

**TRIBHUVAN UNIVERSITY**

**INSTITUTE OF ENGINEERING**

**THAPATHALI CAMPUS**

**A Project Proposal**

**On**

**Plagiarism Detection using Natural Language Processing through Semantic Analysis**

**Submitted By:**

Aashish Kumar Sah (THA076BCT001)

Arun Subedi (THA076BCT010)

Grishma Raj Khanal (THA076BCT016)

Nishant Uprety (THA076BCT023)

**Submitted To:**

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

July, 2021

# ABSTRACT

Plagiarism is the way of copying someone’s works entirely or modifying it by using similar word and claiming it as their own. It has been a major problem in academics and literature field from the past. It is disrespectful to use someone’s works as their own and you may face legal charges for it in some countries. So, in this project we are going to develop the web application which detects the plagiarism between many suspicious documents and the original document using natural language processing (NLP). We are going to preprocessed the suspicious document thoroughly and split it into many segments, sentences, tokens and then use different mathematical formula to assign the certain value to them. These values are used as inputs in trained model to detect plagiarism. We are expecting to get the plagiarism percentage of suspicious document and classify them as their types.

*Keywords: Plagiarism, Naïve Bayes, NLP, Jaccard Similarity Index*

# TABLE OF CONTENTS

[ABSTRACT i](#_Toc122047558)

[TABLE OF CONTENTS ii](#_Toc122047559)

[List of Figures iii](#_Toc122047560)

[List of Tables v](#_Toc122047561)

[List of Abbreviations vi](#_Toc122047562)

[1. INTRODUCTION 1](#_Toc122047563)

[1.1 Background Introduction 1](#_Toc122047564)

[1.2 Motivation 1](#_Toc122047565)

[1.3 Problem Definition 2](#_Toc122047566)

[1.4 Objectives 2](#_Toc122047567)

[2. LITERATURE REVIEW 3](#_Toc122047568)

[2.1 Classification of Plagiarism Detection 3](#_Toc122047569)

[2.2 Limitation of Non NPL based Plagiarism Detection 3](#_Toc122047570)

[2.3 Natural Language Processing Approach 3](#_Toc122047571)

[2.3.1 Plagiarism Detection through Lexical Approach 3](#_Toc122047572)

[2.3.2 String Detection Technique 4](#_Toc122047573)

[2.3.3 Karp- Rabin and String-Matching Algorithm 4](#_Toc122047574)

[2.4 Machine Learning combined with NLP 4](#_Toc122047575)

[3. METHODOLOGY 5](#_Toc122047576)

[3.1 NLP Approach 5](#_Toc122047577)

[3.1.1 Corpora of Plagiarized Texts 5](#_Toc122047578)

[3.1.2 Classification of Suspicious Documents 5](#_Toc122047579)

[3.2 Procedural Sequence 6](#_Toc122047580)

[3.2.1 Data Collection and Preprocessing 6](#_Toc122047581)

[3.2.2 Model Building and Machine Training 6](#_Toc122047582)

[3.3 System Architecture 8](#_Toc122047583)

[3.4 FLOWCHART OF PROPOSED SYSTEM 9](#_Toc122047584)

[3.5 Instrumentation Tools. 10](#_Toc122047585)

[3.5.1 Programming Language 10](#_Toc122047586)

[5. SCOPE AND APPLICATIONS 10](#_Toc122047587)

[6. TIME ESTIMATION 10](#_Toc122047588)

[7. FEASIBILITY ANALYSIS 11](#_Toc122047589)

[7.1 Economic feasibility 11](#_Toc122047590)

[7.2 Technical feasibility 12](#_Toc122047591)

[7.3 Operational feasibility 12](#_Toc122047592)

[References 12](#_Toc122047593)

# List of Figures

[Figure 2‑1:Ceaser Cipher technique 4](#_Toc77951806)

[Figure 3‑1:Basic layout of application 6](#_Toc77951807)

[Figure 3‑2:Login Class model 7](#_Toc77951808)

[Figure 3‑3:Registration Class model 8](#_Toc77951809)

[Figure 3‑4:Userinfo Class model 9](#_Toc77951810)

[Figure 4‑1:Block diagram for first time registration 12](#_Toc77951811)

[Figure 4‑2:Block diagram for Login 13](#_Toc77951812)

[Figure 4‑3:Conversion of encrypted password to decrypted and vice versa 14](#_Toc77951813)

[Figure 4‑4:Block diagram for Updating User data 15](#_Toc77951814)

[Figure 4‑5:Block Diagram of control flow between different layers of system 16](#_Toc77951815)

[Figure 4‑6:Algorithm used for Encryption and Decryption 17](#_Toc77951816)

# List of Tables

[Table 3‑1:Classes used in the program 5](#_Toc78014126)

[Table 6‑1:Gnatt chart 18](#_Toc78014127)

# List of Abbreviations

|  |  |
| --- | --- |
| IOE | Institute of Engineering |
| SFML | Simple and Fast Multimedia Library |
| MIT | Massachusetts Institute of Technology |
| UI | User Interface |
| GCC | GNU Compiler Collection |
| API | Application Programming Interface |
| IDE | Integrated Development Environment |
| I/O | Input/output |
| GUI | Graphical User Interface |
| VoIP | Voice Over Internet Protocol |
| HTTP | Hypertext Transfer Protocol |
| URL | Uniform Resource Locator |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# INTRODUCTION

## Background Introduction

The growth of online services has vastly increased the number of passwords a user has to remember while accessing various online accounts. Although it is essential to create unique and strong password for each account, remembering all those passwords is becoming burden for users. A password manager is a " computer program that allows user to store, generate and manage their passwords for local and online services ".

A password manager simply assists in generating and retrieving the login information of various accounts and automatically enter them into the forms. The automatic form filling feature fills the login information for a particular URL whenever it loads resulting fewer manual errors. As password manager can identify the right URL for a particular login ID and password pair automatically, they are capable of protecting credentials from phishing sites.

Nowadays most of the browser like chrome, safari has added the password manager that can store and generate random password for us. These types of password manager use yet another password commonly known as master password to protect our passwords. If we use Google's chrome browser to store our password and share them across devices, our password will be stored by Google and protected by password for our Google account. Whereas Apple's iCloud keychain relies primarily on our device password and unlocking feature to protect the data on regular basis.

## Motivation

Communicating and working remotely has become the regular routine for us, so it is major necessity to have a strong password for online accounts. Using multiple accounts makes us difficult to memorize all login info. It sometimes can become time consuming to login through different websites and application incase user mistypes or forgets his password. After being troubled for a long time our team members had always a wish to find some easy and quick method to access the login data for every account. Herby realizing the priorities with user's data this project is brought in creation so as to facilitate anyone with simpler method to login different websites without wasting any time.

Many people even still in 2021 are found to follow terrible password practices like using ‘password’ or 12345678’ as their password, either by deliberately accepting greater risk for the sake of convenience. To solve this problem, we aim to add a feature to a basic password manager software which generates a random but strong passwords and passphrases for the users.

## Problem Definition

There are two main problems here. First is to generate a strong passwords/passphrase for different online accounts and secondly to store these passwords safely retrievable only by the owner.

The project aims to generate, store and retrieve the login passwords using encryption and decryption. We propose to develop an efficient fast and easy to use software which also provides user with a random but strong and secure passwords and passphrases.

## Objectives

The main objectives of our project are listed below

* To design and develop a clean and easy to use password managing software.
* To develop a program that also generates random but strong passwords and passphrases for user.

# LITERATURE REVIEW

## Classification of Plagiarism Detection

Plagiarism detection is classified on the basis of detection methodology, number of documents that the metrics can process and complexity of metrics. More complex the metrics are, the more processing power is required and takes up a tremendous amount of power and resources to perform the detection. This is not suitable for users with personal computers.

## Limitation of Non NPL based Plagiarism Detection

The initial methods of plagiarism detection were mostly non-NPL based. These methods included relative frequency models, which count the occurrences of matches and word pair metrics, which calculate sequences in document pairs. However, the initial methods of plagiarism detection was limited only to a superficial level.

However, the initial approaches of the plagiarism detection model were only able to detect the simple copy/paste and paraphrasing or word insertions makes the suspicious documents harder to identify using the traditional tools based on fingerprinting. It is pointed out that it would be desirable to use NLP techniques for plagiarism and could yield better accuracies through the detection of paraphrased text.

## Natural Language Processing Approach

### Plagiarism Detection through Lexical Approach

Another plagiarism detection techniques is lexical approach which focuses on using the lexical features of the text or document and operate at the word level of the document in order to trace the plagiarism scenario from the suspicious documents. This approach relies upon tokenization, lowercasing, punctuation removal and stemming. This approach can also be merged with Natural Language Processing which increases the accuracy of the models.

### String Detection Technique

String Detection Techniques is also one of the techniques of plagiarism detection where N-gram based comparisons either character N-grams or word N-grams fall into this category. All the systems based on N-grams were effective for detecting plagiarism cases with simple copy-paste and small random shuffling cases. However, the efficiency dropped as plagiarism complexity increased. Thus, N-gram based models were found to be less effective when it comes to complex obfuscation. But, the exhibition of good precision shows its potential to be combined and used in hybrid approaches.

### Karp- Rabin and String-Matching Algorithm

The Karp-Rabin algorithm uses fingerprints to find occurrences of one string into another string. It reduces time of comparison of two sequences by assigning hash value to each string and word. This algorithm can also deal with multiple pattern matching. String matching algorithm computers similar strings and performs character by character matching. Karp-Robin’s problem is that it assigns similar hash value are same but strings are non-similar. For removing this drawback and improving efficiency, we use string matching algorithm along with Karp-Rabin algorithm as it keeps string in arrays and checks it character by character in the array. The proposed system has predicted to give precision value up to 85% and above as recall value.

## Machine Learning combined with NLP

The machine learning integrated NLP model uses Naives Bayes and Support Vector Machine (SVM). Naive Bayes is a learning algorithm that represents each instance as the conjunction of its attribute’s values. It classifies the data by selecting the class that has the highest chance value, assuming that attributes are independent to each other. SVM is also a learning algorithm that analyzes data and recognizes patterns. The analyzed data creates a hyper plane separator to divide the data based on each class.

Both NB and SVM allow the choice of kernel function for each and are sensitive to parameter optimization. SVM takes a long time to train large datasets. It requires hyperparameter tuning which is not trivial and takes time while NB is faster as it depends on conditional probability and is easier to implement and evaluate.

# METHODOLOGY

## NLP Approach

Natural Languages Processing refers to processing of human understandable language by the machines. NLP involves ability to understand text and spoken words by the computers.

### Corpora of Plagiarized Texts

In this project we mainly focus on detection of external plagiarism i.e., when both source text and suspicious text is present. A date set from Kaggle which provides us with the samples of plagiarized texts. These data sets will be used to train the model to identify different levels of plagiarism in the texts

### Classification of Suspicious Documents

* Near Copy: The Suspicious text has been completely copied and pasted from the original text document
* Light Revision: Grammatical and other minor alternation of the text from the original text.
* Heavy Revision: Paraphrasing of the sentences from the original document i.e., plagiarism of the idea.
* Non-Plagiarism: Suspicious text has been written from own view and knowledge of the topic.

## Procedural Sequence

### Data Collection and Preprocessing

The corpus of text with fake plagiarized texts has been classified into four categories of plagiarism in the data set provided from Kaggle. The general preprocessing techniques used on the text are

* Parsing Paragraphs into words: The text in the document is split into sentences and then into words and thereby allowing the individual words to be treated as vectors quantity or tokens
* Removal of Punctuation: The unnecessary punctuation symbols which give no meaning to the texts are removed.
* Parts of Speech tagging: Different tokens are assigned with their own grammar tags according to the parts of speech like noun, verb, adverbs etc.
* Stemming and Lemmatization: The individual words are transformed into their stems in order to generalize the comparison analysis. The different forms of words are normalized into their general form. Example riding, rides, ridden, etc. are normalized into ride. In lemmatization the words like better best are normalized into good.
* Similarity Adjustment:The semantic analysis of the text is done by measuring the similarities of the words in the text and looking for the synonyms for the words considering the POS tag

### Model Building and Machine Training

The text in the corpus dataset is divided into near copy, light revision, heavy revision and non-plagiarism categories. Plagiarism detection is based on lexical analysis of the tokens made from parsing of the text. The tokens of the suspicious documents are compared with the tokens of original document. Jaccard Similarity Coefficient is used to compare the tokens of the suspicious and the original documents. The matching process takes the normalized word with the same POS. The value of each text will be compared to the value of the original text and system will be classified into the four categories. The greater the value of the coefficient will define the greater in similarity between the suspicious document and the original document.

J (A, B) =

In Machine Learning, the index is able to quantify the similarities between computer’s identified texts and the training data sets. All the corpus of the data are to be used as testing data for the machine training. Naive Bayes classifier based on the Bayes’ theorem is to be used to generate a probabilistic model of the data.

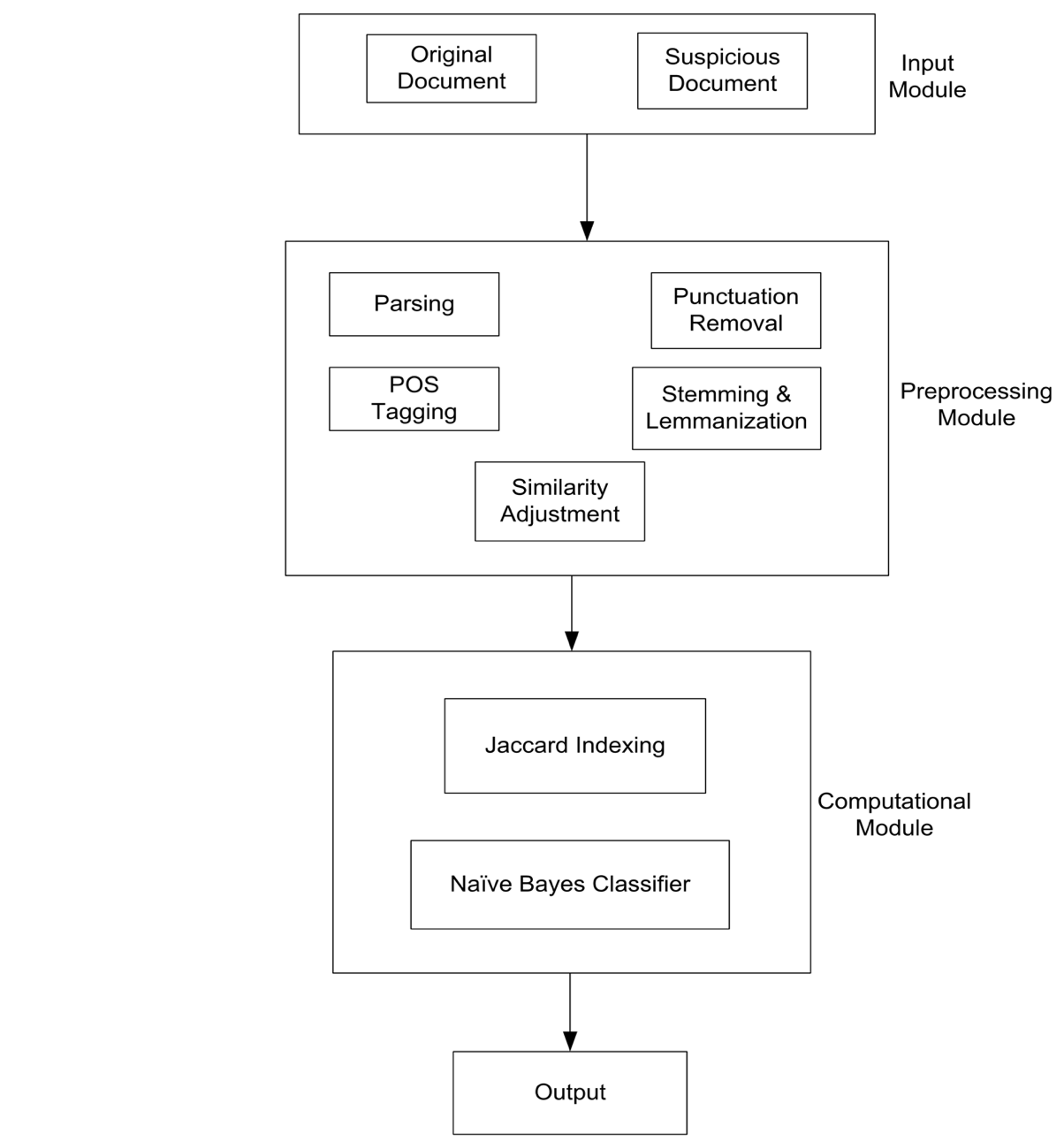
Given a number of features (f1,…, fn) for a number of training examples pre classified into the four level of plagiarism, the aim of the classifiers to learn a model for classifying similar but new cases.

Let C represent the classification variable and c be the value of C then with n individual attributes

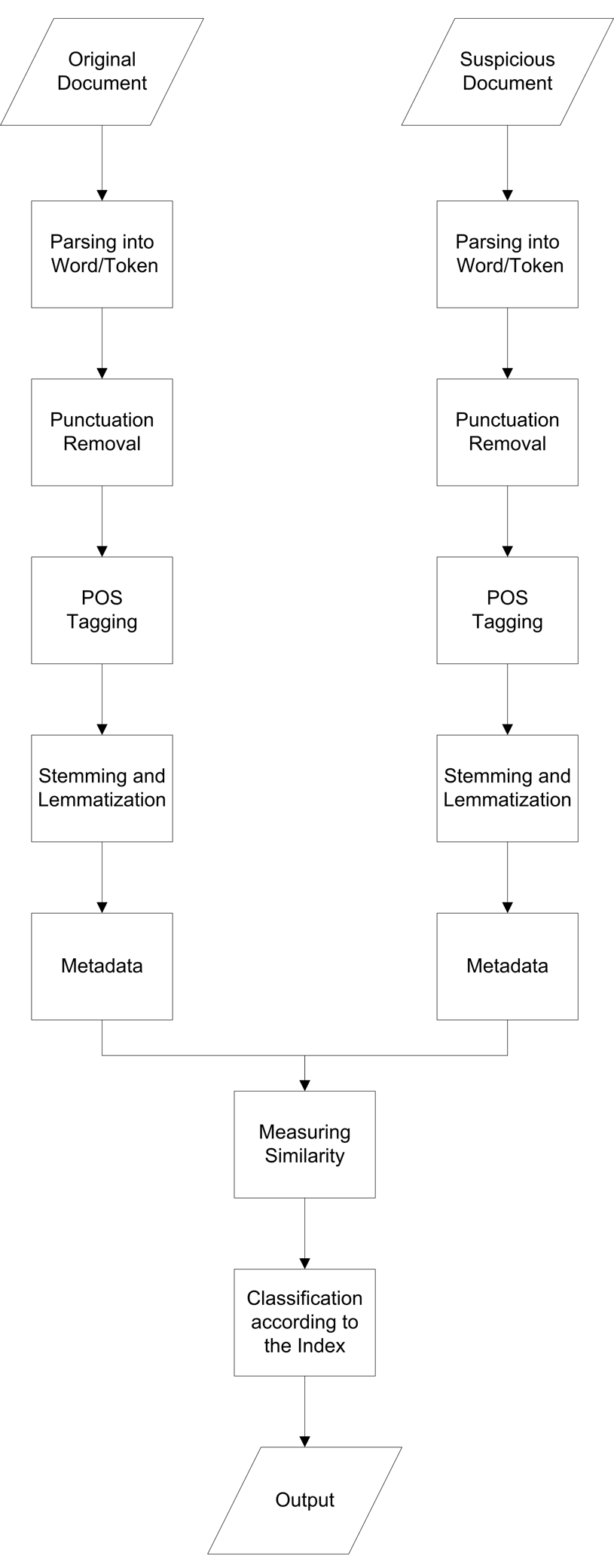
p(E|c) = p(x1,x2,…,xn|c) =

## System Architecture

The figure below describes the architecture of the proposed system. The application will provide user to insert the documents into the system one original and other suspicious ones. Then preprocesses both original and suspicious documents. Next the application will estimate the similarities between the documents provided and provide user with the plagiarism status classified into four categories.



## FLOWCHART OF PROPOSED SYSTEM



## Instrumentation Tools.

### Programming Language

* **Python:** Python Programming Language is mainly used to implement Natural Language Processing for better analyzing and machine training for our model. It provides us with various packages to implement simple mathematical formulae like numpy , matplotlib to complex data science packages like Scikit-learn and NLKT packages.
* **NKLT:** Natural Language Toolkit is a toolkit build for working with NLP in Python. It provides us various text processing libraries with a lot of test datasets.
* **Scikit-learn:** Scikit-learn is a python module for implementing machine learning. It provides us with efficient version of large number of algorithms.
* **Pandas:** Pandas is a python library used to data analyzing and manipulation. It helps us to draw conclusion from big sets of datas

# 5. SCOPE AND APPLICATIONS

Our project aims for:

* The password manager will do the work of creating unique and complicated passwords needed to help protect the online accounts.
* The program helps to keep tracks of all the passwords created beforehand for different websites.

# TIME ESTIMATION

A Gantt chart is a graphical representation of a project that shows each activity task as a horizontal bar whose length is proportional to its time for completion. A Gantt chart for the project deliverables within time frame. This project Gantt chart is shown below:

Table 6‑1:Gnatt chart

0(06/07/2021 2 4 6 8

# 7. FEASIBILITY ANALYSIS

The development of software is disturbed mainly by scarcity of hardware resources, software resources and time constrain. So, it is necessary for us to think about future outcomes while checking the feasibility of the program at the very beginning of the development of the program. The three considerations involved in the feasibility analysis are:

## 7.1 Economic feasibility

This procedure is to determine the benefits and the expected savings from the developed program. Here, we compare the cost of production which includes the cost increased or decreased due to the use of external resources as well as the earnings that may result in benefit for the group. If anything is not found as the plan, then we edit the proposed system and make sure that the economic feasibility of the program is maintained.

As for now, in our project we do not expect any feasibility costs to be spent on as in this program we have only used the open source resources as they are easily available.

## 7.2 Technical feasibility

Technical feasibility focuses on the existing resources such as hardware, software. It also focuses on the extent to which the available resources can be used and if the budget is found as a serious restriction in the completion of the project then the project is judged to be not feasible. Here in the case of our project we have used CODEBLOCKS to write the code and used the Windows OS as a platform whereas we have only used the open resources available. We need members to have knowledge of database, structure and sfml library in C++.

## 7.3 Operational feasibility

People have been known to like the programs that take less space, processing time, that is easy to configure/install and that is entertaining as well as does not stress them out. So, keeping that in mind here in our project, we have made our program easy to configure as well as easily executable regarding the comfort of future clients. The technical background required for the sensors of the game or the I/O are the basic devices that a computer system needs to function properly. So, regarding the operational feasibility the is feasible if the user has basic knowledge of Code block and the program.

# References

|  |  |
| --- | --- |
| [1] | A. Burgher, "A short history of passwords," *We live security,* 2017. |
| [2] | J. Andress, "Cryptography," in *The Basics of Information Security*, 2014. |
| [3] | N. Maccha, "Network Security and cryptography," Nikhil Maccha, 2020. |
| [4] | S. Jain, "Geeksforgeeks," [Online]. Available: https://www.geeksforgeeks.org/caesar-cipher-in-cryptography/. [Accessed July 2021]. |