SMART SENTENCE GENERATOR

Submitted in partial fulfilment of the requirements of the degree of

Bachelor of Engineering

by

Navkar Nimesh Shah – 60003160055
Bhavi Hitesh Modi – 60003178007
Yash Nagesh Swami Adke - 60003178012

Project Guides:

Dr. Abhijit R. Joshi
Mrs. Lakshmi D. Kurup

(Information Technology)

Dwarkadas J. Sanghvi College of Engineering University of Mumbai

2019-2020

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

CERTIFICATE

This is to certify that the project entitled **"Smart Sentence Generator"** is a bonafide work of "**Navkar Nimesh Shah, Bhavi Hitesh Modi and Yash Nagesh Swami Adke" (60003160055, 60003178007, 60003178012)** submitted to the University of Mumbai in partial fulfilment of the requirement for the award of the degree of **"Bachelor of Engineering"** in **"InformationTechnology"**.

Dr. Abhijit R. Joshi
Guide

Mrs. Lakshmi D. Kurup
Guide

Dr. VinayaSawant
Head of Department

Dr. Hari Vasudevan
Principal

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Project Report Approval for B. E.

This project report entitled (***Smart Sentence Generator***) by (***Navkar Nimesh Shah, Bhavi Hitesh Modi and Yash Nagesh Swami Adke***) is approved for the degree of Information Technology.

Examiners: 1.

2.

Date:

Place:

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Navkar Nimesh Shah – 60003160055

Bhavi Hitesh Modi - 60003178007

Yash Nagesh Swami Adke - 60003178012

Acknowledgement

"It is not possible to prepare a project report without the assistance and encouragement of other people. This one is certainly no exception."

On the very outset of this report, we would like to extend our sincere gratitude to all the people who have helped us in this endeavour. Without their active guidance and support this project would not be possible.

We sincerely thank **Dr. Hari Vasudevan**, principal, Dwarakadas J. Sanghvi College of Engineering and **Dr. Vinaya Sawant**, Head of Department, Information Technology for their constant support for our project.

We are extremely thankful to **Mrs. Neha Katre** and **Dr. Abhijit R. Joshi** for introducing us to **Mrs. Lakshmi Kurup**, for her constant supervision regarding the project and for providing all the necessary information related to Sentence Generation and its algorithms. We are extremely thankful to her for sharing the required resources with us and guiding us whenever needed.

We would like to express our special thanks to our project guide**, Dr. Abhijit R. Joshi** and **Mrs. Lakshmi D. Kurup** for their valuable guidance and help throughout the course of this project. Their expertise in this field helped us with our project work. Without their active guidance, help, cooperation and encouragement we would not have made progress in the project. We are ineffably indebted to them for all his support to accomplish this project report.

We would also like to show our gratitude to our friends and family who have always supported us morally and helped us directly or indirectly.

Navkar Nimesh Shah,

Bhavi Hitesh Modi,

Yash Nagesh Swami Adke.

Abstract

We have developed a Smart Sentence Generator for the English language wherein sentence templates are used for sentence generation. The sentence templates, which form the basic structure of the sentences ensures that the generated sentences are syntactically correct as well as meaningful. Currently, system caters to different types of sentences such as simple, complex and compound along with interrogative and exclamatory. We have used Natural Language Processing along with Natural Language Toolkit, WordNet library, NodeBox and a wide range of self-developed classes to pick out words for various parts of the sentence according to context and relevance. One can extend this sentence generator for developing an entire working system for an essay generator, script writing, and legal documents or in advertising, etc.

Content

1.	Introduction	1
1.1	Motivation/Objective	2
1.2	Major Challenges	2
1.3	Report Overview	3
2.	Literature Review	4
2.1	Existing Work	5
2.1.1	Literature Related to Existing Systems	5
2.1.2	Literature Related to Methodology/Approaches	6
2.1.3	Literature Related to Algorithms	7
2.1.4	Literature Related to Technology/Tools/Framework	8
2.2	Observations on Existing Work	9
3.	Proposed Methodology/Approach	10
3.1	Problem Definition	11
3.2	Scope	11
3.2.1	Assumptions and Constraints	11
3.3	Proposed System Approach	12
3.3.1	Features of Proposed System	12
4.	Project Management	14
4.1	Project Schedule	15
4.1.1	Task Network Diagram	15
4.1.2	Timeline Chart	16
4.2	Feasibility Study	17
4.2.1	Technical Feasibility	17
4.2.2	Operational Feasibility	17
4.2.3	Economical Feasibility	18
4.3	Project Resources	18
4.3.1	Hardware Requirements	18
4.3.2	Software Requirements	18
4.4	Project Estimation	19
4.4.1	COCOMO Estimation Model	19
4.4.2	Function Point Analysis	19
4.5	Risk Management Mitigation Planning	20
5.	System Design	22

5.1	Design Diagrams	22
5.1.1	UML Diagrams	22
5.2	System Architecture	23
5.3	Algorithm	25
6.	Implementation	27
6.1	Working of System	27
6.2	Templates	28
6.3	Output	34
6.4	Tools and Technologies Used	43
7	Testing and Result	45
7.1	Test Plan	45
7.2	Test Cases	46
7.3	Testing methods used	48
7.4	Experimental Results and Conclusions	49
8.	Conclusion	51
9.	Future Scope	53
10.	Publications	55
	References	56

List of Figures

Figure No.	Figure Name	Page No.
1	Comparison between various models.	9
2	Task Network Diagram	15
3	Timeline chart	16
4	Use Case Diagram	22
5	System Architecture	23
6	Working of the System	27
7	Homepage	34
8	Types of Sentences	34
9	Simple Sentence Variations	34
10	Simple Sentence Classes	35
11	Normal Simple Sentence Output	35
12	Multiple Subjects Simple Sentence Output	35
13	Multiple Objects Simple Sentence Output	36
14	Interrogative Sentence Variations	36
15	Interrogative Sentence Classes	36
16	Normal Interrogative Sentence Output	37
17	Multiple Subjects Interrogative Sentence Output	37
18	Multiple Objects Interrogative Sentence Output	37
19	Exclamatory Sentence Variations	38
20	Exclamatory Sentence Classes for How	38
21	Exclamatory Sentence Output for How	38
22	Exclamatory Sentence Classes for Order	39
23	Exclamatory Sentence Output for Order	39
24	Exclamatory Sentence Output for Such	39
25	Exclamatory Sentence Output for What	40
26	Exclamatory Sentence Classes for Wish	40
27	Exclamatory Sentence Output for Wish - Singular	40
28	Exclamatory Sentence Output for Wish - Plural	41
29	Compound Sentence Classes	41
30	Compound Sentence Output	41
31	Complex Sentence Classes	42
32	Complex Sentence Output for Although	42

33	Complex Sentence Output for Whenever	42

List of Tables

Table No.	Table Name	Page No.
1	Risk Mitigation and Management	20
2	Test Cases	46

CHAPTER 1

INTRODUCTION

1. Introduction:

As constant learners, we all have to face a point in our academics where we find it difficult to pace ourselves with the depth and complexity of the English language. Not each and every student is able to keep up with the academic teaching carried out in the institutions. The lack of existence of a complete system that can help students is the need for today's generation. Thus, as a part of Intelligent Tutoring System (ITS) we aim to develop a system that will help the students to form sentences in English based on the subject, verb and object. These sentences will be semantically as well as syntactically correct. The development of a system would make it easier for the students to work on their vocabulary. The generation of different kinds of sentences from the provided words will help the students to learn about the variations that can be used in turn to improve their knowledge.

1.1. Motivation/Objective:

English as a language has been proved to be difficult for the students to master. The fear of the language can be tackled by a system that helps the students to use English language flawlessly. Moreover, teaching in the institutions may sometimes prove to be ineffective in guiding the students to overcome their weaknesses. The students may sometimes not be able to understand in class and due to fear of asking doubts they get stuck. This is where a system can be helpful in the understanding of the student. Teaching English is a tedious task. Also, there are currently no existing systems that can help the students through the learning process of the language. The available online assessment and guidance only provide objective answers, which is not quite helpful.

Moreover, the development of a system that can be used for teaching the students requires a perfect accuracy rate and completeness of the sentence. Currently, existing systems do not have accuracy rates even close to what can be suitable for purpose of teaching and learning.

So, we are proposing a system that generates sentences with more accuracy than the existing systems.

1.2. Major Challenges:

The major challenges are:

1. Generating sentences which are semantically correct:

There exist systems which provide semantically correct sentences, but the accuracy rate of the generated sentences is very low.

2. Generating sentences with proper vocabulary (syntactically correct sentences):

Not only the language, but also the vocabulary and the grammatical errors are the major concerns while developing a sentence generation system.

3. Completeness of the generated sentence:

The generated sentence has no meaning if the sentence is incomplete. Incomplete sentences may sometimes also provide a different meaning altogether. Therefore, the completeness of the generated sentence is necessary to avoid ambiguities.

4. The sentences generated should be sensible:

There may arise a condition in which one of the keyword is say orange. So, now the system may get confused in comprehending the meaning of this word because it has two meanings. Orange can be a color as well as a fruit and the system can make an error while generating a sentence. For eg. the sentence generated may be, “Orange is my favorite color. It tastes sweet.”
In the development of the proposed system, the above mentioned issues are decided to address.

1.3. Report Overview:

This report focuses on sentence generation. Chapter one introduces the objective of the project and also the major challenges to be faced while implementation of the system. Chapter two is about the existing systems, the literature review and different methodologies and tools that already exist in the previous systems. Chapter three lets the user know in depth about the problem definition, the major features as well as benefits of the sentence generation system. Chapter four describes the Project Management. It also highlights the feasibility study as well as hardware and software requirements along with project cost estimation as per the COCOMO Model and function point analysis. Finally it describes the Risk Mitigation Management Plan. Chapter five represents the system design which includes various diagrams such as Use Case Diagram, Activity Diagram and the System Architecture. It also gives a brief introduction about the proposed system algorithm. Chapter six covers the implementation part. In this chapter, the actual working of the system is explained in detail. Examples of the dataset and the code with output are provided to get a gist of the system. Chapter eight concludes the entire report and explains why the proposed system is better than the existing system. Chapter nine discusses about the future scope of this system. It talks about how the system can be put to better use by making some modifications or integrating multiple projects to develop an ITS. Finally, the report closes with chapter ten which is the research paper for the very same project.

CHAPTER 2

LITERATURE REVIEW

2. Literature Review:

A literature review surveys scholarly articles, books and other sources (e.g. dissertations, conference proceedings) relevant to a particular issue, area of research, or theory, providing a description, summary, and critical evaluation of each work. The purpose is to offer an overview of significant literature published on a topic. The literature review is a very crucial part for any project implementation. Before undertaking any major project, it is important to perform literature review of the existing works related to the project.

2.1. Existing Work:

At present there is no fully functional system that can be deployed for sentence generation which can be deemed reliable. But there have been researches and partial implementation of similar systems that can be improved on to build a model for English Sentence Generation.

There are numerous different approaches for classifying the text based on the keywords. The Harvard NLG E2E challenge is based on the generation of a neural template using Recurrent Neural Networks. The data used for training is already preprocessed. The use of Bi-directional RNN with attention and dropout makes training of data efficient. However the system is not designed for generation of sentences in English language. Moreover the model is under development phase. The output sentences are generally incomplete or have grammatical errors.

Text Generation using Bidirectional LSTM and Doc2Vec models - David Campion. This sentence generator model is although incomplete but gives a very promising method of sentence generation from the key words. He uses Bidirectional LSTM model for a more accurate sentence formation. The prediction of the next word in the sentence becomes more accurate. However the reliability, semantics and grammar of the sentence cannot be predicted too enough.

2.1.1. Literature Related to Existing Systems:

Actually, there is a lot of literature about text generation using “AI” techniques, and some codes are available to generate texts from existing novels, trying to create new chapters for great success like ‘Game of Thrones’, ‘Harry Potter’, or a complete new theatre scene in the style of ‘Shakespeare’. Sometimes they give interesting results. However, generated texts have a taste of achievement.

Generated sentences do seem to be quite right, with correct grammar and syntax, as if the neural network understood correctly the structure of a sentence. But sometimes, generates text that is insensible.

2.1.2. Literature Related to Methodology/Approaches:

The task at hand is generating sentences that are meaningful and correct. Overtime there have been a raft number of approaches undertaken that enable machines create such meaningful sentences. Some of the methodologies include: Annotation Based, CFG(Context Free Grammar) Based, Example Based.

1. Annotation based sentence:

Annotation based sentence generation is the method of using Images to generate sentences. We humans can draw precise descriptions of pictures, while focusing on the important aspects and neglecting the unimportant matter. Thus most of the descriptions drawn by the humans for an image are a legitimate data source for the machine to learn. Thus, these systems are designed to write down sentences from the objects and their relations identified by the model. The meaning of a sentence derived from human knowledge is given a score. The system is given a score based on the meaning obtained from the sentence it generated and both scores are then compared to get an estimate of the accuracy of system generated sentences.

2. Context free grammar based systems:

Context free grammar based systems are designed to in such a way that they generate sentences that are required to cover certain grammar criteria and meet the constraints demanded by the systems such as parser/compiler, validating grammar, NLP, etc. There are certain predefined rules that are to be followed while generation of sentences, but the rules and criteria defined must be too specific and precise to avoid sentences that are either too complex in structure or too simple. The algorithms are based on grammar coverage criteria that include rule coverage and context dependent rule coverage.

3. Example based sentence:

Example based sentence generation model uses The Hidden Markov Model for sentence reduction technique for text generation. In sentence reduction method the original sentence is reduced to create a new sentence which is simpler in structure and gives a gist of the original sentence. A number of models have been developed that summarize the given text into small and simpler sentences. The template translation learning algorithm that is used for sentence reduction uses examples of long sentences and their reduced forms to generate template rules automatically i.e. the sentences are generated using those rules and keywords.

4. FIB based sentence:

FIB based sentence completion uses a combination of a unique application of word occurrence likelihood and the Google n-grams corpus to select words having strong contextual linking with their surrounding words. This method may prove to be effective in one liners or isolated fill in the blanks, However it is a accuracy takes a hit in text completion paragraphs.

2.1.3. Literature Related to Language Models:

1. Recurrent Neural Network(RNN) :

Recurrent Neural Network (RNN) are a type of Neural Network where the output from previous step are fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other, but in cases like when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. Thus RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is Hidden state, which remembers some information about a sequence. An RNN remembers each and every information through time. It is useful in time series prediction only because of the feature to remember previous inputs as well. This is called Long Short Term Memory (LSTM).

2. GloVe:

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

3. GPT-2:

GPT-2 is a successor of GPT, the original NLP framework by OpenAI. The full GPT-2 model has 1.5 billion parameters, which is almost 10 times the parameters of GPT. GPT-2 give State-of-the Art results as you might have surmised already. The architecture of GPT-2 is based on the very famous Transformers concept. At each step, the model consumes the previously generated symbols as additional input when generating the next output[3].

4. BERT (Bidirectional Encoder Representations from Transformers):

BERT (Bidirectional Encoder Representations from Transformers) is a recent paper published by researchers at Google AI Language. The results show that a language model which is bidirectionally trained can have a deeper sense of language context and flow than single-direction language models.

The researchers detail a novel technique named Masked LM (MLM) which allows bidirectional training in models in which it was previously impossible.

5. XLNet:

XLNet is a BERT-like model instead of a totally different one. But it is a very promising and potential one. In one word, XLNet is a generalized autoregressive(AR) pretraining method. AR language model is a kind of model that using the context word to predict the next word. But here the context word is constrained to two directions, either forward or backward.

6. Microsoft's MT-DNN:

The model "not only leverages large amounts of cross-task data, but also benefits from a regularization effect that leads to more general representations to help adapt to new tasks and domains." MT-DNN builds on a model Microsoft proposed in 2015 and integrates the network architecture of BERT, a pre-trained bidirectional transformer language model proposed by Google last year.

2.1.4. Literature Related to Technology/Tools/Framework:

To make an effective sentence generator we will need to do a lot of Natural Language Processing.There are many tools and libraries available in Python to solve NLP problems. Some of the most popular natural language processing libraries that are available are as follows:

1. spaCy:

spaCyis a relatively young project that labels itself as "industrial-strength natural language processing". The library provides most of the standard functionality (tokenization, PoS tagging, parsing, named entity recognition) and is built to be lightning fast.

2. Natural Language Toolkit:

Natural Language Toolkit is fairly mature (it's in development since 2001) and has positioned itself as one of the primary resources when it comes to Python and language processing.

3. Gensim:

Gensim is a fairly specialized library that is highly optimized for (unsupervised) semantic (topic) modelling. Semantic analysis and topic modelling in particular are a very specific sub-discipline of NLP, but an important and exciting one.

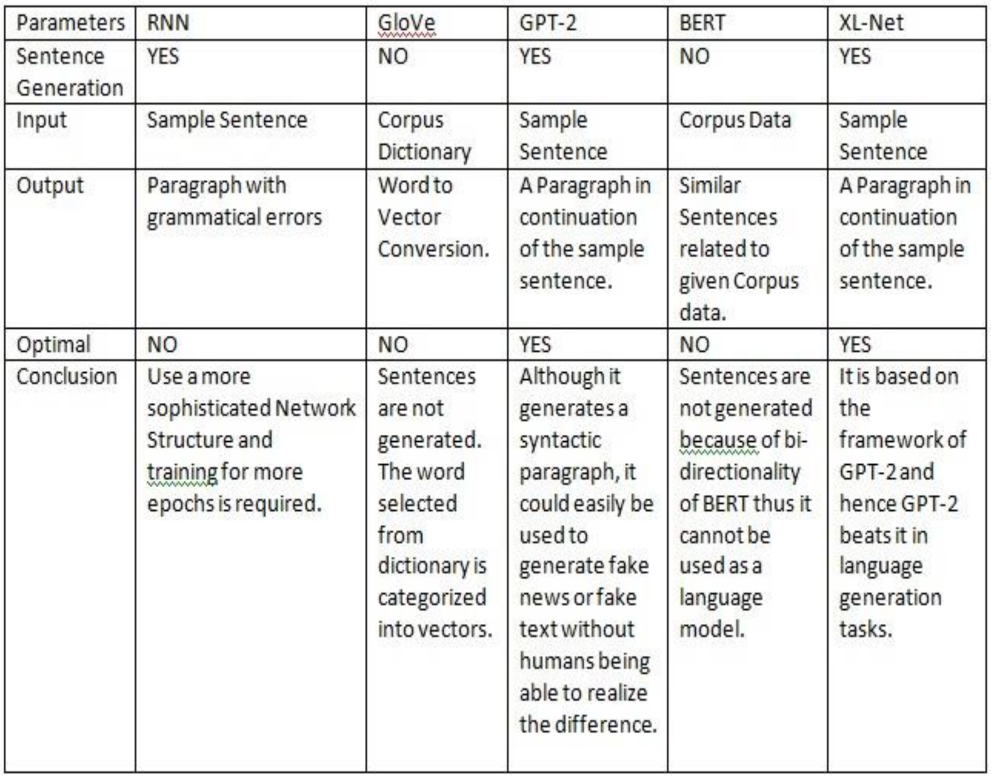
4. Pattern:

Pattern is a web mining library (module) for Python that can be used to crawl and parse a variety of sources such as Google, Twitter, and Wikipedia. However, Pattern comes packed with various NLP tools (PoS tagging, n-Grams, sentiment analysis, WordNet), machine learning capabilities (vector space models, clustering, classification), and various tools for conducting network analysis.

5. Polyglot:

Polyglot is primarily designed for multilingual applications. Within the space of simultaneously dealing with various languages, it provides some very interesting features such as language detection and transliteration that are usually not as pronounced in other packages.

2.2. Observations on Existing Work



Parameters	RNN	GloVe	GPT-2	BERT	XL-Net
Sentence Generation	YES	NO	YES	NO	YES
Input	Sample Sentence	Corpus Dictionary	Sample Sentence	Corpus Data	Sample Sentence
Output	Paragraph with grammatical errors	Word to Vector Conversion.	A Paragraph in continuation of the sample sentence.	Similar Sentences related to given Corpus data.	A Paragraph in continuation of the sample sentence.
Optimal	NO	NO	YES	NO	YES
Conclusion	Use a more sophisticated Network Structure and training for more epochs is required.	Sentences are not generated. The word selected from dictionary is categorized into vectors.	Although it generates a syntactic paragraph, it could easily be used to generate fake news or fake text without humans being able to realize the difference.	Sentences are not generated because of bi-directionality of BERT thus it cannot be used as a language model.	It is based on the framework of GPT-2 and hence GPT-2 beats it in language generation tasks.

Figure 1. Comparison between various models.

CHAPTER 3
PROPOSED METHODOLOGY

3. Proposed Methodology/Approach:

A methodology is a model, which project managers employ for the design, planning, implementation and achievement of their project objectives. There are different project management methodologies to benefit different projects.

For example, there is a specific methodology, which NASA uses to build a space station while the Navy employs a different methodology to build submarines. Hence, there are different project management methodologies that cater to the needs of different projects spanned across different business domains.

3.1. Problem Definition:

In today’s fast paced and internet ruled word, there is still no system to generate syntactical English sentences. Already available web apps only correct the given statements or point out errors. None of the systems are able to generate sentences on its own. Even if the sentences are generated, they are incomplete in most cases. But, English is a vastly used language so to have a system that generates meaningful sentences based on inputs is what our system will focus on. We will provide various classes based on which the system will generate a meaningful and correct sentence on its own using the data set.

3.2. Scope:

The system will generate English sentences for the input provided by the user. This input will consist of subject, verb, object trio on the basis of which the sentence has to be generated. The scope of the system is limited to generation of sentences based on the templates and category of the particular sentence.

- The output sentences will be varying from simple to complex in nature.
- The simple sentences will contain the basic structure of a sentence that will be easy to read and comprehend. The complex sentences will make use of multiple phrases and higher level vocabulary. Thus making suitable for all the levels of education.

3.2.1. Assumptions and Constraints:

Ambiguity is one of the greatest challenges to NLP:

For example:

Fed raises interest rates, where "raises" is the verb, and "Fed" is the noun phrase

Fed raises interest rates, where "interest" is the verb, and "Fed raises" is the noun phrase

This ambiguity occurs at many levels:

- the acoustic level: e.g. mixing up similar-sounding words
- the syntactic level: e.g. multiple plausible grammatical parsing of a sentence
- the semantic level: e.g. some words can mean multiple things ("bank" as in a river or a financial institution); this is called word sense ambiguity
- the discourse(multi-clause) level: eg.unclear what a pronoun is referring to
- Other challenges include:
- Non-standard English: for instance, text shorthand, phrases such as "SOOO PROUD" as opposed to "so proud", or hash tags, etc
- Segmentation issues: [the] [New] [York-New] [Haven] [Railroad] vs. [the] [New York]-[New Haven] [Railroad]
- Idioms (e.g. "get cold feet", doesn't literally mean what it says)
- Neologisms (e.g. "unfriend", "retweet", "bromance")
- World knowledge (e.g. "Mary and Sue are sisters" vs "Mary and Sue are mothers.")

Tricky entity names: "Where is A Bug's Life playing", or "a mutation on the for gene"
The typical approach is to codify knowledge about language & knowledge about the world and find some way to combine them to build probabilistic models.

3.3. Proposed System Approach:

The proposed system approach is the way or the process of developing the entire proposed system. The approach or the method is the carefully executed plan which is formulated and implemented for successful development of the project in a disciplined manner.
The name of our Proposed System is “Smart Sentence Generator”.
In this system, we have followed a particular approach to develop each of the modules of the system. Firstly, we have to design the templates for the type of sentences we are going to design and then we implement the code for the sentence based on the template created using our own proposed algorithm, using Element Tree and NLTK library to generate the sentence based on SVO (Subject-Verb-Object) model.

3.3.1. Features of the Proposed System:

As mentioned in the literature study for the project, there is no model or algorithm that provides you with the correct sentences syntactically as well as semantically. We have seen many different models and algorithms and the outcome of all of those came to be the same i.e., Sentences are incomplete, there are grammatical errors and also the sentences that are generated are meaningless.

So, the feature of the proposed system is that we are able to generate meaningful sentences with no grammatical errors. This is because the templates and the algorithms are devised according to the proposed algorithm to get the correct output.

Nonetheless, to name a few features of the Proposed System :

- Sentences generated are meaningful.
- Sentences generated are complete.
- Sentences generated are grammatically correct.
- Sentences generated are based on the templates of a particular sentence.
- Simple, Interrogative, Exclamatory, Compound and Complex kinds of sentences.

CHAPTER 4

PROJECT MANAGEMENT

4. Project Management:

Project management is the practice of initiating, planning, executing, controlling, and closing the work of a team to achieve specific goals and meet specific success criteria at the specified time. The primary challenge of project management is to achieve all of the project goals within the given constraints. This information is usually described in project documentation, created at the beginning of the development process. The primary constraints are time, quality and budget. The secondary and more ambitious challenge is to optimize the allocation of necessary inputs and apply them to meet predefined objectives.

4.1. Project Schedule:

A project schedule is a document collecting all the work needed to deliver the project on time. A project is made up of many tasks, and each task is given a start and end (or due date), so it can be completed on time. Likewise, people have different schedules, and their availability and vacation or leave dates need to be documented in order to successfully plan those tasks.

4.1.1. Task Network Diagram:

Task Network Schedule is a diagram that depicts the task flow of the development of the system. It starts with the basic components and goes on till the deployment of the project. It shows what task the project should start with and then which task should be followed or be taken upon after which task and in which order. It shows the tasks which go on in a loop and also the tasks which need to be done to get to another task. To sum up, it provides you the workflow of which task is to be performed next to be right on track and which task to be followed if some mishaps occur in the development process.

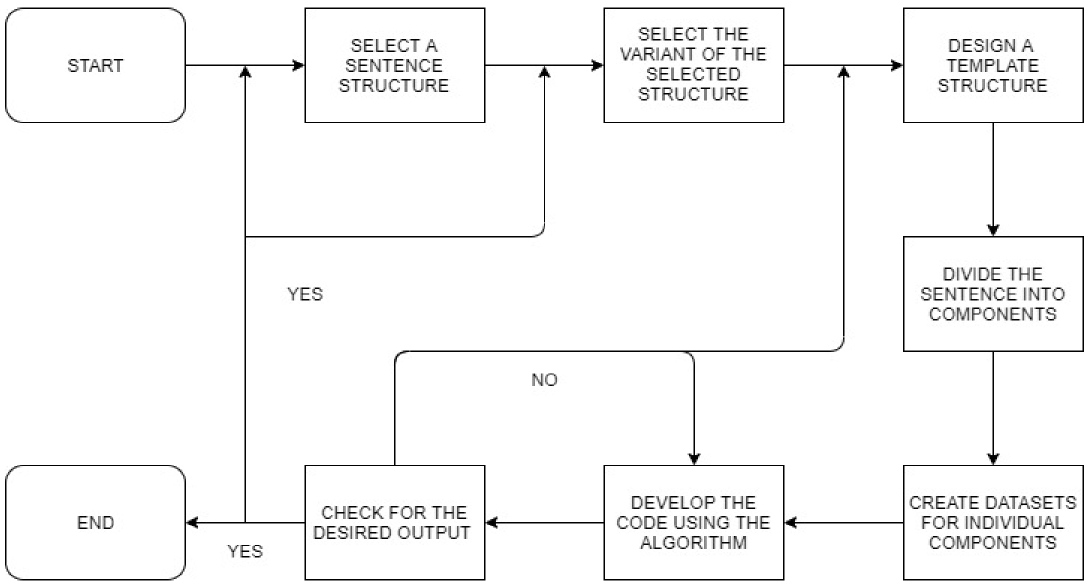

START
SELECT A SENTENCE STRUCTURE
SELECT THE VARIANT OF THE SELECTED STRUCTURE
DESIGN A TEMPLATE STRUCTURE
DIVIDE THE SENTENCE INTO COMPONENTS
CREATE DATASETS FOR INDIVIDUAL COMPONENTS
DEVELOP THE CODE USING THE ALGORITHM
CHECK FOR THE DESIRED OUTPUT
END
YES
NO
YES

Figure 2. Task Network Diagram

The task network schedule for our proposed system is shown above.

We start by choosing a sentence structure for which we are going to generate the sentences. Once we have selected the sentence type, we need to further choose which variant of the sentence are we going to generate. According to the variant chosen, we need to design a template structure for the sentence generation of that variant. Once the template is designed, the sentence is divided into various components for which, sample datasets have to be created. Finally, we need to develop the code according to the algorithm and compile it to check the output. If the output is correct, we can go for the next variant or if the variants are taken care of, then we can go for the next type of sentence in the proposed system. If the output is incorrect, we can check and change the code or we can start off with a new template structure altogether if changing the code is not working.

4.1.2. Timeline Chart:

The timeline chart is a chart which defines milestones or definite points during the course of the entire project development. It diagrammatically represents the tasks which need to be performed with the starting and ending date and the duration of each task calculated accordingly. It is also known as "Gantt Chart". So, basically all the tasks defined with the time span must be followed strictly and in a disciplined manner to meet the established deadlines mutually agreed by the developer and the client.

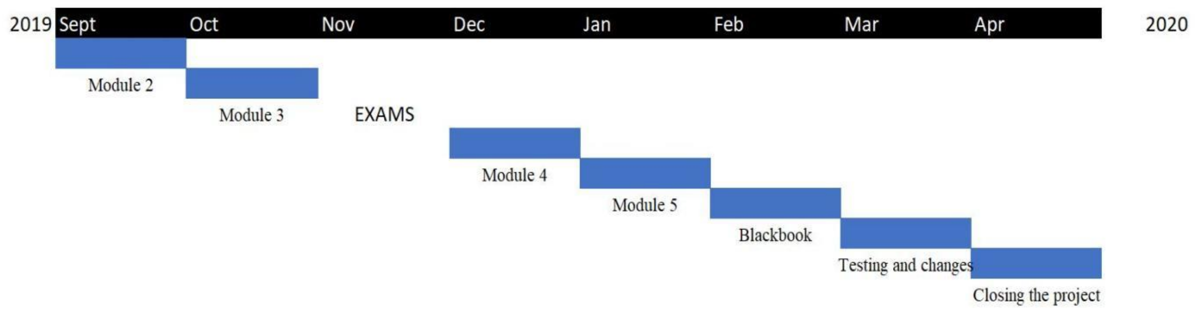


Figure 3. Timeline chart

As our proposed system has modules in which we have to generate various kinds of sentences, the Gantt Chart is not as detailed as it is normally for other projects in general. We have 5 modules that need to be implemented and furthermore, we have a black book creation as well as a testing and changing phase. Therefore, to sum up, we have around 7-8 phases and similar number of months. Hence, we are assigning a months period for every new module, which will take care of one type of sentence that we are going to generate with the variants of those sentences. Also, we have a month's period to create a Black book for the project undertaken to carefully examine and include all the important information needed to create a perfect black book. Finally, we have a month's period for the testing and change making process after the whole system is generated to make sure there are no errors or glitches that may hamper the systems working or any uncertainty that may arise that we are not aware of before deploying the project so that changes can me made immediately to correct the errors and make the system error-free and accurate in functioning.

4.2. Feasibility Schedule:

Feasibility Study is an assessment of the practicality of a proposed project or system. A feasibility study aims to objectively and rationally uncover the strengths and weaknesses of an existing system, opportunities and threats, the resources required to carry through, and ultimately the prospects for success. In its simplest terms, the two criteria to judge feasibility are cost required and value to be attained. It is used to study by taking into consideration various factors which can be given by the acronym TELOS. TELOS stands for technical, economic, legal, operational and scheduling. A feasibility study evaluates the project's potential for success.

4.2.1. Technical Feasibility:

The technical feasibility defines the technical criteria that should be satisfied in order to make the system a success. Various technical conditions have to be met to satisfy technical feasibility. The technical feasibility for our project can be satisfied by the following conditions:

- RAM of the system should be 16gb or more.
- GPU is required for processing data.
- Fast processor, something not less than an Intel i7 5th gen.
- SSD memory of about 128 or 256GB to increase loading and processor speed.

4.2.2. Operational Feasibility:

Operational feasibility is the measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development.

The operational feasibility assessment focuses on the degree to which the proposed development project fits in with the existing business environment and objectives with regard to development schedule, delivery date, corporate culture and existing business processes.

Our proposed system generates sentences with better efficiency than the existing systems.

4.2.3. Economical Feasibility:

Economic feasibility is the Analysis of a project's costs and revenues in an effort to determine whether or not it is logical and possible to complete. The economic feasibility of our project is that the GPU required for processing are expensive and require high power supply. However, we have rented an online GPU that serves the processing capabilities for our system to run successfully. For our solution, the libraries and the development environment we are using are free of cost. Our solution does not include any hardware requirements for deploying, hence hardware charges are not applicable for our solution.

Also, our system would generate sentences digitally. This helps us save the money which would otherwise be utilized on papers, which in turn helps save the environment by not cutting trees which is the main cause of global warming.

4.3. Project Resources:

Project Resources are the resources that are needed to build or develop the system. Without the required resources, one cannot develop the project.

The resources should be ready and updated on a regular basis according to the needs and requirement of these resources. The resources must also be upgraded to the latest available version to catch up with the development in the technologies.

Resources can be classified into various categories viz – hardware, software and operational resources.

4.3.1. Hardware Requirements:

Hardware requirements are the hardware components that are needed for the development of the system. They can be a variety of components to enhance the development process and also the working of the proposed system. The hardware requirements for the proposed system are:

- RAM of system should be 16gb or more.
- Fast processor, something not less than an Intel i7 5th gen.
- SSD memory of about 128 or 256GB to increase loading and processor speed.

4.3.2. Software Requirements:

Software requirements are those software components or applications required to develop the system. These requirements include the IDE, the applications used to develop the system, editors and compilers and other software entities that are required for the proper development of the system. Software requirement for our proposed system are:

- Any Python Editor (Anaconda Python – Jupyter Notebook).
- Any Python Compiler (Anaconda Python – Jupyter Notebook).
- Windows 10.
- Text Editor for templates.
- Any Browser for the XML documents to be verified.
- Python Libraries.

4.4. Project Estimation:

4.4.1. COCOMO Estimation Model:

We have taken into consideration Basic COCOMO Model to estimate the development time of our project. Evaluation the parameters like the size of code, team size, developer experience, working environment, innovation and deadline we can say that we are following the organic development model as per COCOMO model. We estimate our project to be of 12 KLOC (kilo lines of code).

Having the following details as COCOMO model's guidelines, we get:

Effort = 2.4 * (12) 1.05 = 33 person-months

Development Time = 2.5 * (33) 0.38 = 9 months

Thus, from COCOMO model we can estimate that the project can be completed in 9 months as per our project management schedule.

4.4.2. Function Point Analysis:

For function point analysis, we have following values for our project:

User input =50

User output = 70

User enquires = 10

User files = 10

External interfaces = 9

Also, we have Complexity Adjustment Factor (CAF) and all weighing factors as average. So we can calculate Functional points as:

Unadjusted Functional Point (UAF) = 50*4 +70*5+10*4+ 10*10 + 9*7 = 753

Complexity Adjustment Factor (CAF) = 0.65 +(0.01 * 42) = 1.07

Functional points = 753 * 1.07 = 805.71

4.5. Risk Management Mitigation Planning:

Table 1. Risk Mitigation and Management

Risk Event	Risk Severity	Risk Effect	Risk Mitigation	Risk Contingency
Non recognizable text format	Intolerable	Empty spaces in sentences	Revise data every time, pre-processing should be carried out	Eliminate use of those words
Non relationship exist between words	Undesirable	Semantically wrong sentences	Redefine correlation	Elimination of dissimilar correlation
System freezing	Tolerable	System failure	Check the ide and file paths	Restart the IDE
Unpredicted nature of algorithm	Tolerable	Generation of meaningless sentences	Refine algorithm	Use different combinations of input words

CHAPTER 5

SYSTEM DESIGN

5. System Design:

System design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. System Design is an essential module in project development as it gives an overview of the entire project in the form of diagrams. This is effective and very much helpful to explain the project to the people or the customers who have no technical background. The diagrams prove to be an effective way to demonstrate the functioning of the system diagrammatically.

5.1. Design Diagrams

Various diagrams constitute together to form the system design criteria for any undertaken project. These diagrams represent some specific area or some particular aspect of the project.

5.1.1. UML Diagrams

Use case diagrams are usually referred to as behaviour diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors).

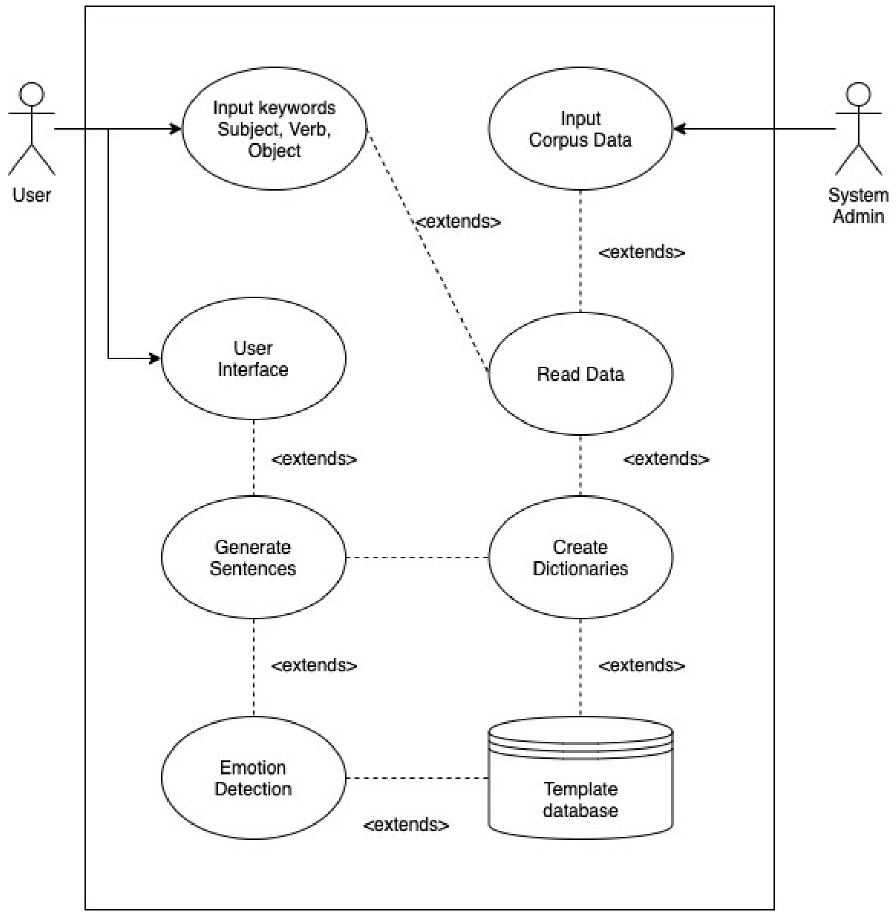


Figure 4. Use Case Diagram

5.2. System Architecture

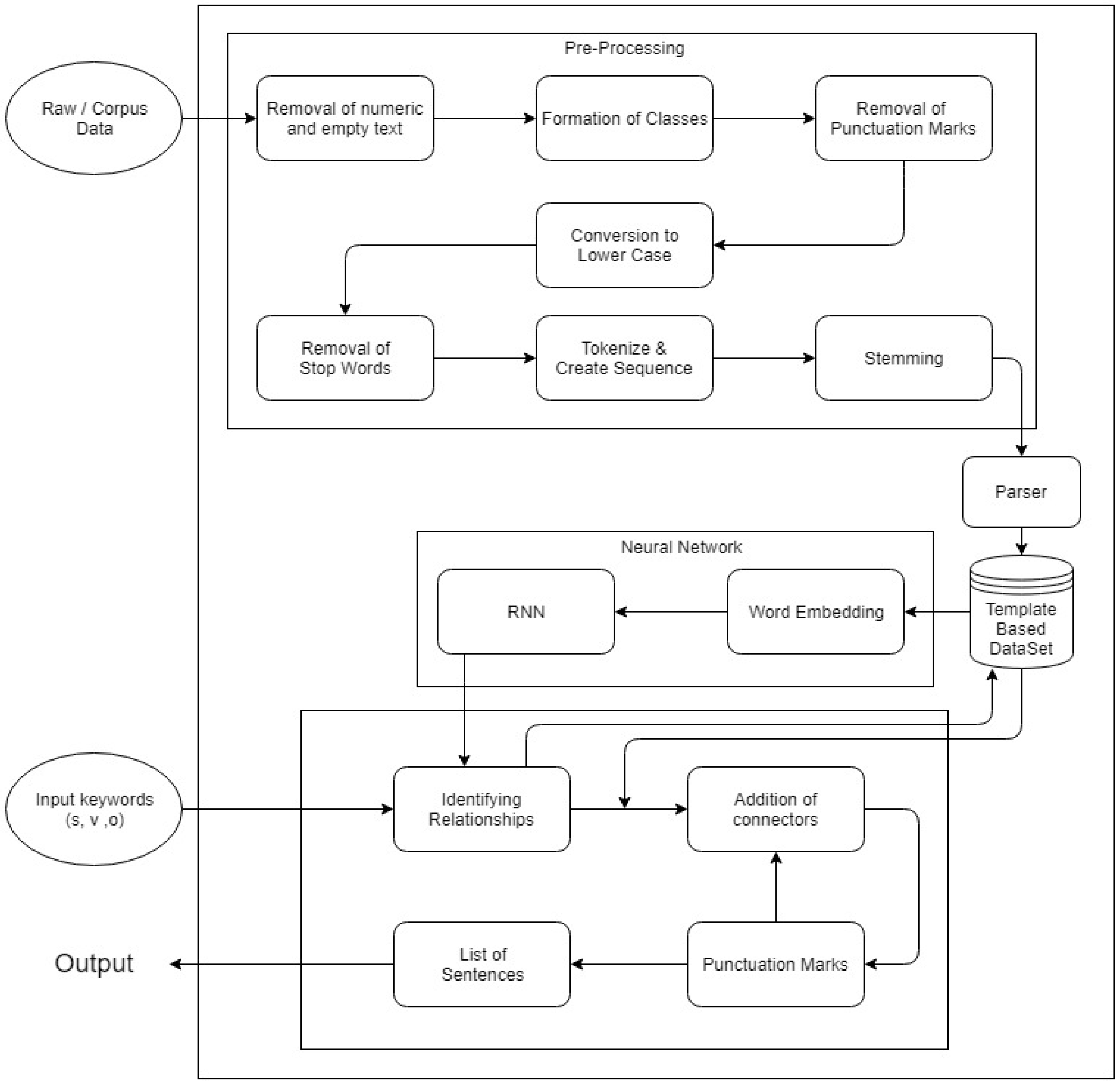


Figure 5. System Architecture

The System admin : The system admin has the functionality for providing the corpus data collected from various sources. Improvement of the Template based dataset by providing new data items at regular intervals.

Pre-processing module : Performs the task of cleansing of the raw data. It also extracts words and their weight vectors. Tokenizing & Stemming is also carried out by this module.

Parser : A Python Parser module which will generate data for the template dataset.

User : The user has to input the subject, verb, object to the system for obtaining the sentences created on the basis of these keywords.

System : The system performs sentence generation using the keywords, connectors and punctuation marks.

- Text preprocessing is one of the most important tasks in Natural Language Processing(NLP). For instance, you have to remove all punctuation marks from text documents before they can be used for text classification. Similarly, you have to extract numbers from a text string. Writing manual scripts for such preprocessing tasks requires a lot of effort and is prone to errors. Keeping in view the importance of these preprocessing tasks, the Regular Expressions (aka Regex) have been developed in different languages in order to ease these text preprocessing tasks.
- A majority of the words in a given text are connecting parts of a sentence rather than showing subjects, objects or intent. Word like "the" or "and" can be removed by comparing text to a list of stop word.
- Tokenization describes splitting paragraphs into sentences, or sentences into individual words. For the former Sentence Boundary Disambiguation (SBD) can be applied to create a list of individual sentences. This relies on a pre-trained, language specific algorithms like the Punkt Models from NLTK.
- Understand parts of speech can make difference in determining the meaning of a sentence. Part of Speech (POS) often requires look at the proceeding and following words and combined with either a rule-based or stochastic method. It can than be combined with other processes for more feature engineering.
- Much of natural language machine learning is about sentiment of the text. Stemming is a process where words are reduced to a root by removing inflection through dropping unnecessary characters, usually a suffix. There are several stemming models, including Porter and Snowball. The results can be used to identify relationships and commonalities across large datasets.
- Lemmazation is an alternative approach from stemming to removing inflection. By determining the part of speech and utilizing WordNet's lexical database of English, lemmazation can get better results.
- Word embedding is the modern way of representing words as vectors. The aim of word embedding is to redefine the high dimensional word features into low dimensional feature vectors. In other words it represents words at an X and Y vector coordinate where related words, based on a corpus of relationships, are placed closer together. Word2Vec and GloVe are the most common models to convert text to vectors.

5.3. Algorithm:

Algorithm is defined as a task or process of developing code for performing the required function. It is well defined structure that has to be followed in sequence in order to execute a particular task. Algorithms are the basic building blocks for the software development of the system.

Our proposed system has self-designed templates and algorithm to generate sentences. This algorithm is based on the template structure that we have defined in the XML format. A dataset has been prepared according to the components needed in the sentence and values for those components are stored in a text file. The algorithm fetches the words of the sentence from such datasets created for various components of the sentence such as SVO or adjectives and adverbs.

Steps of the algorithm:

1. Select the sentence type for which the sentence needs to be generated.
2. Select the variant of the sentence type selected.
3. Create a template for the sentence structure.
4. Create datasets for various components in the sentence.
5. Develop the code using python.
6. Merge the output of every component to form a sentence.

CHAPTER 6

IMPLEMENTATION

6. Implementation:

Implementation is the working of the system as per the desired functionality. Proper implementation means getting the desired output for a predefined set of inputs.

Implementation depends on the algorithm generated and also accordingly to the process or task flow that is followed according to the various system designs devised and also according to the approaches and methodologies that are meant to be followed during the development process.

Implementation shows the destined working of the system with the purpose of the development of the system satisfied to get the beneficial success.

6.1. Working of System:

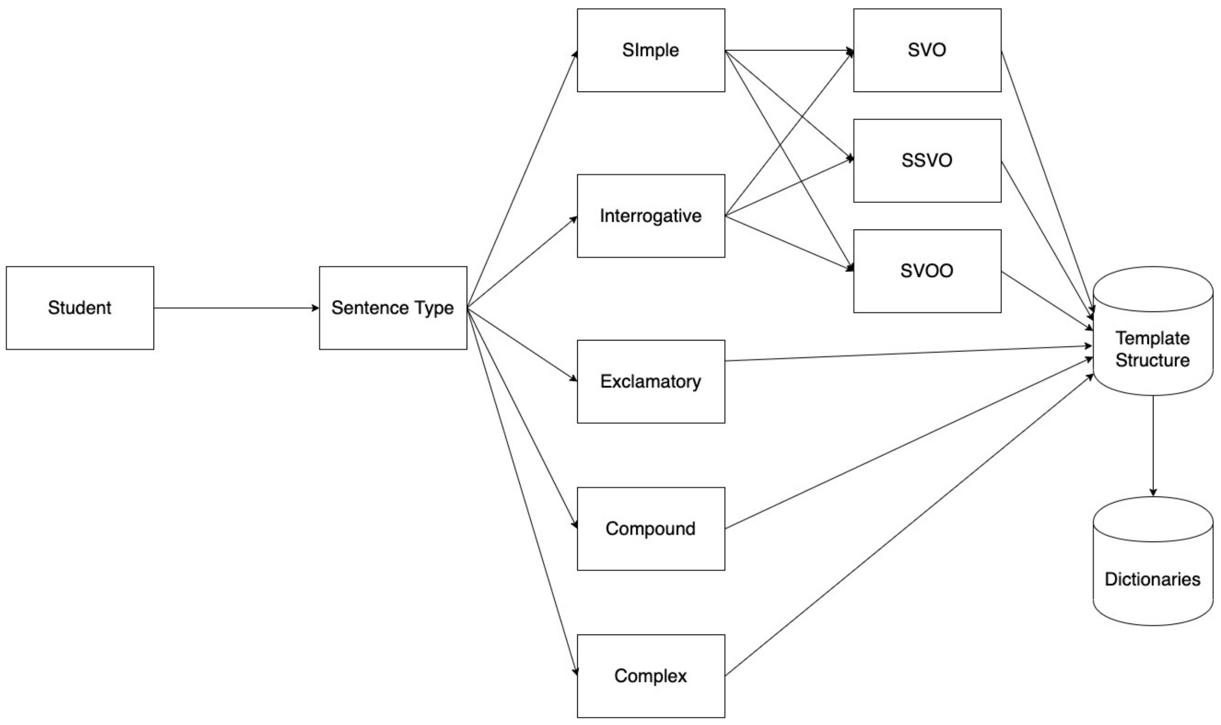


Figure 6 : Working of the system

The following are the steps for working of the system:

- The student selects the type of sentence that he/she needs to study for.
- Upon selecting the sentence type from Simple/ Interrogative/ Exclamatory/ Compound; the template structure for the selected type is retrieved.
- The template fetches relations and parts of speech, from the dictionaries to generate and display the sentence that has been queried.
- A number of variations are provided for each sentence category with its sub-categories to cover all the possible noun-verb relations and types for the student to study.

6.2. Templates:

The template structures are designed in XML format. The various types of sentences include simple sentences, interrogative sentences, exclamatory sentences, compound and complex sentences. The sentence structures are broken down into individual components being, the expression, subject, adjective, verb, adverb, objects and auxiliary verbs, etc. The components are basically the parts of speech of the English Language which are used to make a sentence or create a sentence. After the datasets are created, finally the algorithm is used to design the code. In the code, we are using the Python libraries like Element Tree to parse the XML data, access the parts of sentences to perform the desired operations for every part of the sentence for which we have devised the components.

EXAMPLES :

```
<simple>

<SENTENCE class="askable" NO="1">
<EXAMPLE>Mary asked questions in the class. | class="askable"</EXAMPLE>

<SUBJECTPHRASE >

<SNOUN class="name" ></SNOUN>

</SUBJECTPHRASE>

<VERBPHRASE>
<VERB class="askable"></VERB>
</VERBPHRASE>

<OBJECTPHRASE>

<CNOUN class="askable"></CNOUN>
<PREPOSITION class="place"></PREPOSITION>
<ARTICLE1 class="place"></ARTICLE1>
<CNOUN1 class="place"></CNOUN1>

</OBJECTPHRASE>

</SENTENCE>

</simple>
```

```
<interrogative>
<SENTENCE class="close" NO="1">
<EXAMPLE>When did you close the door? |class="close"</EXAMPLE>

<PREFIX>When</PREFIX>

<SUBJECTPHRASE>
<VERB class="do" ></VERB>
<PRONOUN class="fp"></PRONOUN>
</SUBJECTPHRASE>

<VERBPHRASE>
<VERB class="closeablepresent"></VERB>
</VERBPHRASE>

<OBJECTPHRASE>
<ARTICLE class="article"></ARTICLE>
<CNOUN class="closable"></CNOUN>
</OBJECTPHRASE>

</SENTENCE>
</interrogative>



<exclamatory>

<SENTENCE class="order" NO="1">
<EXAMPLE>Read the book, Amy! | class="order"</EXAMPLE>

<ORDERPHRASE>
<VERB class="readable" ></VERB>
<ARTICLE class="noun"></ARTICLE>
<CNOUN class="readable"></CNOUN>

</ORDERPHRASE>
```

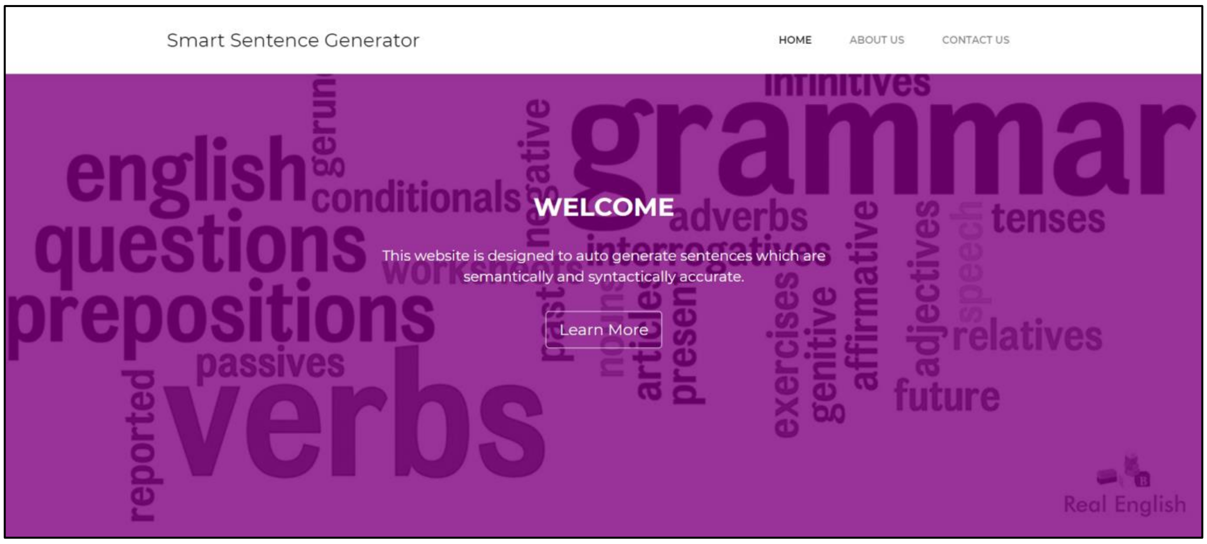
```
<SUBJECTPHRASE><SNOUN class="name" ></SNOUN></SUBJECTPHRASE>

</SENTENCE>

<SENTENCE class="order" NO="2">
<EXAMPLE>Shut the door, Amy! | class="order"</EXAMPLE>

<ORDERPHRASE>
<VERB class="openable" ></VERB>
<ARTICLE class="noun"></ARTICLE>
<CNOUN class="openable"></CNOUN>

</ORDERPHRASE>

<SUBJECTPHRASE><SNOUN class="name" ></SNOUN></SUBJECTPHRASE>

</SENTENCE>

<SENTENCE class="order" NO="3">
<EXAMPLE>Bring the thing, Amy! | class="order"</EXAMPLE>

<ORDERPHRASE>
<VERB class="bring" ></VERB>
<ARTICLE class="noun"></ARTICLE>
<CNOUN class="bring"></CNOUN>

</ORDERPHRASE>

<SUBJECTPHRASE><SNOUN class="name" ></SNOUN></SUBJECTPHRASE>

</SENTENCE>

<SENTENCE class="order" NO="4">
<EXAMPLE>Eat the veggies, Amy! | class="order"</EXAMPLE>

<ORDERPHRASE>
<VERB class="eatable" ></VERB>
<ARTICLE class="noun"></ARTICLE>
```

```
<CNOUN class="eatable"></CNOUN>

</ORDERPHRASE>

<SUBJECTPHRASE><SNOUN class="name" ></SNOUN></SUBJECTPHRASE>

</SENTENCE>
<SENTENCE class="order" NO="5">
<EXAMPLE>Drink the juice, Amy! | class="order"</EXAMPLE>

<ORDERPHRASE>
<VERB class="drinakble" ></VERB>
<ARTICLE class="noun"></ARTICLE>
<CNOUN class="drinakble"></CNOUN>

</ORDERPHRASE>

<SUBJECTPHRASE><SNOUN class="name" ></SNOUN></SUBJECTPHRASE>

</SENTENCE>

<SENTENCE class="order" NO="6">
<EXAMPLE>Pick up the thing, Amy! | class="order"</EXAMPLE>

<ORDERPHRASE>
<VERB class="pick" ></VERB>
<ARTICLE class="noun"></ARTICLE>
<CNOUN class="pick"></CNOUN>

</ORDERPHRASE>

<SUBJECTPHRASE><SNOUN class="name" ></SNOUN></SUBJECTPHRASE>

</SENTENCE>

<SENTENCE class="order" NO="7">
<EXAMPLE>Shut up your mouth, Amy! | class="order"</EXAMPLE>
```

```
<ORDERPHRASE>
<EXPRESSION>Shut up</EXPRESSION>
<PERSONALPRONOUN class="noun"></PERSONALPRONOUN>
<CNOUN class="order"></CNOUN>

</ORDERPHRASE>

<SUBJECTPHRASE><SNOUN class="name" ></SNOUN></SUBJECTPHRASE>

</SENTENCE>

</exclamatory>




<compound>
<SENTENCE class="bought" NO="1">
<EXAMPLE>Mary went to the market and she bought some fruits. |class="bought"</EXAMPLE>

<CLAUSE1>
<SNOUN class="name"></SNOUN>
<VERB class="move"></VERB>
<PREPOSITION class="place"></PREPOSITION>
<ARTICLE></ARTICLE>
<CNOUN class="shopping"></CNOUN>
</CLAUSE1>

<CONJUNCTION class="compulsory"></CONJUNCTION>

<CLAUSE2>
<PRONOUN class="male"></PRONOUN>
<VERB class="buy"></VERB>
<ADJECTIVE class="little"></ADJECTIVE>
<CNOUN class="eatable"></CNOUN>
</CLAUSE2>

</SENTENCE>
</compound>
```

```
<complex>
<SENTENCE class="play" NO="1">
<EXAMPLE>Whenever John was lonely, he played with his toys. |class="play"</EXAMPLE>

<PREFIX>Whenever</PREFIX>

<CLAUSE1>
<SNOUN class="name"></SNOUN>
<VERB class="duration"></VERB>
<ADJECTIVE class="alone"></ADJECTIVE>
</CLAUSE1>

<CLAUSE2>
<PRONOUN class="male"></PRONOUN>
<VERB class="play"></VERB>
<PREPOSITION class="use"></PREPOSITION>
<PRONOUN class="belonging"></PRONOUN>
<NOUN class="entertainment"></NOUN>
</CLAUSE2>

</SENTENCE>
</complex>
```

6.3. Output:



This website is designed to auto generate sentences which are semantically and syntactically accurate.

Figure 7. Home Page

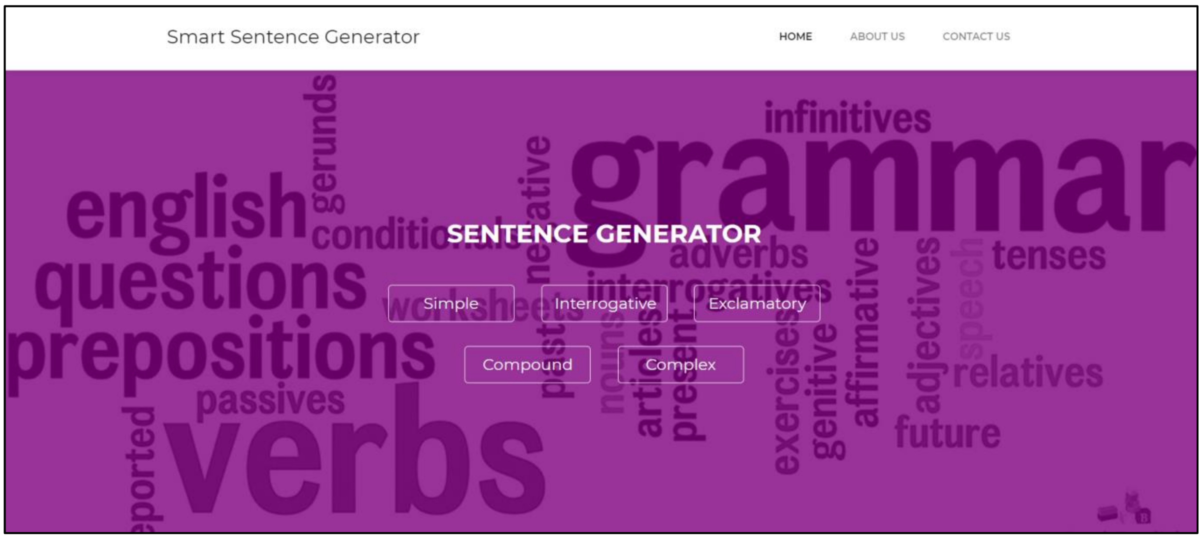


Figure 8. Types of sentences

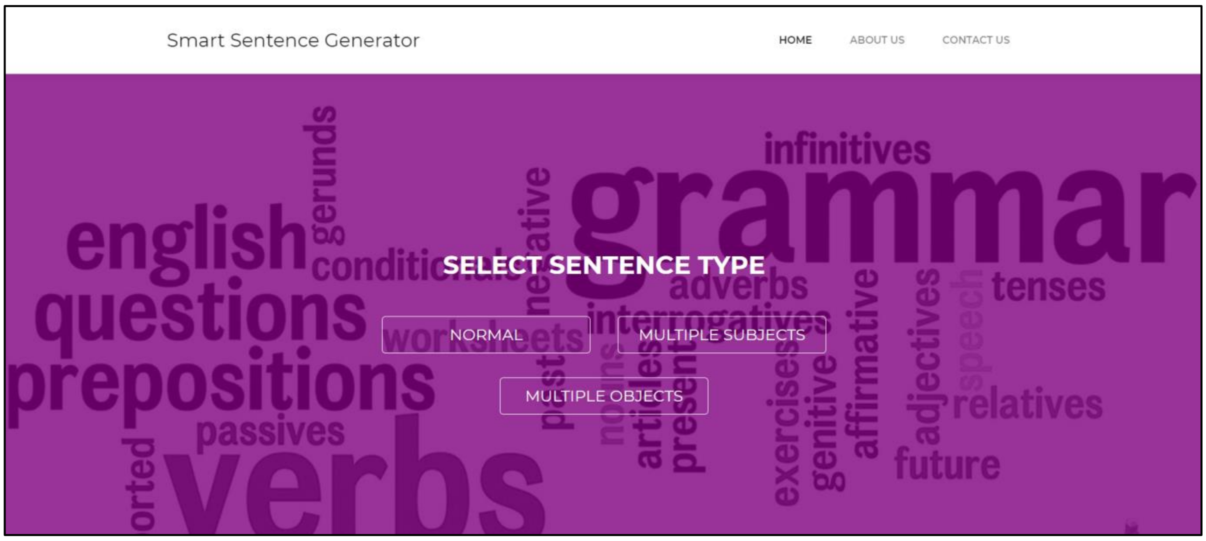


Figure 9. Simple Sentence Variations

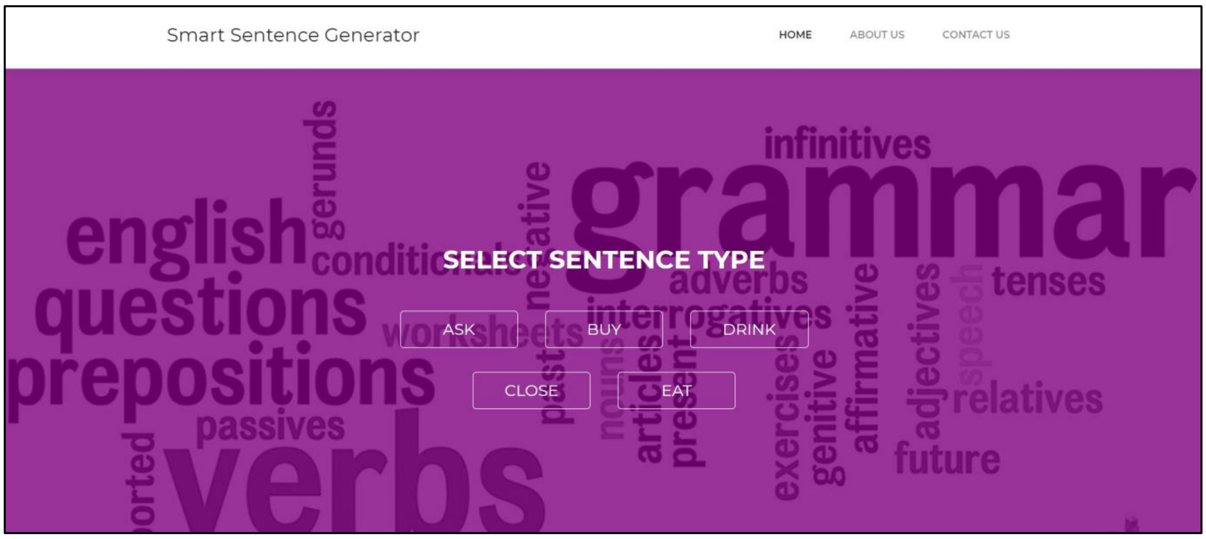
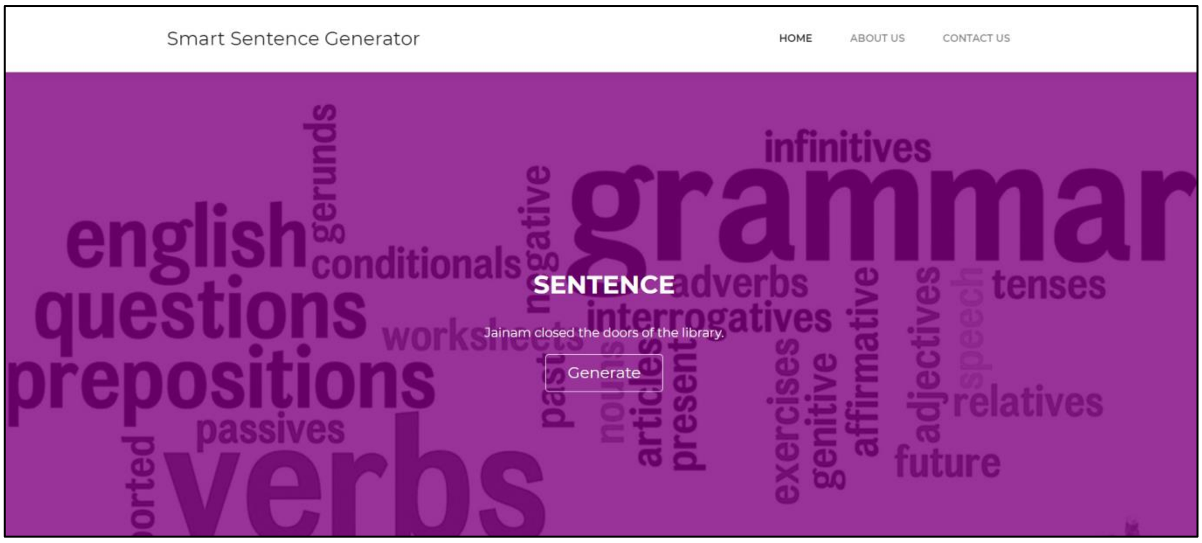
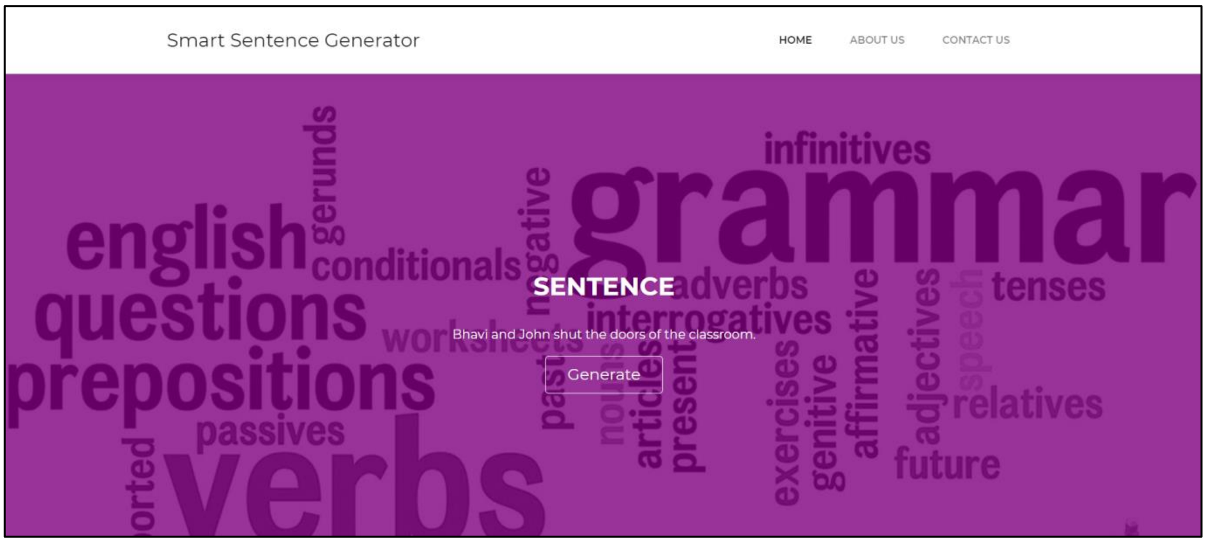


Figure 10. Simple Sentence Classes



Jainam closed the doors of the library.

Figure 11. Normal Simple Sentence Output



Bhavi and John shut the doors of the classroom.

Figure 12. Multiple Subjects Simple Sentence Output

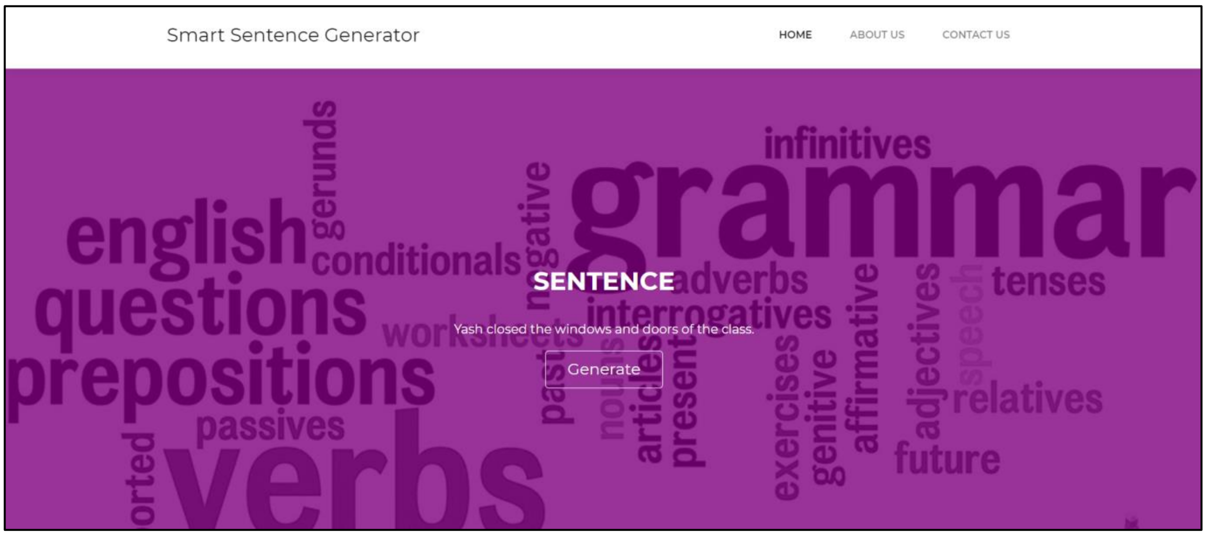


Figure 13. Multiple Objects Simple Sentence Output

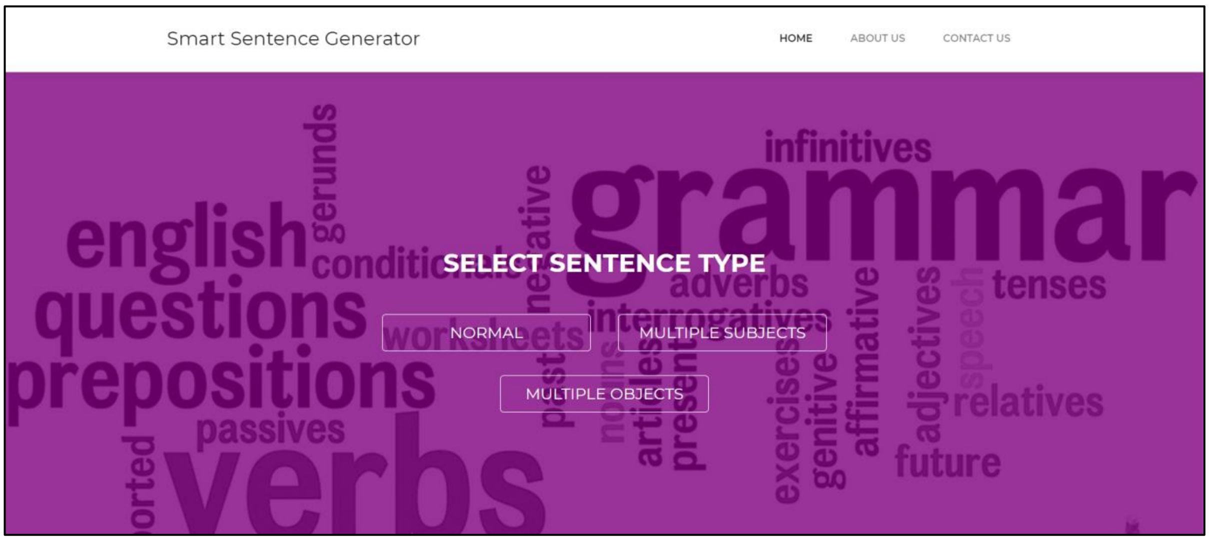


Figure 14. Interrogative Sentence Variations

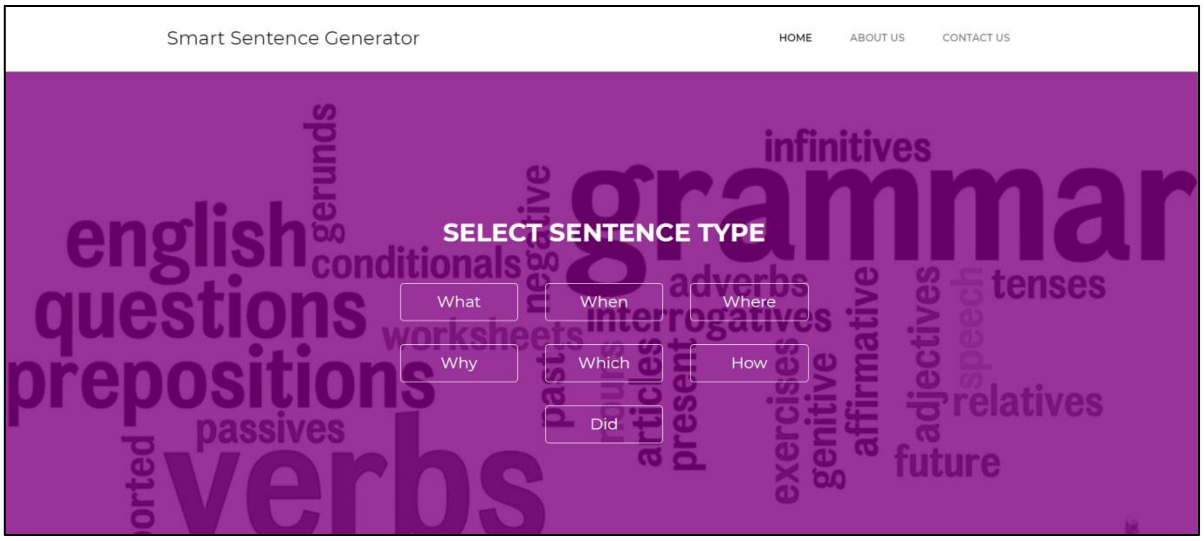
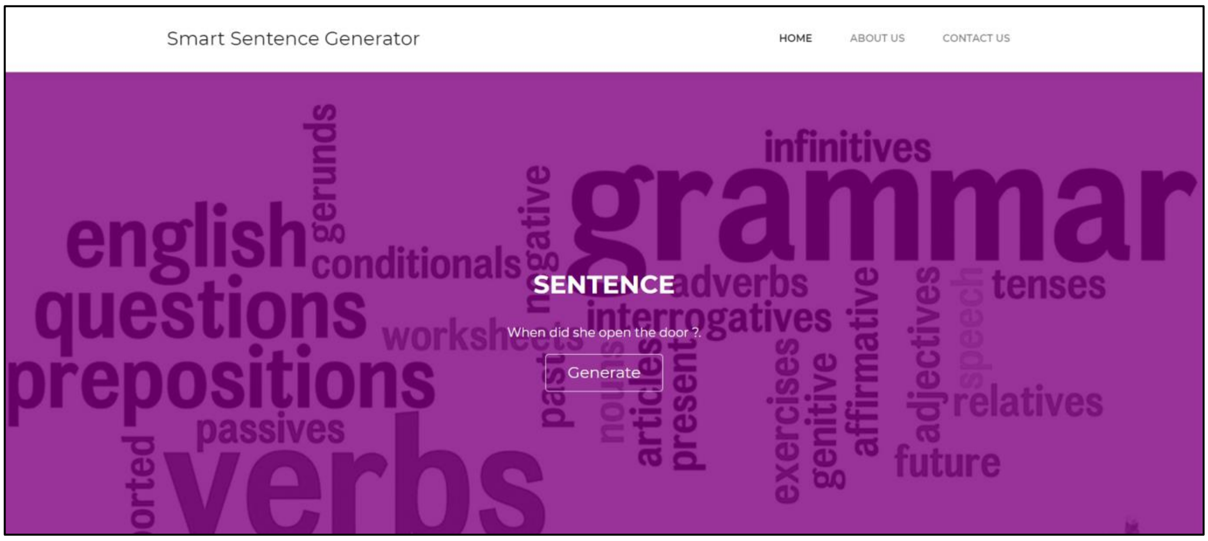
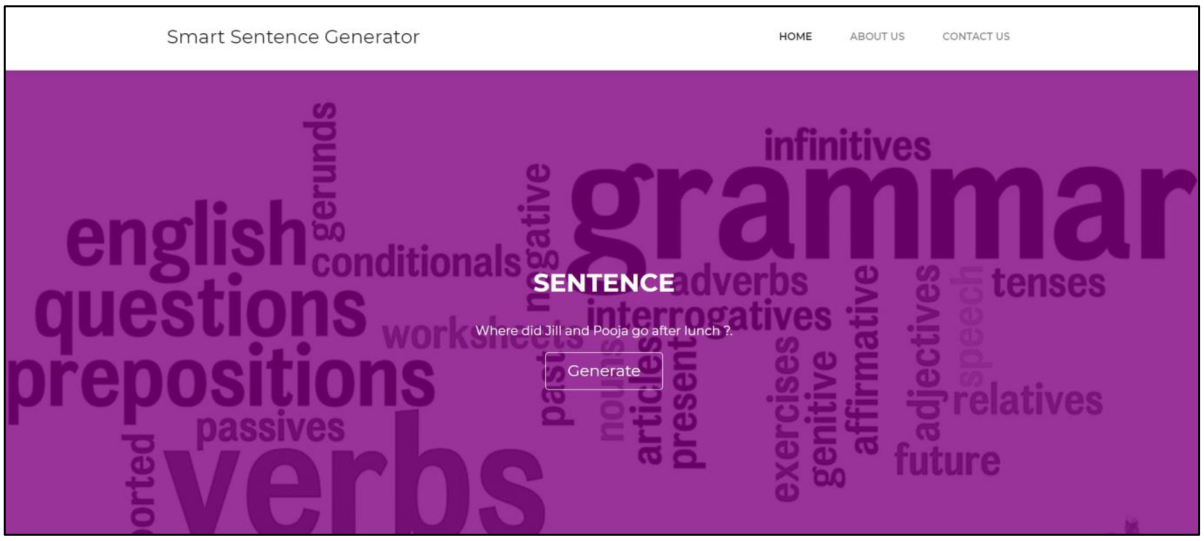


Figure 15. Interrogative Sentence Classes



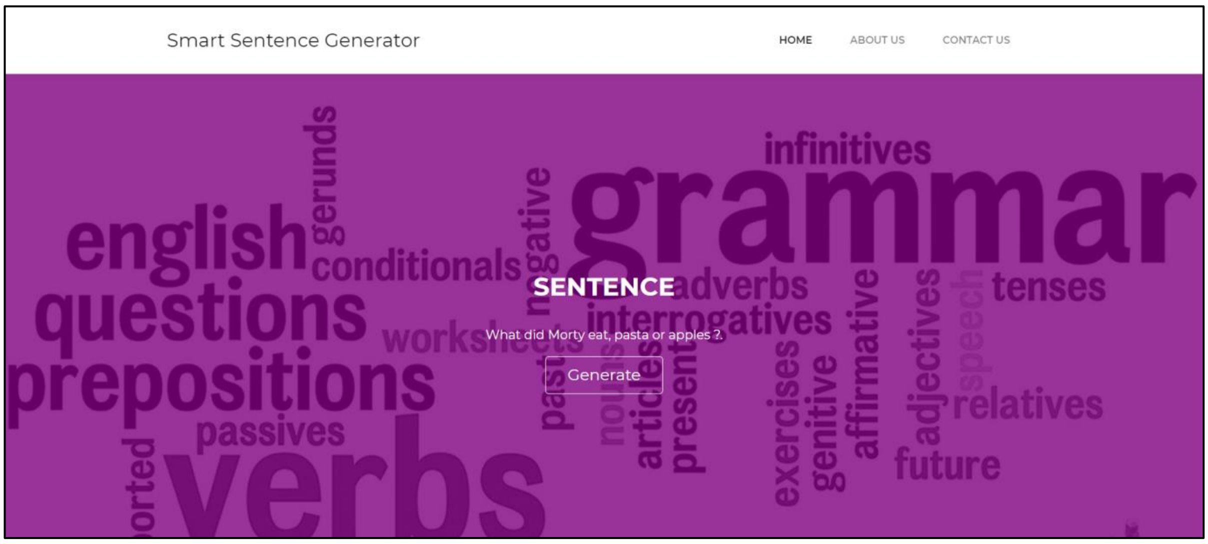
When did she open the door ?.

Figure 16. Normal Interrogative Sentence Output



Where did Jill and Pooja go after lunch ?.

Figure 17. Multiple Subjects Interrogative Sentence Output



What did Morty eat, pasta or apples ?.

Figure 18. Multiple Objects Interrogative Sentence Output

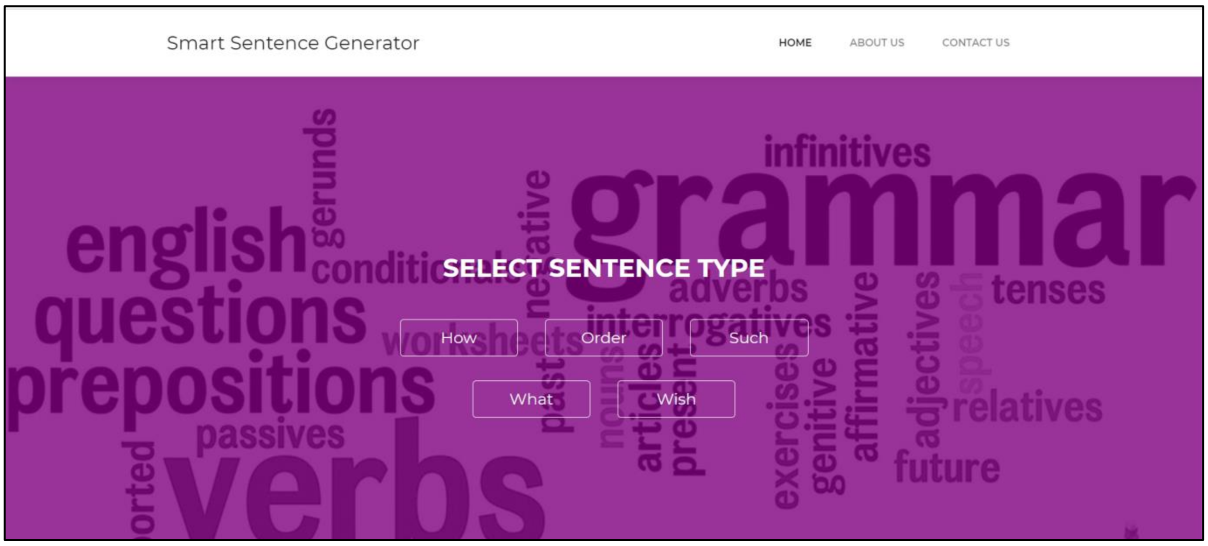


Figure 19. Exclamatory Sentence Variations

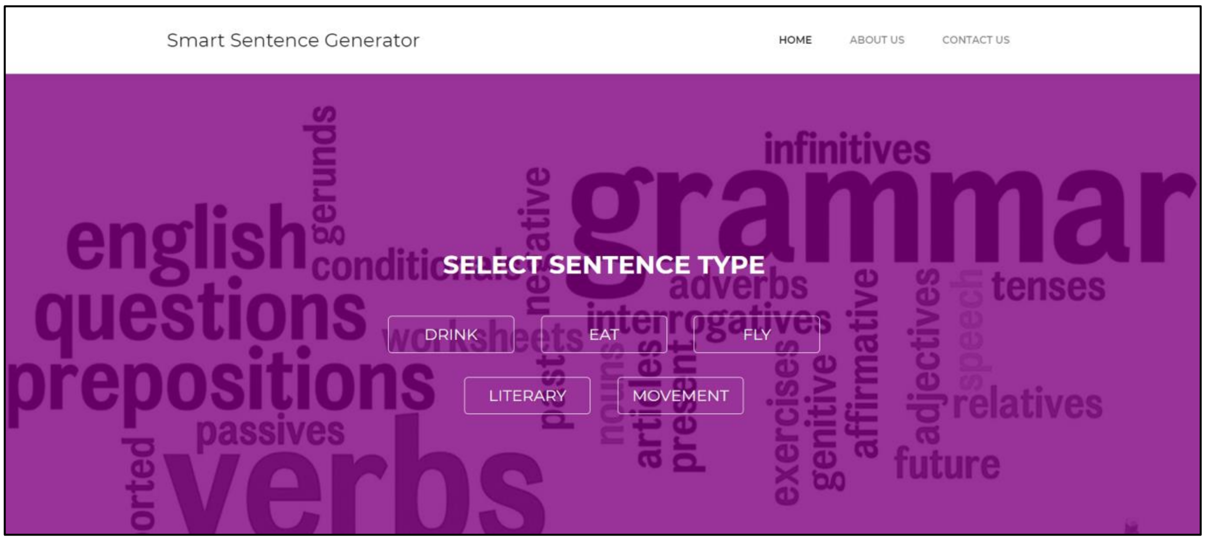
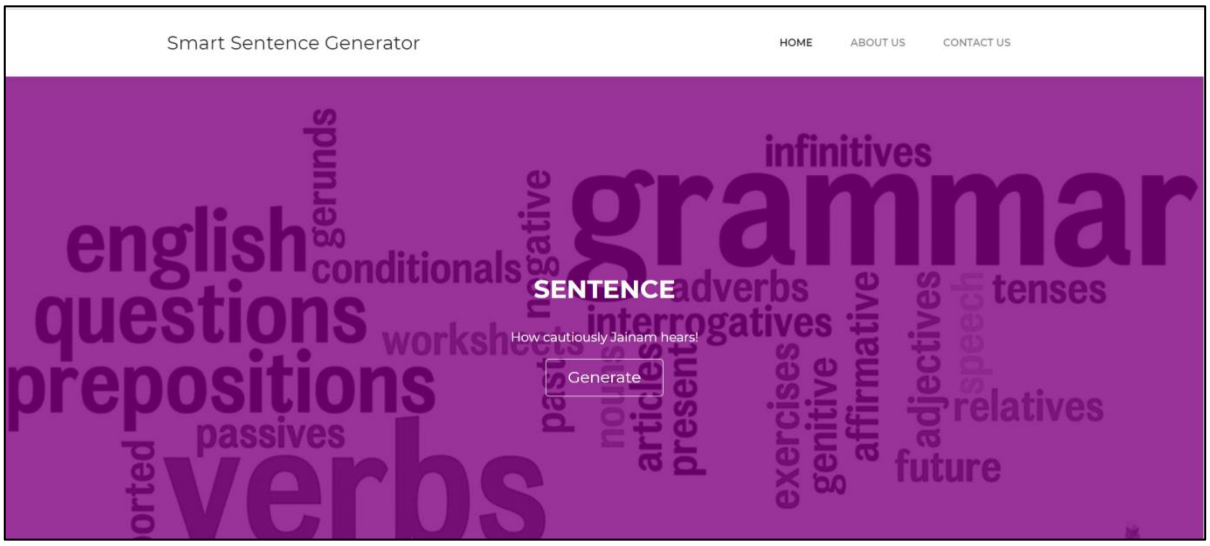


Figure 20. Exclamatory Sentence Classes for How



How cautiously Jainam hears!

Figure 21. Exclamatory Sentence Output for How

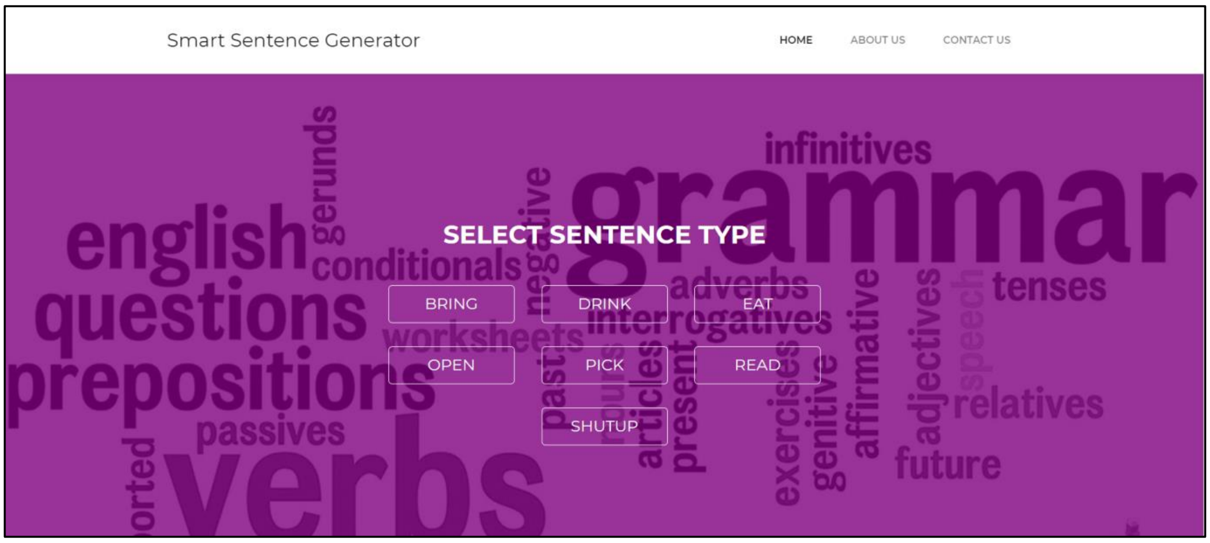
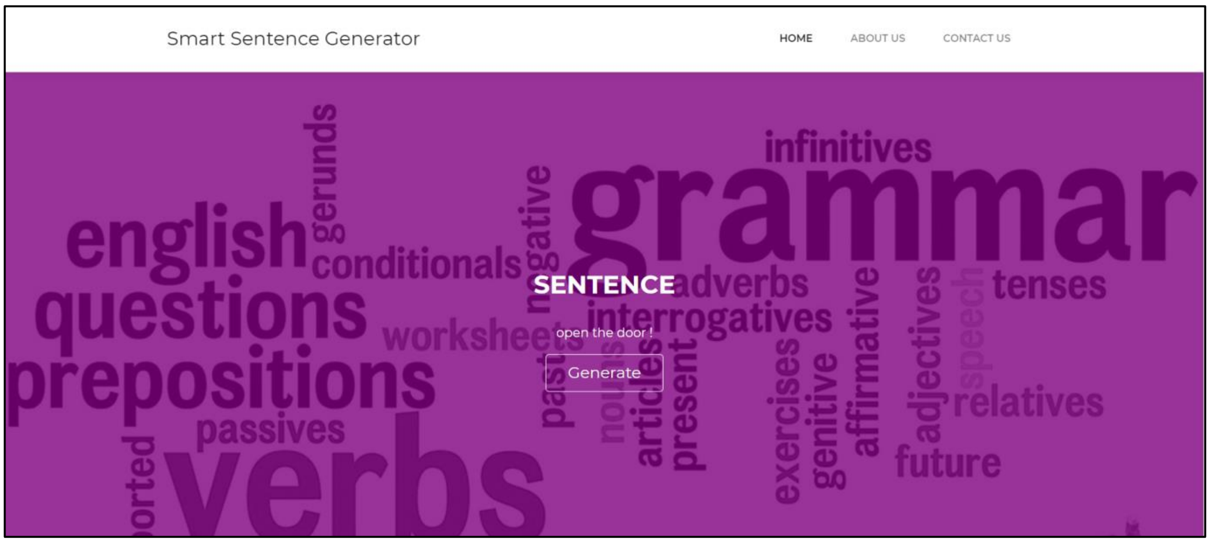
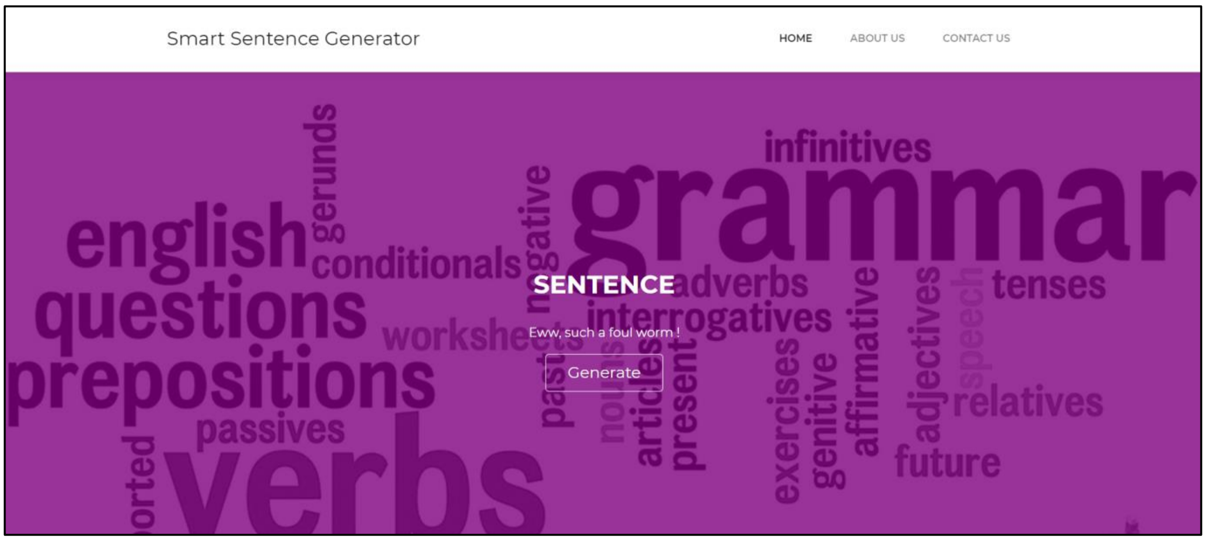


Figure 22. Exclamatory Sentence Classes for Order



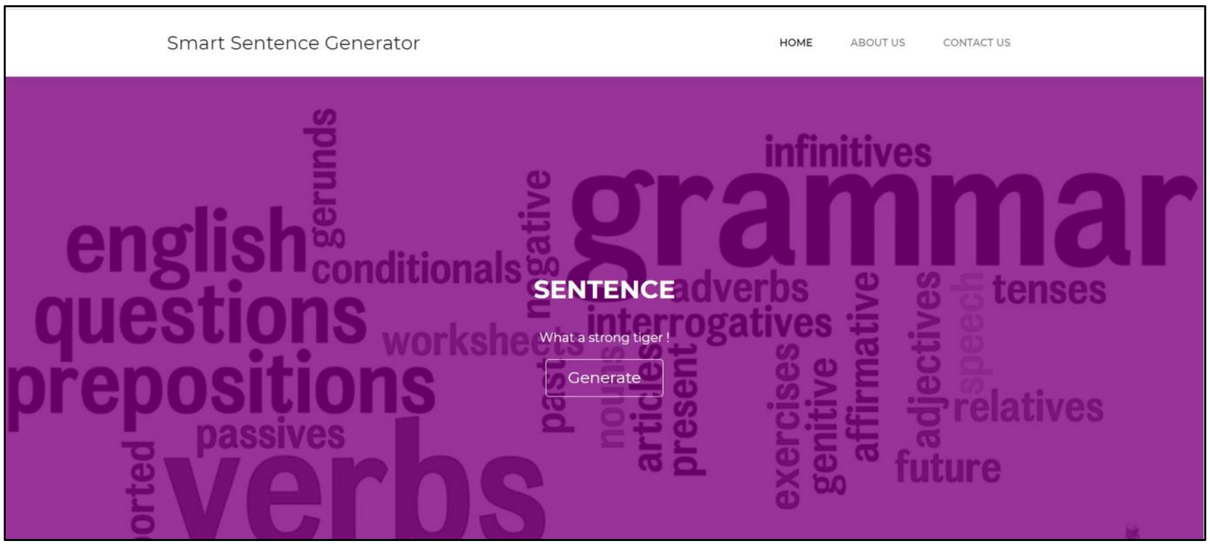
open the door !

Figure 23. Exclamatory Sentence Output for Order



Eww, such a foul worm !

Figure 24. Exclamatory Sentence Output for Such



What a strong tiger !

Figure 25. Exclamatory Sentence Output for What

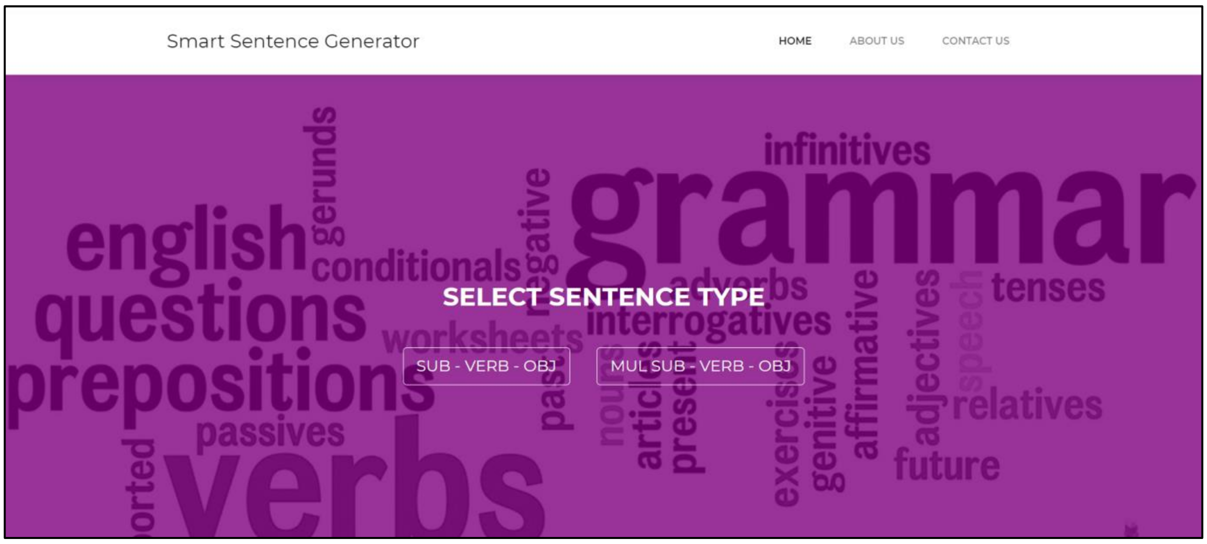
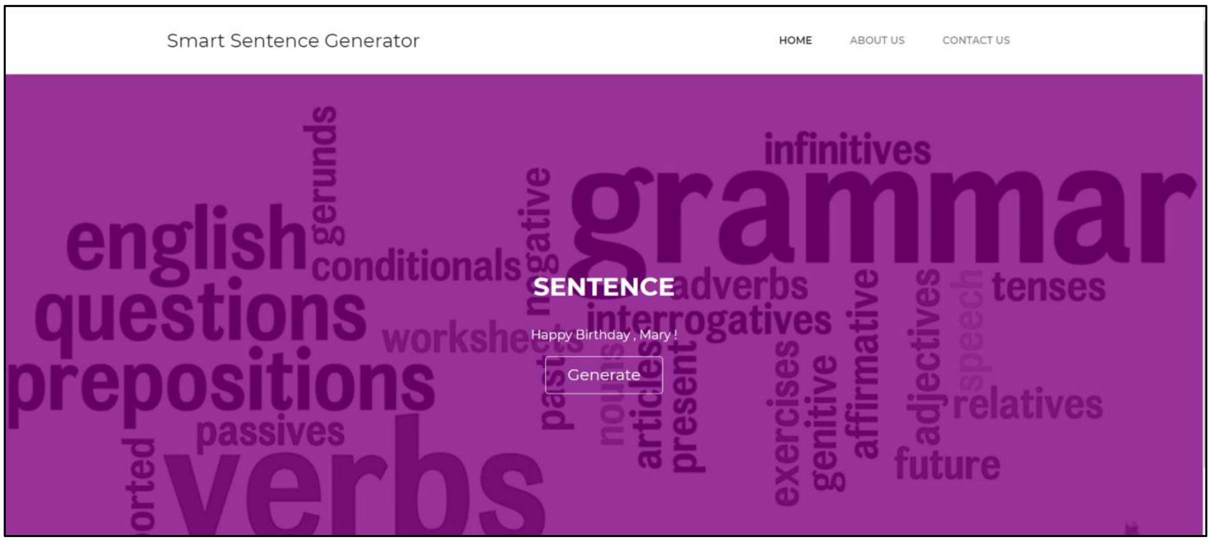
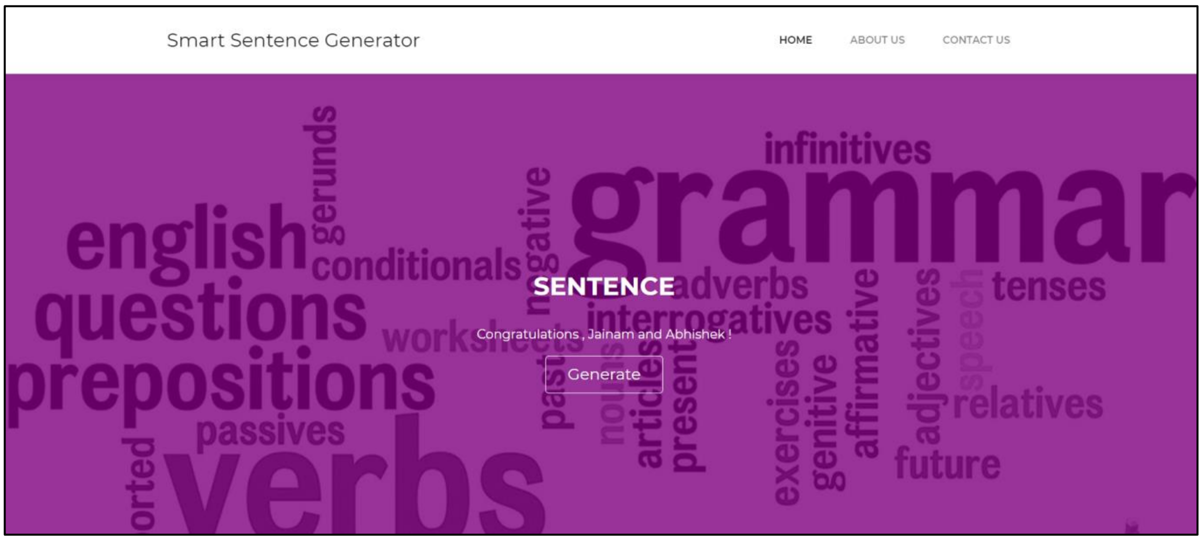


Figure 26. Exclamatory Sentence Classes for Wish



Happy Birthday , Mary !

Figure 27. Exclamatory Sentence Output for Wish Singular



Congratulations , Jainam and Abhishek !

Figure 28. Exclamatory Sentence Output for Wish Plural

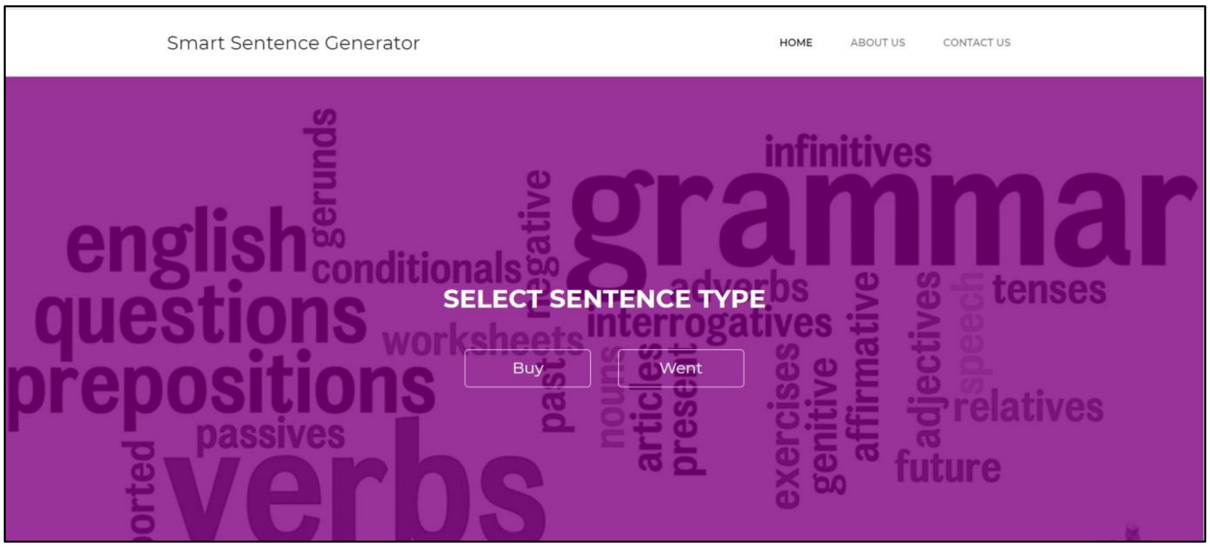
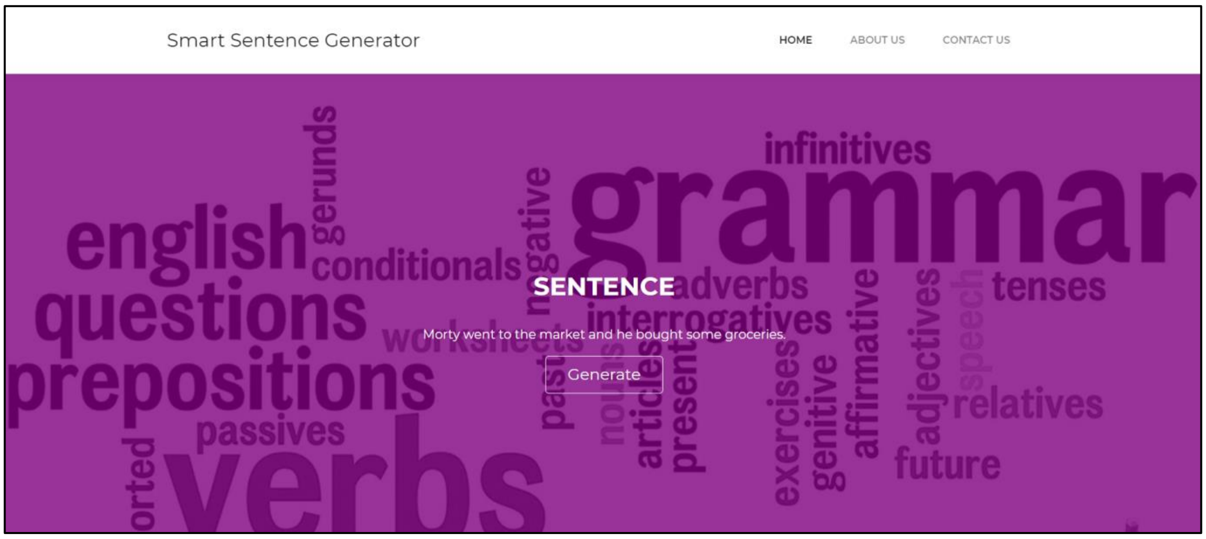


Figure 29. Compound Sentence Classes



Morty went to the market and he bought some groceries.

Figure 30. Compound Sentence Output

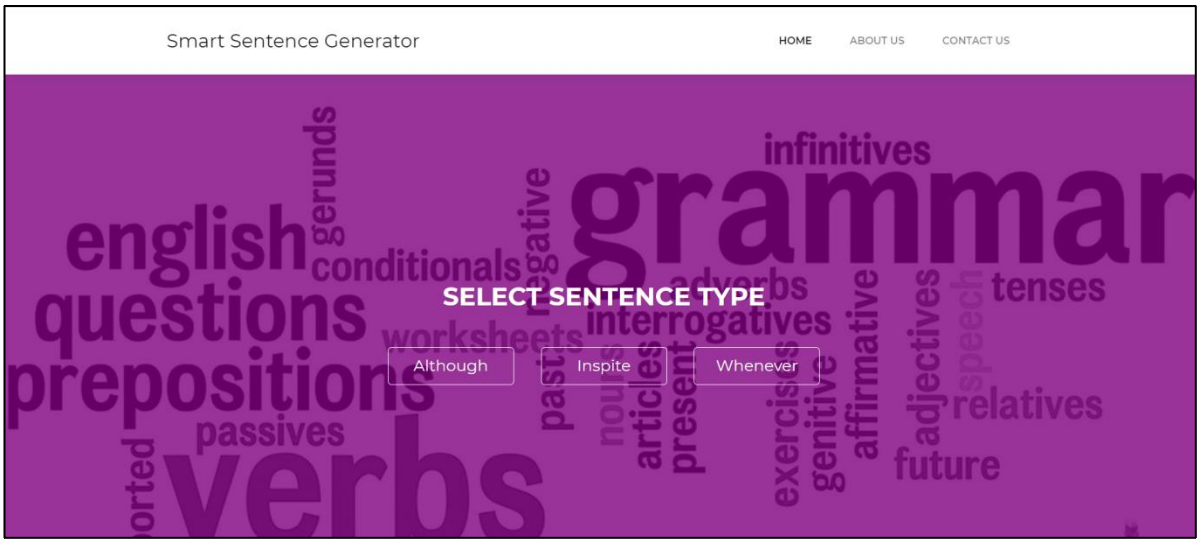
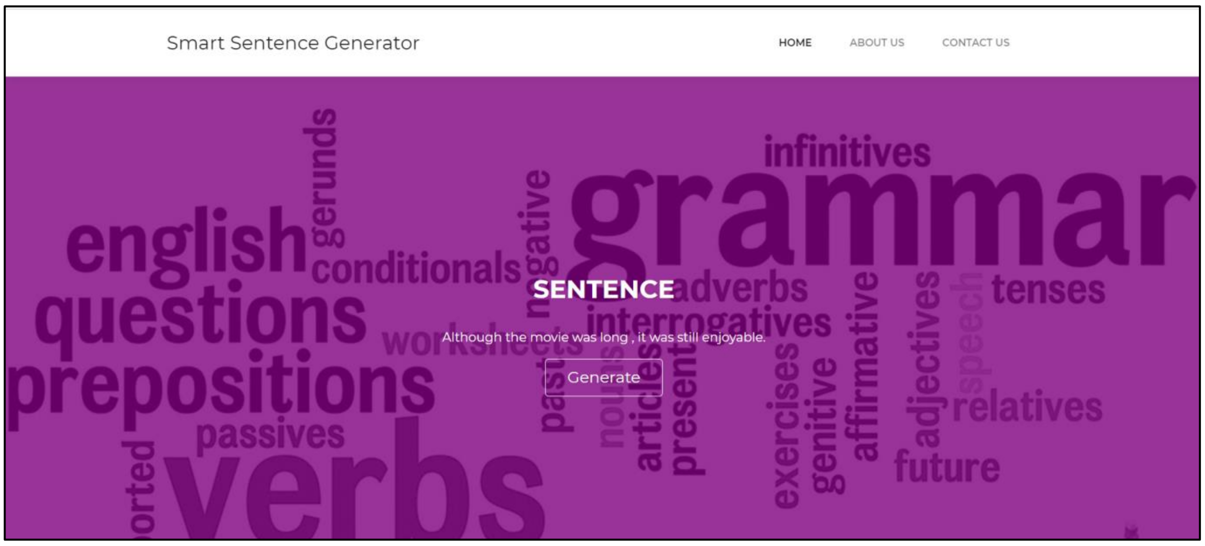
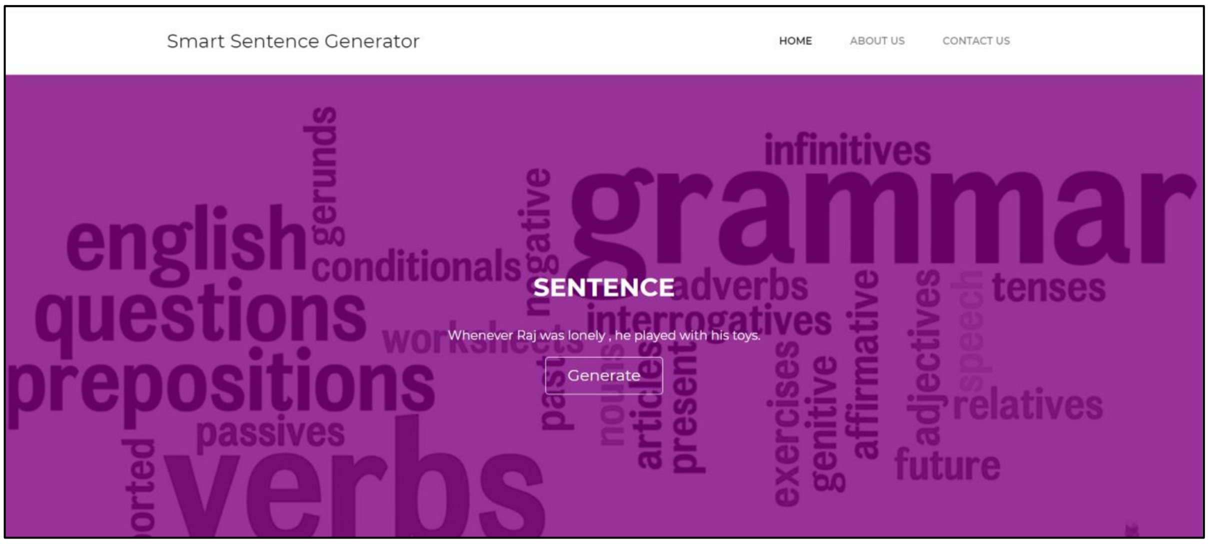


Figure 31. Complex Sentence Classes



Although the movie was long , it was still enjoyable.

Figure 32. Complex Sentence Output for Although



Whenever Raj was lonely , he played with his toys.

Figure 33. Complex Sentence Output for Whenever

6.4. Tools and Technology Used:

The following are the tools and technologies which are going to be used in this project:

1. **Django Framework** : Python Django is a free open-source web Framework useful for efficient development of web development, that focuses on writing your app without needing to reinvent the wheel.

2. **PyCharm IDE** : PyCharm is an integrated development environment (IDE), specifically designed for the Python language, developed by JetBrains. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems and supports web development with Django as well as Data Science with Anaconda. It is cross-platform, with Windows, macOS and Linux versions.

3. **Web Develpment** : The GUI for the user is developed using web technologies like HTML, CSS, JavaScript, BootStrap for a beautifully designed web interface.

4. **Code** : Python, Element Tree, NLP NodeBox Library.

CHAPTER 7

TESTING AND RESULTS

7. Testing and Results:

7.1 Test Plan:

A test plan is a general document for the entire project that defines the scope, approach to be taken, and the schedule of testing as well as identifies the test items for the entire testing process and the personnel responsible for the different activities of testing.

The test planning can be done well before the actual testing commences and can be done in parallel with the design and coding phase. The inputs for test plan are:

1. Project Plan
2. Requirement Document
3. System Design Document

The project plan is needed to make sure that the test plan can be consistent with the overall plan for the project and testing schedule matches that of the project plan.

The requirement documents and the design documents are the basic documents for selecting the test unit and deciding the approaches to be used during testing. A test plan should contain the following:

1. Test Unit:

Set of one or more modules, together with associated data, that are from single program and that are the object of testing.

2. Features to be Tested:

- Ability to access the system at all times.
- The files and documents should be accessible by the program whenever required.
- The system must generate accurate sentences every time it is run.
- The system should be able to give results on the screen in least time as possible.

3. Approaches for testing:

- Unit testing
- Integration Testing

7.2 Test Cases:

The application was tested against the following test cases which gave the following results:

Table 2. Test Cases

Test Case ID	Case Description	Expected Result	Actual Result	Pass/Fail
1	Clicking on Learn More Button	Go to Home Page	Go to Home Page	Pass.
2	Clicking on simple button	Go to variants of simple page	Go to variants of simple page	Pass.
3	Clicking on any variant of simple sentence	Go to the classes of simple sentences page	Go to the classes of simple sentences page	Pass.
4	Clicking on any class of simple sentence class	Generate sentence	Generate sentence	Pass.
5	Clicking on interrogative button	Go to variants of interrogative page	Go to variants of interrogative page	Pass.
6	Clicking on any variant of interrogative sentence	Go to the type of interrogative sentences page	Go to the type of interrogative sentences page	Pass.
7	Clicking on any type of interrogative sentence	Go to the classes of interrogative sentences page	Go to the classes of interrogative sentences page	Pass.
8	Clicking on any class of the interrogative sentence	Generate Sentence	Generate Sentence	Pass.
9	Clicking on the	Go to the variants of	Go to the variants of	Pass.

	Exclamatory Button	exclamatory sentence page	exclamatory sentence page	
10	Clicking on any variant of the exclamatory sentence	Go to the class of the interrogative sentences page	Go to the class of the interrogative sentences page	Pass.
11	Clicking on any class of the exclamatory sentence	Generate Sentence	Generate Sentence	Pass.
12	Clicking on the Compound Button	Go to the classes of the compound sentence page	Go to the classes of the compound sentence page	Pass.
13	Clicking on any of the classes of the Compound Sentence	Generate Sentence	Generate Sentence	Pass.
14	Clicking on the Complex Button	Go to the variants of the complex sentences page	Go to the variants of the complex sentences page	Pass.
15	Clicking on any of the variants of the Complex Sentence	Go to the classes of the complex Sentences page	Go to the classes of the complex Sentences page	Pass.
16	Clicking on any of the classes of the Complex Sentence	Generate Sentence	Generate Sentence	Pass.
17	Clicking on the generate button	Generate a new sentence	Generate a new sentence	Pass.

18	Clicking on the home button	Go to Home Page	Go to Home Page	Pass.
19	Clicking the back button	Go one page to the back	Go one page to the back	Pass.
20	Clicking on the about us button	Go to the about us page	Go to the about us page	Pass.

7.3 Testing Methods Used:

System testing is a critical phase implementation. Testing of the system involves hardware devise and debugging of the computer programs and testing information processing procedures. Testing can be done with text data, which attempts to simulate all possible conditions that may arise during processing. The testing methods adopted during the testing of the system are:

1. Unit testing:

Unit testing is a method by which individual units of source code, sets of one or more program modules together with associated control data, usage procedures, and operating procedures, are tested to determine if they are fit for use. In our tool, we considered each module as one unit and tested these units with help of test cases and test plan developed. Unit testing was carried out on each module and on every function within the module. Output of each unit was assessed for accuracy and if found incorrect, appropriate corrections were made.

2. Integration testing:

Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. The modules of our tool were integrated together in order to verify that they provide the required functionalities appropriately. The various modules were tested together to check for their accuracy and compatibility.

3. System Testing:

System testing of software is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. In this testing, we tested the system as whole to ensure that it provides the appropriate output as stated in the requirements. Overall performance of the system was also tested simultaneously.

7.4 Experimental Results and Conclusions:

The accuracy of the system was good and as expected. The sentences are generated as per the inputs given to the system. From the figures given below, it is evident that the system performs close to what was expected. The primary goal of the system was to generate syntactically and semantically accurate sentences that helped students to learn them. The test case was performed for generating a simple sentences based on user’s inputs which output a simple sentence with multiple objects in it. Other sentences are tested as well for their performance. All these are tested which gives accurate results as required.

CHAPTER 8

CONCLUSION

8. Conclusion:

The sentence generation algorithms and the models that are available do not give the complete output. Also the output is not grammatically correct and the modes are meaningless.

The proposed system guarantees all such problems to be eliminated and a proper sentence is generated by the algorithm that we have devised. The templates that we have made are self-developed and are the main constituent of the sentence generation algorithm. The final output is as we desired and accurate.

Thus, we have developed a system which can generate sentences complete and meaningful with no grammatical errors whatsoever.

Also, the datasets that we have made can me modified and changed as per requirement for further development to try to generate more variety of sentences and complex as well as compound sentences.

CHAPTER 9

FUTURE SCOPE

9. Future Scope:

The future scope of the proposed system is very interesting. This sentence generation system can be made to generate sentences to help students learn some kinds of sentences which are necessary for basic English. What we can try to aim is to create a full-fledged Intelligent Tutoring System (ITS) with a validator as well like the ones used in the competitive exams like GRE and GMAT to grade the students.

The learning process can be made easier for the students to learn as well as the teachers to teach. For further implementation, the whole learning process can be made automatic. The whole system can me made a Machine Learning Project and the system can be used to train itself from the datasets and then be used to train the students without the help of a tutor. Necessary guidelines and tutorials can be provided to the students to help them navigate and use the system to make the most benefit out of it.

With the development of such a system, we can overcome all the drawbacks specified in the challenges section and will be the first ever system to be developed to be an automatic ITS for the students to learn the generation of various kind of sentences.

CHAPTER 10

PUBLICATIONS

10. Publications:

1. DJASCII – Smart Sentence Generator.
2. SRIJAN 2020- Submitted.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” 24th May, 2019.

[2] Anis Ernawati, “An Analysis Types of Sentences used by the Students’ Essay Writing at the 3rd Semester of Iain Tulungagung in the 2013/2014 Academic Years.” 2014.

[3] Chiyu Wang, Hong Li, Xun Hu and Jiale Zhou, “Attention Model Using Full-Time Information for Sentences Emotion Classification,” in IEEE 8th Joint International Information Technology and Artificial Intelligence Conference(ITAIC), May 2019.

[4] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, et al, “XLNet: Generalized Autoregressive Pretraining for Language Understanding,” 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

[5] M. Chandhana Surabhi, “Natural Language Processing Future”, IEEE International Conference, 2013.

[6] Phuong Le-Hong and Anh-Coung Le, “A Comparative Study of Neural Network Models for Sentence Classification,” October 2018.

[7] Dr. Abhijit R. Joshi, “Building an Intelligent Environment for Learning Indian Language: Marathi ”, April 2012.

[8] Josephine E. Petralba, “An Extracted Database Content From WordNet for Natural Language Processing and Word Games.” International Conference on Asian Language Processing, 2014.

[9] Jeffrey Pennington, Richard Socher and Christopher D. Manning, “GloVe: Global Vectors for Word Representation”.