    public List<String> getData() {
        String[] array = new String[5];

        String demovar = ps.getmyvariable();
        System.out.println("Got verb==" + demovar);
```

```
      Lexicon lexicon = Lexicon.getDefaultLexicon();
      NLGFactory nlgFactory = new NLGFactory(lexicon);
      Realiser realiser = new Realiser(lexicon);


      p1 = nlgFactory.createClause();
      p3 = nlgFactory.createClause();

      p1.setSubject("you");
      p1.setVerb("are");
      p1.addModifier(demovar);
      p1.setFeature(Feature.INTERROGATIVE_TYPE, InterrogativeType.WHY);

      Realiser r1 = new Realiser();
      String output2 = r1.realiseSentence(p1);


      p3.setSubject("I");
      p3.setVerb("am");
      p3.setObject("happy for you");

      Realiser r2 = new Realiser();
      String output3 = r1.realiseSentence(p3);


      System.out.println(output3);


      WordElement word = new WordElement(demovar, LexicalCategory.ADJECTIVE);


      SPhraseSpec p1 = nlgFactory.createClause();

      NLGElement s2 = nlgFactory.createSentence("That's great");
      p1.addComplement("keep it up!!!!!!!!");
      p1.addComplement("Just stay positive:)   ");

      String ops = realiser.realiseSentence(s2);
      String myget = realiser.realiseSentence(p1);


      String oops = ops + myget;
//      String message = p.getParagraph();

      SPhraseSpec p11 = nlgFactory.createClause();
      p11.setSubject("I");
      p11.setVerb("am");
      p11.setObject("wishing you great days ahead");


      List<String> list = new ArrayList<String>();
```

```
        Realiser r = new Realiser();


        array[0] = output2;
        array[1] = output3;
        array[2] = oops;
       array[3] = r.realiseSentence(p11);

        for (int i = 0; i < array.length; i++) {
            if (array[i] != null)
                list.add(array[i]);
        }

        return list;

    }
}
```

OpenNLPCategorizer.java

```
package SentimentAnalyzer;

import opennlp.tools.doccat.DoccatModel;
import opennlp.tools.doccat.DocumentCategorizerME;
import opennlp.tools.doccat.DocumentSampleStream;
import opennlp.tools.util.ObjectStream;
import opennlp.tools.util.PlainTextByLineStream;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;

public class OpenNLPCategorizer {
   static DoccatModel model;
    int sentimentreturnvalue;




    public static void callfunction()
    {
        trainModel();

    }

    public static void trainModel() {
        InputStream dataIn = null;
        try {
            dataIn = new FileInputStream("input/amazon_cells_labelled.txt");
            ObjectStream lineStream = new PlainTextByLineStream(dataIn, "UTF-8");
            ObjectStream sampleStream = new DocumentSampleStream(lineStream);
            // Specifies the minimum number of times a feature must be seen
            int cutoff = 2;
            int trainingIterations = 30;
            model = DocumentCategorizerME.train("en", sampleStream, cutoff,
                    trainingIterations);
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (dataIn != null) {
                try {
                    dataIn.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }

    public int classifyNewTweet(String tweet) {

            DocumentCategorizerME myCategorizer = new DocumentCategorizerME(model);
            double[] outcomes = myCategorizer.categorize(tweet);
            String category = myCategorizer.getBestCategory(outcomes);

            if (category.equalsIgnoreCase("1")) {
                System.out.println("The tweet is positive :) ");
                sentimentreturnvalue = 1;
            } else {
                System.out.println("The tweet is negative :( ");
                sentimentreturnvalue = 0;
            }
            return sentimentreturnvalue;
        }

}
```

```
package service;

import ApplicationTestData.API;
import ApplicationTestData.GetMessages;
import ApplicationTestData.TrainingModel;
import LanguageGeneration.*;
import SentimentAnalyzer.OpenNLPCategorizer;
import ServerConfig.ConfigurationClass;
import com.codahale.metrics.annotation.Timed;
import com.google.gson.Gson;
import com.google.gson.JsonElement;
import com.google.gson.JsonObject;
import com.google.gson.JsonParser;
import db.PersonDB;
import jdk.nashorn.internal.parser.JSONParser;
import models.Message;
import models.Person;
import org.json.JSONObject;
import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;
import retrofit2.converter.scalars.ScalarsConverterFactory;

import javax.annotation.Generated;
import javax.ws.rs.*;
import javax.ws.rs.core.MediaType;
import javax.xml.stream.Location;
import java.lang.reflect.Type;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;

@Path("/person")
public class PersonService {

    PersonDB person=new PersonDB();
    int showingValue;
    String mymessage,MymessageObject;
OpenNLPCategorizer sentimentanalyzer=new OpenNLPCategorizer();
boolean sentiments=true;
int oneval,myval;
Message message,msg1;
int pos=0;
    GetMessages initialization=new GetMessages();
    GetMessages getMessages=new GetMessages();
   PosTaggerNew posTaggerNew;//=new PosTaggerNew();
    int i=0,f=1;
    int m,g,val,randnum;
    String initialmessage,listmessage,showSentimentVal;

        FetchingVaues fetchingVaues=new FetchingVaues();
    List<String> list=new ArrayList<String>();
    List<String> poslist=new ArrayList<String>();
    PositiveChatGeneration positiveChatGeneration=new PositiveChatGeneration();


    public int getListSize()
    {
        list=fetchingVaues.getData();
        return list.size();
    }

    public int getPoslistSize()
    {
        poslist=positiveChatGeneration.getData();
        return poslist.size();
    }

    public int generateRandomNumber()
    {
        Random rand=new Random();

        int x=rand.nextInt(7);

        return x;
    }

    String ServerMessages;
    String showValue;

    public PersonService() {

    }
```

PersonService.java

```
@POST
@Timed
@Path("/posts")
 @Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
public Message getMyMessage(Message messageObj)
{
    String oneString;
    System.out.println(messageObj.getMessage());
    String intentmessage=messageObj.getMessage();
        TrainingModel trainingModel;
        //poslist=getPoslist();



        if(f==1)
        {
            OpenNLPCategorizer.callfunction();
            System.out.println("Call function");
            f++;
        }
        if(!intentmessage.equalsIgnoreCase("bye"))
        {
            if (sentiments == true) {
                showingValue = sentimentanalyzer.classifyNewTweet(intentmessage);
                showValue = Integer.toString(showingValue);
                System.out.println("I am in showvalue"+showValue);
                sentiments = false;
            }
            if (showValue.equalsIgnoreCase("1"))
            {
                posTaggerNew=new PosTaggerNew(intentmessage);
```

```
                 m=getPoslistSize();
                 if(i==0) {
                     System.out.println("" + intentmessage);
                   //  initialmessage = initialization.getInitializationMessages();
                     mymessage = poslist.get(i);
                     ServerMessages = mymessage;
                     message = new Message(ServerMessages);
                     System.out.println("i ka value"+i);
                     System.out.println(poslist.get(i));
                 }
                if(i<m)
                {
                    System.out.println("i ka value "+i);
                    System.out.println("m ka value "+m);
                    ServerMessages = poslist.get(i);
                    System.out.println(ServerMessages);
                    message = new Message(ServerMessages);
                }
                if(i==m)
                {
                    message=new Message("i think i should go now!!!!Bye.See you soon :) ");
                }
                if(i>=m)
                {
                    showingValue = sentimentanalyzer.classifyNewTweet(intentmessage);
                    showValue = Integer.toString(showingValue);

                }
                i++;
                return message;
            }
            else
                {
                oneString = "negative" + showValue;
    negChat:        posTaggerNew=new PosTaggerNew(intentmessage);
                m=getListSize();
                if (i == 0) {
                    System.out.println("Print intent Message "+intentmessage);
                    initialmessage = initialization.getInitializationMessages();
                    mymessage=list.get(i);
                    ServerMessages=initialmessage+"\n"+mymessage;
                    message = new Message(ServerMessages);
                    System.out.println(list.get(i));
                }
                else if(i<m)
                    {
//                         val=i-1;
//                      System.out.println("m ka value "+m);
                    ServerMessages = list.get(i);
                    System.out.println(ServerMessages);
                    message = new Message(ServerMessages);
                    System.out.println("I ka value "+i);
            }
            else
            {
              System.out.println("i ka value "+i);
              if (i % 6 == 0) {
                showingValue = sentimentanalyzer.classifyNewTweet(intentmessage);
                 showSentimentVal = Integer.toString(showingValue);
                 System.out.println("Value is"+showingValue);

                        }
                 if(i%2!=0)
                 {
                    if(!intentmessage.equalsIgnoreCase("no"))
                    {
                       randnum=generateRandomNumber();

                       switch (randnum)
                       {
                          case 0:
                             MymessageObject="Let me tell you one motivational story "+"\n";
                             ServerMessages=MymessageObject+"\n"+getMessages.getstoryMessages()+"\n are you satisfied with this tell me yes
or no\n";
                             message=new Message(ServerMessages);
                             break;
                          case 1:
                             MymessageObject="Let me give you one motivational video links "+"\n";
                             ServerMessages=MymessageObject+"\n"+getMessages.getvideoMessages()+"\nare you satisfied with this tell me yes
or no\n";
                             message=new Message(ServerMessages);

                             break;
                          case 2:
                             MymessageObject="Let me tell you some motivational quotes "+"\n";
                             ServerMessages=MymessageObject+"\n"+getMessages.getquotesMessages()+"\nare you satisfied with this tell me yes
or no\n";
                             message=new Message(ServerMessages);

                             break;
                          case 3:
                             MymessageObject="Let me tell you some jokes "+"\n";
                             ServerMessages=MymessageObject+"\n"+getMessages.getJokesMessages()+"\nare you satisfied with this tell me yes
or no\n";
                             message=new Message(ServerMessages);
                             break;
                          case 4:
                             MymessageObject="Let me tell you some interesting movies "+"\n";
                             ServerMessages=MymessageObject+"\n"+getMessages.getmovieMessages()+"\n are you satisfied with this tell me yes
or no\n";
                             message=new Message(ServerMessages);
                             break;
                          case 5:
                             MymessageObject="Let me tell you about some interesting books "+"\n";
                    ServerMessages=MymessageObject+"\n"+getMessages.getBookMessages()+"\nare you satisfied with this tell me yes or
no\n";
                    message=new Message(ServerMessages);
                    break;
                case 6:
                    MymessageObject="Let me tell you some places to travel"+"\n";
                    ServerMessages=MymessageObject+"\n"+getMessages.gettravellingMessages()+"\nare you satisfied with this tell
me...yes or no\n";
                    message=new Message(ServerMessages);
                    break;
            }

        }
        else if(intentmessage.equalsIgnoreCase("no"))
        {
                ServerMessages="i think i should leave now";
                message=new Message(ServerMessages);
        }

    }
    else
    {
        randnum=generateRandomNumber();
        if(!intentmessage.equalsIgnoreCase("no")) {
            switch (randnum) {
                case 0:
                    MymessageObject = "Let me tell you one motivational story "+"\n";
                    ServerMessages = MymessageObject+"\n"+ getMessages.getstoryMessages();
                    message = new Message(ServerMessages);
                    break;
                case 1:
                    MymessageObject = "Let me give you one motivational video links "+"\n";
                    ServerMessages = MymessageObject+"\n"+ getMessages.getvideoMessages();
                    message = new Message(ServerMessages);

                    break;
                case 2:
                    MymessageObject = "Let me tell you some motivational quotes "+"\n";
                    ServerMessages = MymessageObject+"\n"+ getMessages.getquotesMessages();
                    message = new Message(ServerMessages);

                    break;
                case 3:
                    MymessageObject = "Let me tell you some jokes "+"\n";
                    ServerMessages = MymessageObject +"\n"+ getMessages.getJokesMessages();
                    message = new Message(ServerMessages);
                    break;
                case 4:
                    MymessageObject = "Let me tell you some interesting movies "+"\n";
                    ServerMessages = MymessageObject +"\n"+ getMessages.getmovieMessages();
                    message = new Message(ServerMessages);
                    break;
                    case 5:
                        MymessageObject = "Let me tell you about some interesting books "+"\n";
                        ServerMessages = MymessageObject +"\n"+ getMessages.getBookMessages();
                        message = new Message(ServerMessages);
                        break;
                    case 6:
                        MymessageObject = "Let me tell you some places to travel"+"\n";
                        ServerMessages = MymessageObject +"\n"+getMessages.gettravellingMessages();
                        message = new Message(ServerMessages);
                        break;


                }
              }
              else if(intentmessage.equalsIgnoreCase("no"))
              {
                  ServerMessages="oops i am sorry,i think i have only these many options if you still want to know more options you can type
yes else say bye";
                  message=new Message(ServerMessages);
              }

            }

          }
          i++;
          return message;
        }
      }
      else
      {
          message=new Message("Okez.Take care.Good bye.See you soon");
          return message;
      }

  }



/*    @POST
    @Timed
    @Path("/save")
    @Produces(MediaType.TEXT_PLAIN)
    @Consumes({MediaType.APPLICATION_JSON})
    public String addPerson(Person person) {
        return PersonDB.save(person);
    }*/
}
```

ConfigurationClass.java

```
package ServerConfig;

import ApplicationTestData.API;
import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;
import retrofit2.Retrofit;
import retrofit2.converter.scalars.ScalarsConverterFactory;

public class ConfigurationClass
{
    String mymessage;


}
```

Chapter 9

SOFTWARE TESTING

9.1 TYPE OF TESTING USED

Manual Testing

9.2 TEST CASES AND TEST RESULTS

Title:

Positive chat is initialized when Questionnaire result is positive.

Description:

Chat is initialized with a positive note and context.

Precondition:

User must select the options which convey positive score and hence positive result.'

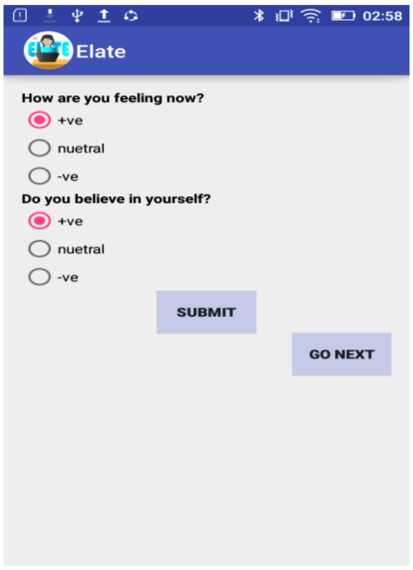
Test Steps:

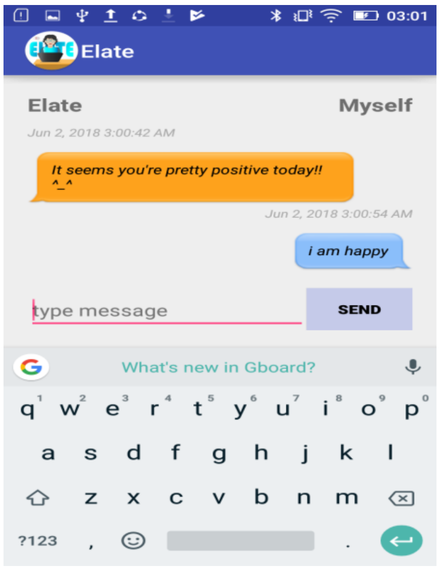
1. User starts the application
2. Questionnaire is shown to user
3. User selects the options
4. Result is generated
5. System analyses the result and deduces that User is positive
6. Positive chat is initialised

Expected Result:

The chatter bot GUI will display chat which will make user continue to stay positive.

Obtained Output





Title:

Chat will try to make user happy and motivated is initialized when Questionnaire result is Negative.

Description:

Chat is initialized with a positive note and context that will be intended to make user happy and feel motivated.

Precondition:

User must select the options which convey negative score and hence negative result.’

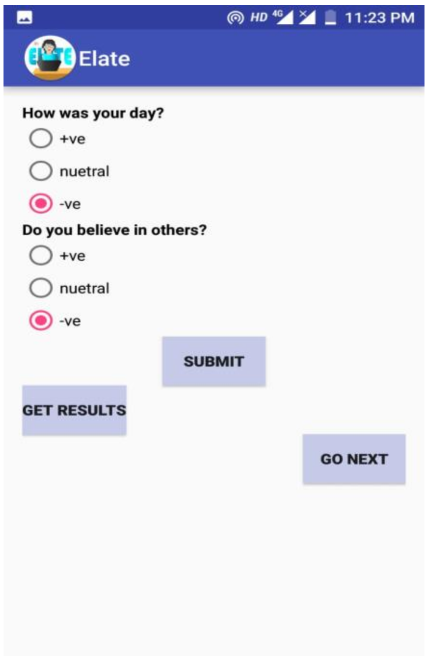
Test Steps:

1. User starts the application
2. Questionnaire is shown to user
3. User selects the options
4. Result is generated
5. System analyses the result and deduces that User is negative
6. Positive chat is initialised

Expected Result:

The chatter bot GUI will display chat which will make user continue to stay positive.

Obtained Output:


Elate

How was your day?

+ve

nuetral

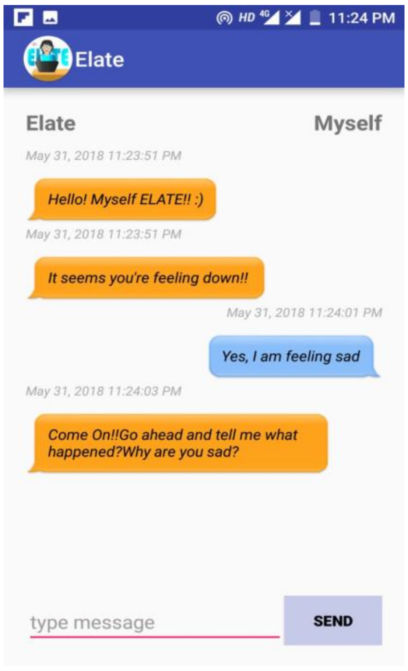
[✓ -ve]

Do you believe in others?

+ve

nuetral

[✓ -ve]


Elate

Elate — Myself

May 31, 2018 11:23:51 PM
Hello! Myself ELATE!! :)

May 31, 2018 11:23:51 PM
It seems you're feeling down!!

May 31, 2018 11:24:01 PM
Yes, I am feeling sad

May 31, 2018 11:24:03 PM
Come On!!Go ahead and tell me what happened?Why are you sad?

Title:

Chat is extended to varied topics like Books, movies, jokes

Description:

ELATE chat system is designed to make user happy by just not asking and answering to user but by chatting about interesting topics which user might choose and like

Precondition:

User must select the topic she likes from the list of options

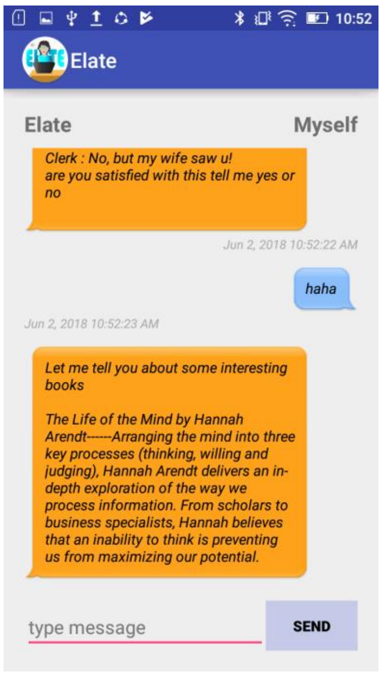
Test Steps:

1. User responds to ELATE’s reply
2. ELATE’s tries to extract sentiment of user
3. If sentiment of user are negative list of topics is shown to user
4. User select the topic she likes
5. Chat according to topic selected is initiated

Expected Result:

The chatter bot GUI will display chat which will distract user from negativity by talking on interesting topics like books, quotes, jokes, etc.

Obtained Output:


Elate Myself

Clerk : No, but my wife saw u!
are you satisfied with this tell me yes or no

Jun 2, 2018 10:52:22 AM

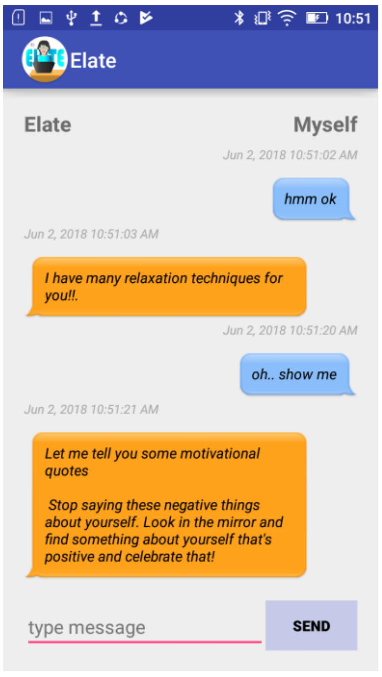
haha

Jun 2, 2018 10:52:23 AM

Let me tell you about some interesting books

The Life of the Mind by Hannah Arendt------Arranging the mind into three key processes (thinking, willing and judging), Hannah Arendt delivers an in-depth exploration of the way we process information. From scholars to business specialists, Hannah believes that an inability to think is preventing us from maximizing our potential.

type message SEND


Elate Myself

Jun 2, 2018 10:51:02 AM

hmm ok

Jun 2, 2018 10:51:03 AM

I have many relaxation techniques for you!!.

Jun 2, 2018 10:51:20 AM

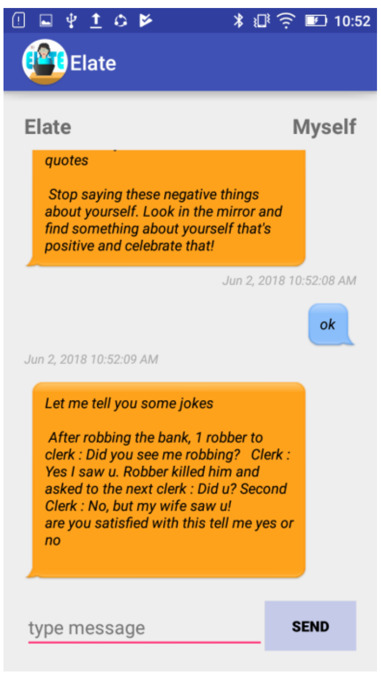
oh.. show me

Jun 2, 2018 10:51:21 AM

Let me tell you some motivational quotes

Stop saying these negative things about yourself. Look in the mirror and find something about yourself that's positive and celebrate that!

type message SEND


Elate Myself

quotes

Stop saying these negative things about yourself. Look in the mirror and find something about yourself that's positive and celebrate that!

Jun 2, 2018 10:52:08 AM

ok

Jun 2, 2018 10:52:09 AM

Let me tell you some jokes

After robbing the bank, 1 robber to clerk : Did you see me robbing? Clerk : Yes I saw u. Robber killed him and asked to the next clerk : Did u? Second Clerk : No, but my wife saw u!
are you satisfied with this tell me yes or no

type message SEND

Title:

Check if the bot is able to respond with an exit message if user feels to exit the chat window

Description:

The bot should be able to understand when an input of user if she is intending to exit the chat window

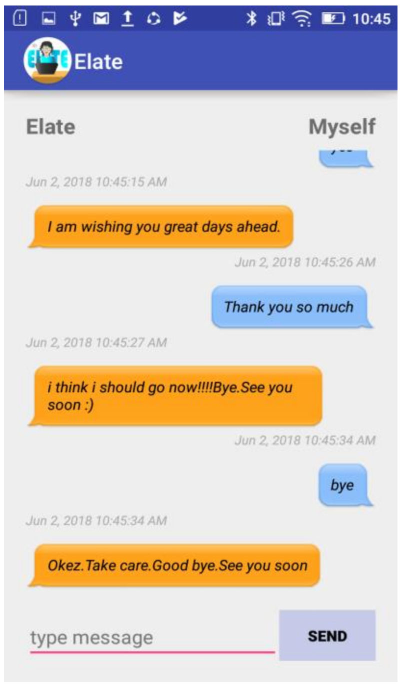
Test Steps:

1. User enters an exit message.
2. Bot analyses the input and responds with an exit message.

Expected Results:

The chat bot should respond with an exit message.

Obtained Output:



10:45

Elate

Elate | Myself

Jun 2, 2018 10:45:15 AM

I am wishing you great days ahead.

Jun 2, 2018 10:45:26 AM

Thank you so much

Jun 2, 2018 10:45:27 AM

i think i should go now!!!!Bye.See you soon :)

Jun 2, 2018 10:45:34 AM

bye

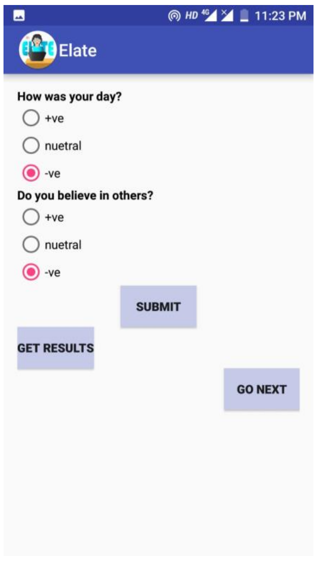
Jun 2, 2018 10:45:34 AM

Okez.Take care.Good bye.See you soon

Chapter 10

RESULTS

10.1 SCREEN SHOTS OF OUTPUT

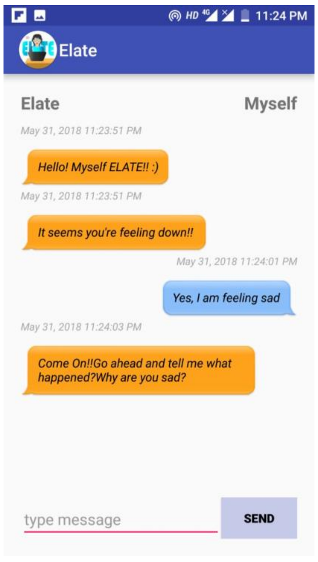


How was your day?

+ve nuetral [✓ -ve]

Do you believe in others?

+ve nuetral [✓ -ve]



Elate Myself

May 31, 2018 11:23:51 PM

Hello! Myself ELATE!! :)

May 31, 2018 11:23:51 PM

It seems you're feeling down!!

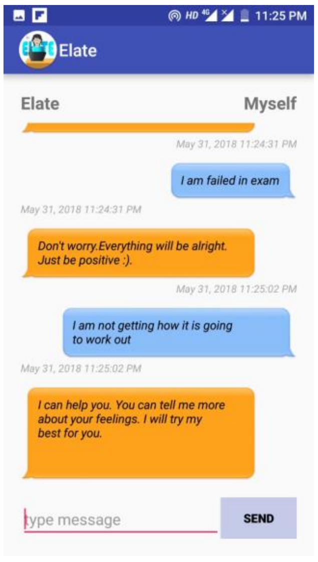
May 31, 2018 11:24:01 PM

Yes, I am feeling sad

May 31, 2018 11:24:03 PM

Come On!!Go ahead and tell me what happened?Why are you sad?

type message



Elate Myself

May 31, 2018 11:24:31 PM

I am failed in exam

May 31, 2018 11:24:31 PM

Don't worry.Everything will be alright. Just be positive :).

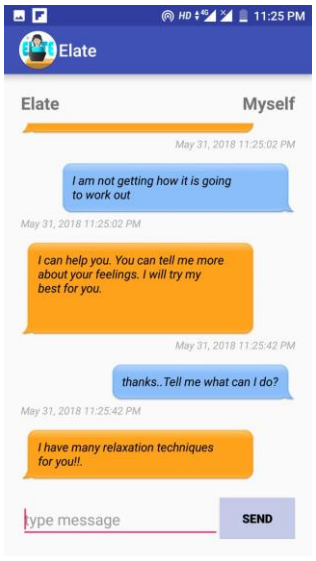
May 31, 2018 11:25:02 PM

I am not getting how it is going to work out

May 31, 2018 11:25:02 PM

I can help you. You can tell me more about your feelings. I will try my best for you.

type message



Elate Myself

May 31, 2018 11:25:02 PM

I am not getting how it is going to work out

May 31, 2018 11:25:02 PM

I can help you. You can tell me more about your feelings. I will try my best for you.

May 31, 2018 11:25:42 PM

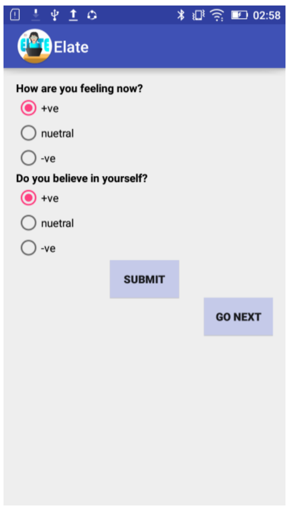
thanks..Tell me what can I do?

May 31, 2018 11:25:42 PM

I have many relaxation techniques for you!!.

type message

Positive chat initiation and conversation

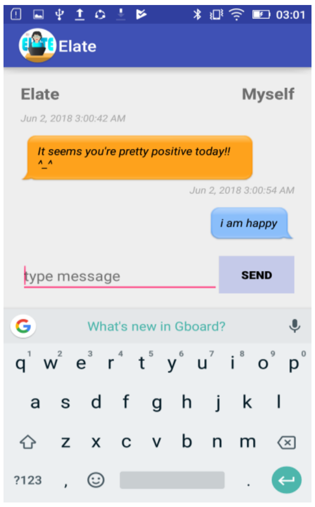

Elate

How are you feeling now?

[✓ +ve] nuetral -ve

Do you believe in yourself?

[✓ +ve] nuetral -ve

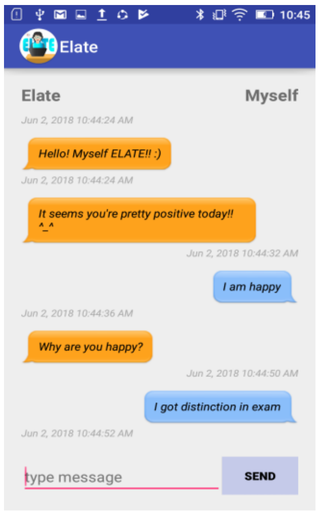

Elate | Myself

Jun 2, 2018 3:00:42 AM

It seems you're pretty positive today!! ^_^

Jun 2, 2018 3:00:54 AM

i am happy


Elate | Myself

Jun 2, 2018 10:44:24 AM

Hello! Myself ELATE!! :)

Jun 2, 2018 10:44:24 AM

It seems you're pretty positive today!! ^_^

Jun 2, 2018 10:44:32 AM

I am happy

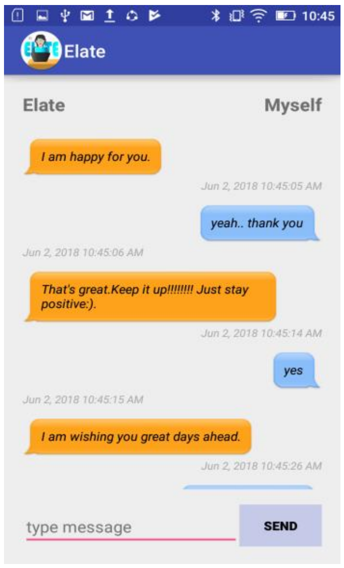
Jun 2, 2018 10:44:36 AM

Why are you happy?

Jun 2, 2018 10:44:50 AM

I got distinction in exam

Jun 2, 2018 10:44:52 AM


Elate | Myself

I am happy for you.

Jun 2, 2018 10:45:05 AM

yeah.. thank you

Jun 2, 2018 10:45:06 AM

That's great.Keep it up!!!!!!! Just stay positive:).

Jun 2, 2018 10:45:14 AM

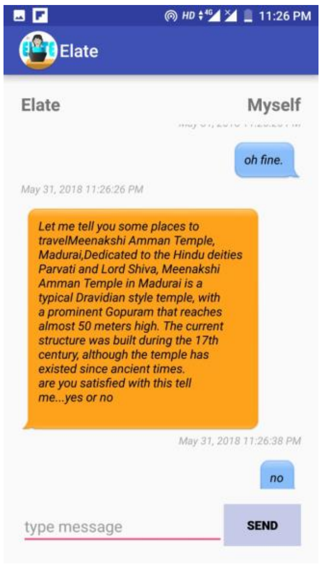
yes

Jun 2, 2018 10:45:15 AM

I am wishing you great days ahead.

Jun 2, 2018 10:45:26 AM

Negative chat initiation and conversation



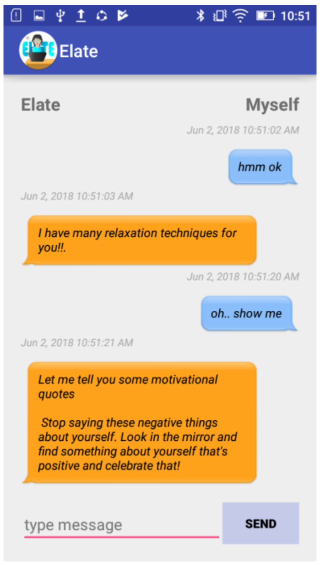
oh fine.

May 31, 2018 11:26:26 PM

Let me tell you some places to travelMeenakshi Amman Temple, Madurai,Dedicated to the Hindu deities Parvati and Lord Shiva, Meenakshi Amman Temple in Madurai is a typical Dravidian style temple, with a prominent Gopuram that reaches almost 50 meters high. The current structure was built during the 17th century, although the temple has existed since ancient times.
are you satisfied with this tell me...yes or no

May 31, 2018 11:26:38 PM

no



Jun 2, 2018 10:51:02 AM

hmm ok

Jun 2, 2018 10:51:03 AM

I have many relaxation techniques for you!!.

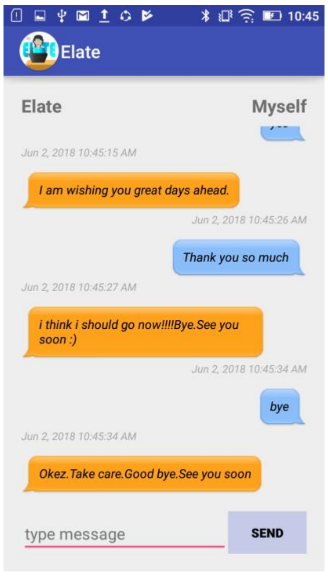
Jun 2, 2018 10:51:20 AM

oh.. show me

Jun 2, 2018 10:51:21 AM

Let me tell you some motivational quotes

Stop saying these negative things about yourself. Look in the mirror and find something about yourself that's positive and celebrate that!



Jun 2, 2018 10:45:15 AM

I am wishing you great days ahead.

Jun 2, 2018 10:45:26 AM

Thank you so much

Jun 2, 2018 10:45:27 AM

i think i should go now!!!!Bye.See you soon :)

Jun 2, 2018 10:45:34 AM

bye

Jun 2, 2018 10:45:34 AM

Okez.Take care.Good bye.See you soon

Converstaion on different topics and termination

Chapter 11

DEPLOYMENT AND MAINTENANCE

11.1 INSTALLATION AND UN-INSTALLATION

As the application is under development so it's not deployed on Android app store.

Once deployed user can install the app from play store and similarly can uninstall.

11.2 USER HELP

User need to have a mobile phone with Android Version 4.0.3(Ice cream sandwich) minimum.

Chat bot is designed for English language speakers and women from age group 18 to 24 only.

Chapter 12

CONCLUSION AND FUTURE SCOPE

In 21st century era, where atomization is posing as driving force for development. Likewise world is becoming more competitive and fast. In this competitive environment humans are losing their emotional hold of feelings. Unlike older time, where human-human interaction was easy, but in today's busy world not all individuals suffering from stress or any emotional turmoil can seek for help. In such case, chatter bot can prove fruitful to help individuals to feel positive and feel that they are cared and are being monitored. As psychology of men and women differs according to age group, we decided to design a chatter bot which will focus on specific age group and gender. As our project group comprise of all female members it was convenient for us to understand the psychology of women from age group 18 to 24 and thereby design a suitable chat system. Our chatter bot application is still under research and we aim at making the chat system as human like as possible .In our future scope with intend to deploy this application on Android Play store which will be available for free to use. Thus, we can conclude that motivating chatter bots can help individuals to seek for emotional advice whenever they need to, as chatter bots are always available.

ANNEXURE A

REFERENCES

[1]. Prof.Nikita Hatwar, Ashwini Patil , Diksha Gondane, " AI BASED CHATBOT", International Journal of Emerging Trends in Engineering and Basic Sciences (IJEEBS), Volume 3, Issue 2 (March-April 2016), PP.85-87

[2]. JOSEPH WEIZENBAUM, "ELIZA A Computer Program For the Study of Natural Language Communication Between Man And Machine", Massachussets Institute of Technology, Cambridge, Mass., volume 9 / Number / / January., 1966

[3]. John Zakos,Liesl Capper, "CLIVE – An Artificially Intelligent Chat Robot for Conversational

[4]. Language Practice", Griffith University, Gold Coast, Australia, SETN 2008, LNAI 5138, pp. 437–442, 2008

[5]. AlbertGattand,EhudReiter, " SimpleNLG : A realization engine for practical applications", Department of Computing Science University of Aberdeen Aberdeen AB24 3UE, UK, Proceedings of the 12th European Workshop on Natural Language Generation, pages 90–93, Athens, Greece, 30 – 31 March 2009.

[6]. Bayan AbuShawar, Eric Atwell, " Automatic Extraction of Chat bot Training Data from Natural Dialogue Corpora", IT department; School of Computing Arab Open University; University of Leeds Amman, Jordan; leeds, Uk

[7]. Anirudh Khanna, Bishwajeet Pandey, Kushagra Vashishta, Kartik Kalia, Bhale Pradeepkumar and Teerath Das, " A Study of Today's A.I. through Chatbots and Rediscovery of Machine Intelligence", Chitkara University, Punjab, India, International Journal of u- and e- Service, Science and Technology Vol.8, No. 7 (2015), pp.277-284

[8]. Nicole Radziwill and Morgan Benton, " Evaluating Quality of Chat bots and Intelligent Conversational Agents" ,2016

[9]. M. A. Mioc, " Client Server Chat Application", University Stefan cel Mare Suceava Integrated Center for research, development and innovation in Advanced Materials, Nanotechnologies, and Distributed Systems – MANSiD Suceava, Romania, Journal of Multidisciplinary Engineering Science and Technology (JMEST) ISSN: 2458-9403 Vol. 3 Issue 7, July - 2016

[10]. Revati R. Dudhatra, Dr. Yogesh A Jogsan, " Mental Health and Depression among Working and NonWorking Women", Department of Psychology, Saurashtra University, Rajkot, India, International Journal of Scientific and Research Publications, Volume 2, Issue 8, August 2012

[11]. Holger Stenzhorn, "XtraGen — A Natural Language Generation System Using XML- and Java-Technologies", XtraMind Technologies GmbH Stuhlsatzenhausweg 3 66123 Saarbr¨ucken, Germany, October 2002

[12] Sameena Thabassum, “Sentiment Analysis on Interactive Conversational Agent/Chatbots”, User. International Journal of Computer Applications 120(1):21-24, June 2015.

[13] A. TURING, “I.—COMPUTING MACHINERY AND INTELLIGENCE”, *Mind*, vol., no. 236, pp. 433-460, 1950.

ANNEXURE B

LABORATORY ASSIGNMENTS ON PROJECT ANALYSIS OF ALGORITHMIC DESIGN

A.1 IDEA Matrix

B.1.1 Problem Statement

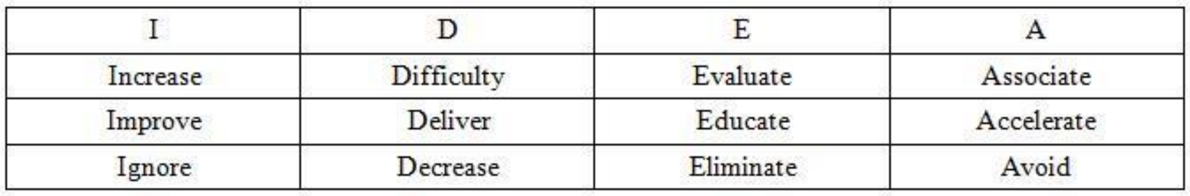
To develop the problem under consideration and justify feasibility using concepts of knowledge canvas and IDEA Matrix.

B.1.2 Project Problem Statement

Create an intelligent therapist chatter bot mobile application which will converse with user by first analysing the sentiment or mood of the user and thereby initializing a positive, negative or neutral chat format to motivate and elate the user.

B.1.3 IDEA Matrix

The concept of idea matrix emerges from study of various ideas, parameters and feasibility study of a particular problem statement. It consists of keywords on the basis them a problem statement should be developed.



I	D	E	A
Increase	Difficulty	Evaluate	Associate
Improve	Deliver	Educate	Accelerate
Ignore	Decrease	Eliminate	Avoid

B.1.4 I

B.1.4.1 Increase

There would be a significant increase in performance of system as it replaces static 2D graphic simulation system by 3D simulators. It allows the user to analyze the data by putting different manual values increases accuracy rate than static simulators.

B.1.4.2 Improve

Current methods do not give a clear and accurate picture of how the rocket works, as they ignore the fact that various other parameters such as forces of environment and fuel are also acting on it. Furthermore, they do not give any idea about what happens during the different stages of rocket launch. 3D simulator simulates the state and trajectory of satellite once it is deployed in orbit. Thus improves accuracy as well as other parameters like knowledge about fuel and environmental forces acting on it.

B.1.4.3 Ignore

Simulator will be developed considering the demand of 3D simulator which works by considering various new parameters. Other simulators may have different modules and flow. The user should ignore the differences in design, flow and their working conditions. System is robust enough to fault tolerance.

B.1.5 D

B.1.5.1 Difficult

In this model, only the difficulty is it's platform dependence. The new 3D model is only for the Windows Operating System. This may be resolved in near future.

B.1.5.2 Deliver

This model will help to deliver high performance interface to analyze the different cases by manually putting values of various types of forces acting on it.

B.1.5.3 Decrease

Tool will be able to decrease the inaccurate and unclear picture of how the rocket works. It will also decrease faults and danger related to fuel and environmental forces.

B.1.6 E

B.1.6.1 Evaluate

Analysing various parameters manually with different perspectives and evaluate which parameters are best for the rocket launching. Thus efficiency can be evaluated by modeling the 3D simulator using Unity3D and blender techniques.

B.1.6.2 Educate

End user expects to learn how thrust; time of thrust and mass of a rocket can be changed to make a rocket go as high as possible. This model helps students, teachers and researchers to learn the simulation of rocket launching in detail.

B.1.6.3 Eliminate

Currently, considering the requirements of user, we are going to eliminate the unnecessary information and extract only the data which will help user to learn how speed and forces acting on changing environment during the launching process.

B.1.7.1 Associate

Many users can get associated with the simulator. As simulator model is much easy to under-stand and operate.

B.1.7.2 Accelerate

The acceleration would be using GPU to increase parallelism in graphic rendering as tasks can be divided on different cores.

B.1.7.3 Avoid

Considering the technical feasibility of the project, platform dependency can be avoided so that the implemented component will be able to run on only windows system.

IDEA	**PROCESS**	**PARAMETERS**
Increase	Increase the accuracy of prediction	Intent
Improve	Chat intelligence	Strong NL generation
Ignore	Wrong sentiment prediction	
Deliver	Forecasting values, quick predication	Power
Decrease	Redundancy	Large dataset and NLG
Educate	Educate Project members and users	References & papers
Evaluate	Accuracy of algorithm	Correct sentiment analysis & NLG
Accelerate	Training time	Reach sentiment corpus

B.2 NP Analysis

B.2.1 Problem Statement

Perform feasibility assessment of Project Problem statement using NP-Hard, NP- Complete or satisfiability issues using modern algebra and/or relevant mathematical models.

B.2.2 Project Problem Statement

Create an intelligent therapist chatter bot mobile application which will converse with user by first analysing the sentiment or mood of the user and thereby initializing a positive, negative or neutral chat format to motivate and elate the user.

B.2.3 Objective

Understand the scope and feasibility of the project problem statement. Determine if the problem is NP-hard, NP- Complete or satisfiability issues using modern algebra and/or relevant mathematical models.

B.2.4 Basic Concepts

B.2.4.1 P - Polynomial Time Solving

Problems which can be solved in polynomial time, which take time like O(n), O(n2), O(n3).

Example:

1. Finding maximum element in an array or to check whether a string is palindrome or not. So there are many algorithms for this problem which can be solved in polynomial time.
2. Say you have an algorithm that finds the smallest integer in an array. One way to do this is by iterating over all the integers of the array and keeping track of the smallest number you've seen up to that point. Every time you look at an element, you compare it to the current minimum, and if it's smaller, you update the minimum.

B.2.4.2 NP - Non deterministic Polynomial time solving

Problem which can't be solved in polynomial time like TSP (traveling salesman problem) or an easy example of this is subset sum: given a set of numbers, does there exists a subset whose sum is zero?

B.2.4.3 Reducibility

Take two problems A and B, and both are NP problems. If we can convert one instance of problem A into problem B (NP-problem) then it means that A is reducible to B.

B.2.4.4 NP Hard

What does NP-hard mean? A lot of times you can solve a problem by reducing it to a different problem. We can reduce Problem B to Problem A if, given a solution to Problem A, we can easily construct a solution to Problem B. (In this case, easily means "in polynomial time".) If a problem is NP-hard, this means we can reduce any problem in NP to that problem. This means if we can solve that problem, we can easily solve any problem in NP. If we could solve an NP-hard problem in polynomial time, this would prove P = NP. Now suppose we found that A is reducible to B, then it means that B is at least as hard as A.

B.2.4.5 NP-complete

A problem is NP-complete if the problem is both NP-hard and in NP. The group of problems which are both in NP and NP-hard are known as NP-Complete problem.

B.2.5 Problem Analysis

1. User has to apply different manoeuvre to successfully land the rocket. This can either fail or success.

Thus it is a decision problem.

2. It is subjective to user i.e. it varies from user to user.
3. User has a choice to select any service at random and this cannot be determined in polynomial time.
4. Although, some instances like calculating the position, orientation, force, velocity and status and displaying them to user can be done in polynomial time.
5. To solve the given problem, it can be reduced to a problem like Circuit Satisfiability problem where we decide whether a given Boolean circuit has as assignment of its inputs that makes the output true.

The same is applicable to our problem as the user decides which manoeuvre to use from the set of different manoeuvre which exists so as to make the output true i.e. successful landing.

B.2.6 Conclusion

Since, our problem is NP type and is also reducible to already proved NP-Complete problem i.e. C-SAT problem, thus we conclude that our problem is NP-Complete Problem.

ANNEXURE C

LABORATORY ASSIGNMENTS ON PROJECT QUALITY AND RELIABILITY TESTING OF PROJECT DESIGN

LABORATORY ASSIGNMENTS ON PROJECT QUALITY AND RELIABILITY TESTING OF PROJECT DESIGN

C.1 Testing

C.1.1 Problem Statement

Apply testing methodology for the project problem statement using generated test data selection and appropriate use of testing tools.

C.1.2 Basic Concepts

C.1.2.1 Unit Testing

This is the most basic testing mechanism at the developer level. This covers very narrow and well defined scope. We isolate the code from any outside interaction or any dependency on any module. Unit tests focus on very small unit of functionality. They cover the interaction of the code with memory only and do not cover any interaction with network, database or file systems. These dependencies are hard coded into the code while testing.

They provide a simple way to check smallest units of code and prove that units can work perfectly in isolation. However, we need to check further that when these units are combined they work in a cohesive manner which leads us to further types of tests.

C.1.2.2 Regression Testing

Regression testing focuses on finding defects after a major code change has occurred.

Specifically, it seeks to uncover software regressions, as degraded or lost features, including old bugs that have come back. Such regressions occur whenever software functionality that was previously working, correctly, stops working as intended.

Typically, regressions occur as an unintended consequence of program changes, when the newly developed part of the software collides with the previously existing code. Common methods of regression testing include re- running previous sets of test-cases and checking whether previously fixed faults have re-emerged. The depth of testing depends on the phase in the release process and the risk of the added features. They can either be complete, for changes added late in the release or deemed to be risky, or be very shallow, consisting of positive tests on each feature, if the changes are early in the release or deemed to be of low risk. Regression testing is typically the largest test effort in commercial software development, due to checking numerous details in prior software features, and even new software can be developed while using some old test-cases to test parts of the new design to ensure prior functionality is still supported.

C.1.2.3 Performance Testing

Performance testing is generally executed to determine how a system or sub-system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate, measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage.

C.1.2.4 Acceptance Tests

These form the final level of test plan. Every feature is checked from the user's perspective by the testing team and the feature is marked accepted or rejected. These tests test the application from end user's perspective like detecting crashes after going through a certain flow. The results of these tests are very subjective in nature and it takes a while to figure out the exact issue in the code.

C.1.3 Test Cases

C.1.3.1 Case 1: Installation Testing

As the application is under development so it's not deployed on Android app store.
Once deployed user can install the app from play store and similarly can uninstall.
For now, system requirements are as follows:

- User need to have a mobile phone with Android Version 4.0.3(Ice cream sandwich) minimum.
- Chat bot is designed for English language
- Application is available for women from age group 18 to 24 only.

C.1.3.2 Case 2: User Interface Testing

Our primary aim is to create a user friendly interface which will consists of user's side chat box and ELATE's side chat box. UI is simple to access and use. Font and colours of chat application are decided to make user more intuitive and friendly with application.

Positive test case: If user answers the questionnaire honestly, user will be presented with proper chat format.

Negative test case: If the user didn't answer the questionnaire honestly and correctly user will be presented with some non-contextual chat which might annoy user.

C.1.3.3 Case 3: Performance Testing

Our primary aim here is to reduce the lag and have a synchronous real time chat. We achieved this by adopting client server architecture.

All the chat related data, program, libraries are stored at server side so as to minimize the load at client side.

As chat related data is stored at server, size of application is considerably reduced and user side overhead of memory and storage is reduced as well.

Thus application is highly responsive and performs efficiently.

C.1.3.4 Case 4: Functionality Testing

User is conversed with chat format which depends on the mood of user.

We have limited our mood analysis to 3 categories and have designed a chat for each category. User is asked set of questions which enables system to analyse the sentiment of user.

If the questionnaire is answered correctly and honestly by user, system initialises a chat conversation based on the sentiment analysed.

System is also understands the input of user conveying an exit message.

System successfully portrays an exit message terminating the chat.

Expected Result: ELATE will detect the correct sentiment of user and will initiate a chat base on the derived sentiments .ELATE will eventually help users feeling negative or sad and make their mood positive and happy.

Fail Result: ELATE might derive wrong sentiment and might covey non-contextual chat thereby annoying the user.

Test Number	Test Scenario	Action	Expected Results	Status(Fail/Pass)
1.	Installation Testing	Install ELATE on system meeting the requirements.	Should install properly and does not require reboot.	System still under development
2	Performance Testing	Starting the application.	Should start and load the application without lags.	Passed

3	User interface testing	Typing in the input box and obtaining output on dialog box.	All Elements of GUI should work.	Passed
4	Functionality Testing	Add null input	Should require the user to enter some input.	Passed.
5	Functionality	Checking for repetitive consecutive statements.	Should give a generic response if user enters the same response back to back.	Passed
6	Functionality testing	Checking the sentiment extraction from questionnaire	Should be able to extract the use's sentiment and thereby initialize chat based on sentiment	Passed
7	Functionality testing	Checking weather chatter bot is able to understand user intention to terminate the chat	Should be able to appropriate exit message.	Passed

ANNEXURE D

REVIEWERS COMMENTS OF PAPER SUBMITTED

REVIEWERS COMMENTS OF PAPER SUBMITTED

1. Paper Title: ELATE : An Intelligent Therapist Chatter Bot
2. Name of the Conference/Journal where paper submitted :Vishwakon Paper Presentation conference
3. Paper accepted/rejected : Accepted
4. Review comments by reviewer : To improve the chat system
5. Corrective actions if any: Chat system was re-designed to meet the expected standards.

ANNEXURE E
TERM-II PROJECT LABORATORY

ASSIGNMENTS

1. Review of design and necessary corrective actions taking into consideration the feedback report of Term I assessment, and other competitions/conferences participated like IIT, Central Universities, University Conferences or equivalent centres of excellence etc.

To improve the chat system design. Chat system was re-designed to meet the expected standards and was constricted to specific age group and gender. This enabled to achieve feasibility and convenient designing of chat system.

2. Project workstation selection, installations along with setup and installation report preparations.

Project requirements:

- Java IDE Eclipse
- Programming Language : Java
- Intellij IDE
- Files directory structure
- Android studio
- Retrofit library
- OpenNLP library
- SimpleNLG library

3. Programming of the project functions, interfaces and GUI (if any) as per 1 st Term term-work submission using corrective actions recommended in Term-I assessment of Term-work.

In TERM 1 chat application was developed entirely in android studio which added an overhead of IDE lag because of large libraries and dataset. Memory storage handling and maintenance was responsible for low performance of application.

Corrective measure: To make chat application synchronous client server architecture was adopted. This way all the chat related data was stored at server side. Only client side code and GUI related was stored over application which enabled us to overcome memory and performance related issue.

4. Test tool selection and testing of various test cases for the project performed and generate various testing result charts, graphs etc. including reliability testing.

Manual Testing is performed.

5. Installations and Reliability Testing Reports at the client end.

Project not yet deployed.

ANNEXURE F
INFORMATION OF PROJECT GROUP MEMBERS

1. Name :Akanksha Patil
2. Exam Seat No : B120294334
3. Date of Birth : 20/02/1996
4. Gender : Female
5. Permanent Address: Flat no. B-6,”Laxman Residency”, Morwadi, Pimpri Chinchwad, Pune
6. Paper Published :-Vishwakon Paper presentation
7. E-Mail : akankshapatil002@gmail.com
8. Mobile/Contact No. : 9403386097
9. Placement Details :Placed at Finiq

1. Name : Kshitija Pandit
2. Exam Seat No : B120394332
3. Date of Birth : 29/01/1997
4. Gender : Female
5. Permanent Address: A-7/15,Indira Nagar Lower,Bibvewadi,Pune-37
6. Paper Published :-Vishwacon Paper presentation
7. E-Mail : kshitijapandit01@gmail.com
8. Mobile/Contact No. : 9168703821
9. Placement Details :

1. Name :Aishwarya Nerlekar
2. Exam Seat No : B120394326
3. Date of Birth : 13/10/1996
4. Gender : Female
5. Paper published : Vishwacon Paper Publication
6. Permanent Address: E-2/304, Shivsagar residency, opp.Sun orbit ,Suncity road,off Sinhagad road,Pune-411051
7. Paper Published :-Vishwakarma Paper presentation
8. E-Mail : aishwaryanerlekar96@gmail.com
9. Mobile/Contact No. : 9823712488
10. Placement Details :

1. Name : Anuja Tatpuje
2. Exam Seat No : B120394398
3. Date of Birth : 22/03/1997
4. Gender : Female
5. Permanent Address : F-6 , Yashashri apartment,80,A,Yadogopal Peth,Samarth Mandir,Satara-415002
6. Paper published : Vishwacon Paper Publication
7. E-Mail : tatpuje.anu@gmail.com
8. Mobile/Contact No. :840792316
9. Placement Details : Placed at Nice