A MINI-PROJECT REPORT ON

MORSE SUBSTITUTION CIPHER

Submitted in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

In

Computer Science and Engineering

By

K.Yogeswari (18A81A05K1)

Under the Esteemed Supervision of

Ms. M. Anantha Lakshmi, M.Tech, Asst. Professor

Department of Computer Science and Engineering (Accredited by N.B.A.)
SRI VASAVI ENGINEERING COLLEGE(Autonomous)
(Affiliated to JNTUK, Kakinada)
Pedatadepalli, Tadepalligudem-534101, A.P 2020-21

SRI VASAVI ENGINEERING COLLEGE (Autonomous)

Department Of Computer Science and Engineering

Pedatadepalli, Tadepalligudem

Certificate

This is to certify that the Project Report entitled "MORSE SUBSTITUTION CIPHER" submitted by **K. Yogeswari (18A81A05K1)** for the award of the degree of Bachelor of Technology in the Department of Computer Science and Engineering during the academic year 2020-2021.

Name of Project Guide

Ms. M. Anantha Lakshmi M.Tech,
Assistant Professor.

Head of the Department

Dr. D Jaya Kumari M.Tech.,Ph.D..
Professor & HOD.

External Examiner

DECLARATION

We hereby declare that the project report entitled “MORSE SUBSTITUTION CIPHER” submitted by us to Sri Vasavi Engineering College(Autonomous), Tadepalligudem, affiliated to JNTUK Kakinada in partial fulfilment of the requirement for the award of the degree of B.Tech in Computer Science and Engineering is a record of Bonafide project work carried out by us under the guidance of Ms. M. Anantha Lakshmi , M.Tech, Asst.Professor. We further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree in this institute or any other institute or University.

Project Associate

K.Yogeswari (18A81A05K1)

ACKNOWLEDGEMENT

First and foremost, we sincerely salute to our esteemed institute **SRI VASAVI ENGINEERING COLLEGE,** for giving us this golden opportunity to fulfill our warm dream to become an engineer.

Our sincere gratitude to our project guide **Ms. M. Anantha Lakshmi**, Assistant Professor, Department of Computer Science and Engineering**,** for her timely cooperation and valuable suggestions while carrying out this project.

We express our sincere thanks and heartful gratitude to **Dr. D. Jaya Kumari**, Professor & Head of the Department of Computer Science and Engineering, for permitting us to do our project.

We express our sincere thanks and heartful gratitude to **Dr. G.V.N.S.R. Ratnakara Rao**, Principal, for providing a favourable environment and supporting us during the development of this project.

Our special thanks to the management and all the teaching and non-teaching staff members, Department of Computer Science and Engineering, for their support and cooperation in various ways during our project work. It is our pleasure to acknowledge the help of all those respected individuals.

We would like to express our gratitude to our parents, friends who helped to complete this project.

Project Associate

K.Yogeswari (18A81A05K1)

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Morse Substitution Cipher is a process of transmitting text information as a series of dits and dahs. Here dits refer to dots and dahs refer to dash. This was mainly used in World war because it greatly improved the speed of communication. Naval ships were able to communicate with their bases and provide the critical information to each other. This is also used for the long-distance communication. It is the base for the morse code to transmit or receive.

And the implementation of a Morse Substitution Cipher will be done by using Python. The translator makes uses of a programming language as a key and simple algorithm of changing characters to dots and dashes. It is a character encoding and decoding scheme. The algorithm is very simple. Every character in the English language is substituted by a series of ‘dots’ and ‘dashes’ and vice versa. This system was designed to transmit message securely to long distance. By using this system this will overcome the security problems.

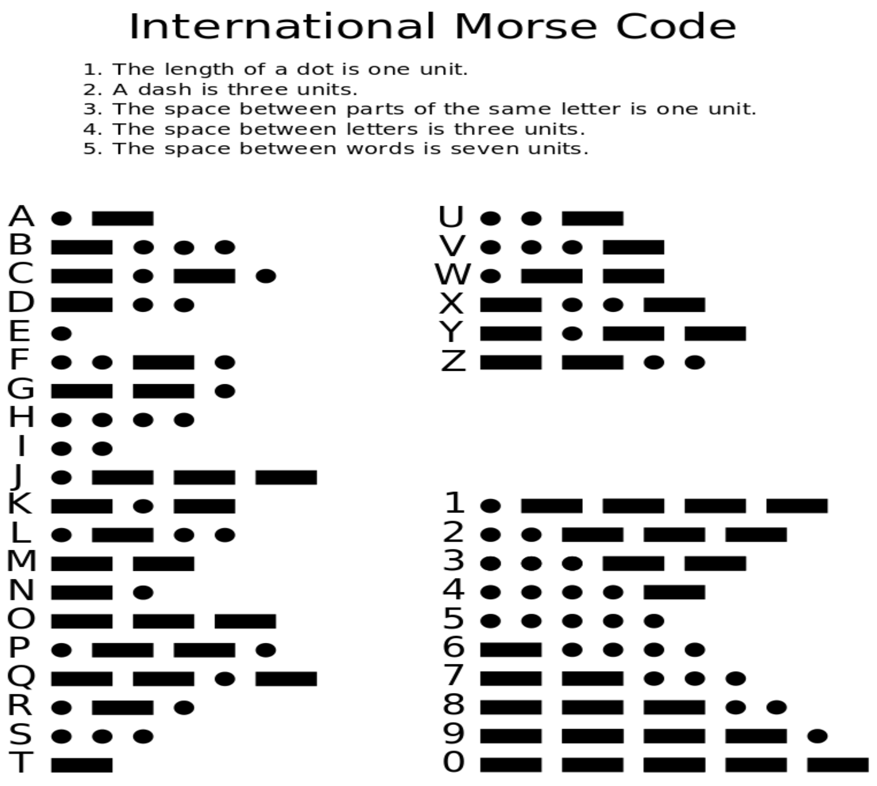
CHAPTER 1

INTRODUCTION

1.1 Introduction about Morse Code

Morse code is a method used in telecommunication to encode text characters as standardized sequences of two different signal durations, called *dots* and *dashes*, or *dits* and *dahs*. Morse code is named after Samuel Morse, one of the inventors of the telegraph.

International Morse Code encodes the 26 Latin letters A through Z, one non-Latin letter, the Arabic numerals, and a small set of punctuation and procedural signals (prosigns). There is no distinction between upper and lower case letters.[1] Each Morse code symbol is formed by a sequence of *dits* and *dahs*. The *dit* duration is the basic unit of time measurement in Morse code transmission. The duration of a *dah* is three times the duration of a *dit*. Each *dit* or *dah* within an encoded character is followed by a period of signal absence, called a *space*, equal to the *dit* duration. The letters of a word are separated by a space of duration equal to three *dits*, and words are separated by a space equal to seven *dits*.


International Morse Code

1. The length of a dot is one unit.
2. A dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units.

Fig 1.1 International Morse Code

Morse code can be memorized and sent in a form perceptible to the human senses, e.g. via sound waves or visible light, such that it can be directly interpreted by persons trained in the skill. Morse code is usually transmitted by on-off keying of an information-carrying medium such as electric current, radio waves, visible light, or sound waves

Since many natural languages use more than the 26 letters of the Latin alphabet, Morse alphabets have been developed for those languages, largely by transliteration of existing codes.

To increase the efficiency of encoding, Morse code was designed so that the length of each symbol is approximately inverse to the frequency of occurrence of the character that it represents in text of the English language. Thus the most common letter in English, the letter **E**, has the shortest code: a single *dit*. Because the Morse code elements are specified by proportion rather than specific time durations, the code is usually transmitted at the highest rate that the receiver is capable of decoding. Morse code transmission rate (*speed*) is specified in *groups per minute*, commonly referred to as *words per minute*.

1.2 Development

Early in the nineteenth century, European experimenters made progress with electrical signaling systems, using a variety of techniques including static electricity and electricity from Voltaic piles producing electrochemical and electromagnetic changes. These experimental designs were precursors to practical telegraphic applications.

Following the discovery of electromagnetism by Hans Christian Orsted in 1820 and the invention of the electromagnet by William Sturgeon in 1824, there were developments in electromagnetic telegraphy in Europe and America. Pulses of electric current were sent along wires to control an electromagnet in the receiving instrument.

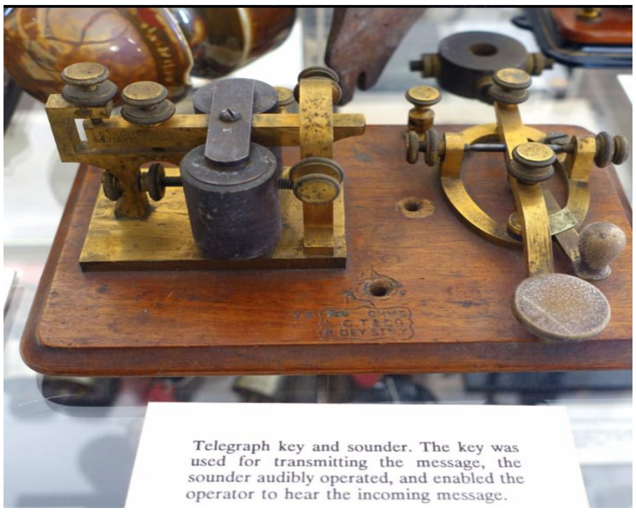
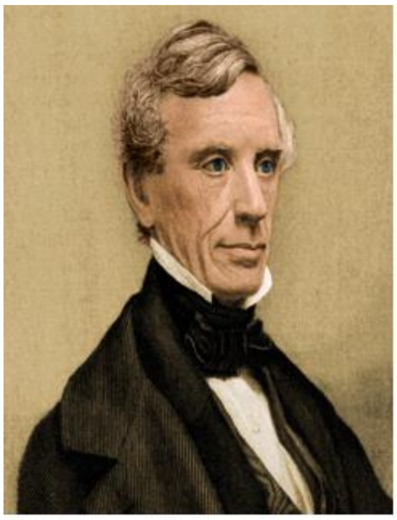

Telegraph key and sounder. The key was used for transmitting the message, the sounder audibly operated, and enabled the operator to hear the incoming message.

Fig 1.2 Telegraph

William Cooke and Charles Wheatstone in Britain developed an electrical telegraph that used electromagnets in its receivers. They obtained an English patent in June 1837 and demonstrated it on the London and Birmingham Railway, making it the first commercial telegraph. Carl Friedrich Gauss and Wilhelm Eduard Weber (1833) as well as Carl August von Steinheil (1837) used codes with varying word lengths for their telegraph systems. In 1841, Cooke and Wheatstone built a telegraph that printed the letters from a wheel of typefaces struck by a hammer.

1.3 History

The American artist Samuel Morse, the American physicist Joseph Henry, and mechanical engineer Alfred Vail developed an electrical telegraph system. It needed a method to transmit natural language using only electrical pulses and the silence between them. Around 1837, Morse, therefore developed an early forerunner to the modern International Morse code.


Samuel Morse

Alfred Vail

The Morse system for telegraphy, which was first used in about 1844, was designed to make indentations on a paper tape when electric currents were received. Morse's original telegraph receiver used a mechanical clockwork to move a paper tape. When an electrical current was received, an electromagnet engaged an armature that pushed a stylus onto the moving paper tape, making an indentation on the tape. When the current was interrupted, a spring retracted the stylus and that portion of the moving tape remained unmarked. Morse code was developed so that operators could translate the indentations marked on the paper tape into text messages.

1.4 Elements of Morse Code

- short mark, dot or *dit* (▪): "dit duration" is one time unit long
- long mark, dash or *dah* (▬): three time units long
- inter-element gap between the *dits* and *dahs* within a character: one dot duration or one unit long
- short gap (between letters): three time units long
- medium gap (between words): seven time units long

1.5 Transmission

Morse code is transmitted using just two states (on and off). Morse code may be represented as a binary code, and that is what telegraph operators do when transmitting messages. Morse code sequence may be made from a combination of the following five bit-strings:

- short mark, dot or *dit* (▪): 1
- longer mark, dash or *dah* (▬): 111
- intra-character gap (between the *dits* and *dahs* within a character): 0
- short gap (between letters): 000
- medium gap (between words): 0000000

They are different ways of transmitting the Morse Code

- As radio signal.
- Sounds like car horns.
- Mechanical or visual signals(pulling a rope, flash lights)

The Speed of transmitting is

$$T = \frac{1200}{W}$$

- T stands for unit of time.
- W stands for speed in wpm(words per minutes)

CHAPTER 2

LITERATURE SURVEY

Title: Morse code translator using Python
Authors: Saikumar Kothamasu, Vaibhav Gopinath and Rrochish
Abstract:

Morse code is a process of transmitting text information as a series of on-off tones and lights or clicks .if they use a tapping device recipient can understand the message without additional decoding equipment. Morse code is represented by the form of dits and dahs. Here dits refer to dots and dahs refer to dash. Morse code used to transmit only numerals at first. After that, Alfred Vail included letters and characters. Morse code can be transmitted by using electric telegraph wire, light, and sound, through a different medium in different ways. Tap code is used by American prisoners. Morse code is used for long-distance communication. International Morse code was devised by European nations in 1851. It is the base for the morse code to transmit or receive. Morse code is a character encoding and decoding scheme.

Title: Morse Code Datasets for Machine Learning
Authors: Sourya Dey, Keith M. Chugg and Peter A. Beerel
Abstract:

They present an algorithm to generate synthetic datasets of tunable difficulty on classification of Morse code symbols for supervised machine learning problems, in particular, neural networks. The datasets are spatially one-dimensional and have a small number of input features, leading to high density of input information content. This makes them particularly challenging when implementing network complexity reduction methods. We explore how network performance is affected by deliberately adding various forms of noise and expanding the feature set and dataset size. Finally, we establish several metrics to indicate the difficulty of a dataset, and evaluate their merits. The algorithm and datasets are open-source.

CHAPTER 3

PROBLEM STUDY

3.1 Existing System

- In this existing system, they used Python programming language for translatation. And its web based application.
- Only Text translation was done here using web browser.

Disadvantages:

❖ Data and Information

❖ Navigation

❖ Response Time

❖ Task Efficiency

❖ Security

❖ Browser Compatibility Issues

3.2 Proposed System

- We propose a Graphical User Interface(GUI) application in Python. It's User friendly. And easy to understand. Here, translation can be done via Text and Audio.
- For using audio files, It has been recognized as an effective computer access method for people who are not able to use a keyboard or mouse.

Advanatges:

❖ It's is easy and fast to implement as compared to any other GUI toolkit.

❖ More flexible and stable.

❖ Easy to understand.

❖ Tkinter provides three geometry managers: place, pack, and grid.

3.3 Functional Requirements

Functional requirements specify which output file should be produced from the given file they describe the relationship between the input and output of the system, for each functional requirement a detailed description of all data inputs and their source and the range of valid inputs must be specified.

3.4 Non-Functional Requirements

- Describe user-visible aspects of the system that are not directly related with the functional behaviour of the system.
- Non-Functional requirements include quantitative constraints, such as response time or accuracy.

A) Usability

This section includes all of those requirements that affect usability. Our system is plug and play type. It does not require any special skills or training.

B) Reliability

The system is more reliable because of the qualities that are inherited from the chosen platform python. The code built by using python is more reliable.

C) Performance

The performance characteristics of the system are outlines here:

- Response time
- Accuracy

D) Supportability

The system is designed to be the cross platform supportable. The system is supported on a wide range of hardware and any software platform which supports python and Tkinter

3.5 Hardware Requirements

• Processor	:	I3
• Hard Disk	:	16 GB
• Ram	:	1 GB
• Key Board	:	Standard Windows Keyboard
• Mouse	:	Two or Three Button Mouse

3.6 Software Requirements

• Operating system	:	Windows 10
• Language	:	Python
• IDE Tool	:	IDLE
• Package	:	Tkinter, Playsound, ImageTk

.

CHAPTER-4

SYSTEM STUDY AND ANALYSIS

4.1 Preliminary Investigation

Before transmitting, we shall take precautions to ensure that its emissions will not interfere with transmissions already in progress; if such interference is likely, the message shall await an appropriate break in the communications in progress.

If, these precautions having been taken, the emissions of the message should, nevertheless, interfere with a transmission already in progress, the following rules shall be applied:

a) Clear the input data field before type a message in it.

b) Make sure the space between dots and dashes are in appropriate. Because, space plays a vital role in Morse Code.

4.2 System Architecture

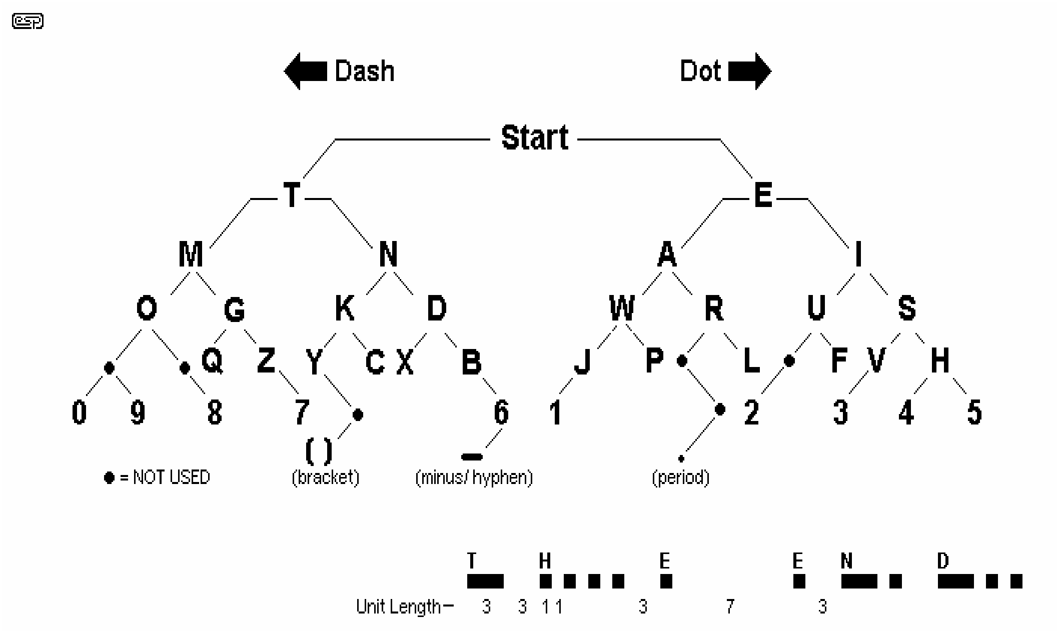


Fig 4.2 System Architecture

CHAPTER 5

SYSTEM DESIGN

5.1 System Design Architecture

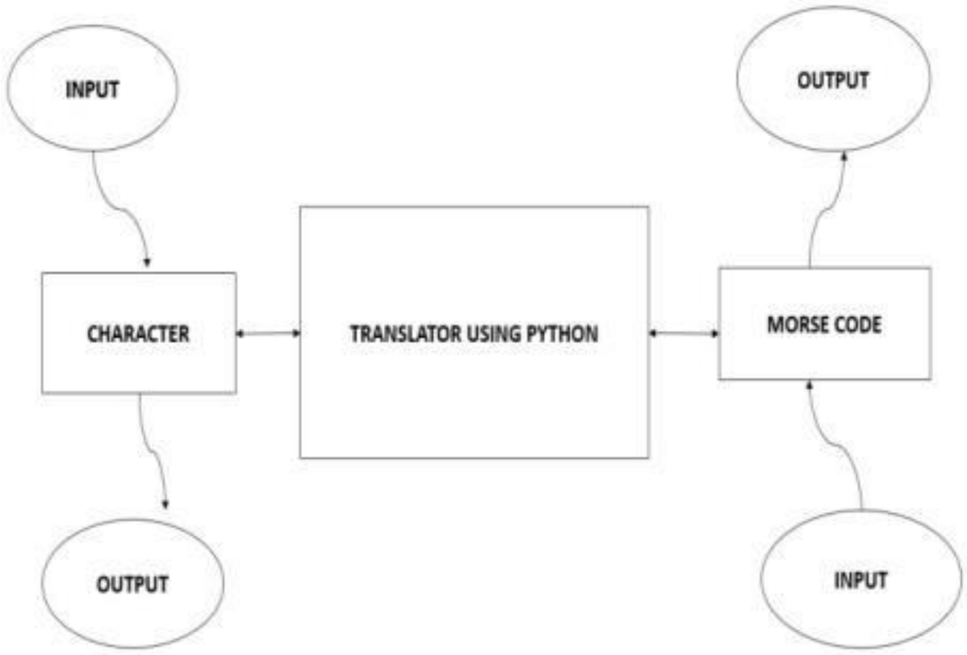


Fig 5.1 Design Architecture

5.2 Input Design And Output Design

During this process of translation from the above diagram, we can say that when a character is given as input by translating it we get a morse code from the hashmaps as output. When Morse code is given as input then by translating it we get a character as output.

Input Design

Input Design plays a vital role , it requires very careful attention for sending information. The input design is to feed data to the application as accurately as possible. So inputs are supposed to be designed effectively so that the errors occurring

while feeding are minimized. Error messages are developed to alert the user whenever he commits some mistakes and guide him in the right way so that invalid entries are not made.
Input design is the process of converting the user entered input . The goal of the input design is to make the data entry and free from errors. The application has been developed in a user-friendly manner. Whenever a user enters erroneous data, an error message is displayed.

Output Design

In output design, when a character is given as input by translating it we get a morse code from the hashmaps as output. When Morse code is given as input then by translating it we get a character as output. The use of the hashmaps is most predominant in python for accessing the data structure .So every key of the character has its value in the database we need to access through a function called itertools. Because we choose python as it is efficient in both fast and accessing a database. Cipher stores the morse translated the form of the English string. Decipher stores the English translated form of the morse string. Context stores morse code of a single character. It keeps count of the spaces between morse characters.
The developed system is highly user friendly and can be easily understood by anyone using it even for the first time.

5.3 Design Diagrams Using UML Approach

UML is a method for describing the system architecture in detail using the blue print. UML represents a collection of best engineering practice that has proven successful in the modeling of large and complex systems. The UML is very important parts of developing object oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the helps UML helps project teams communicate explore potential designs and validate the architectural design of the software.

5.3.1 Use Case Diagram

Use case diagram represents the functionality of the system. Use cases focus on the behavior of the system from an external point of view. Actors are external entities that interact with the system.
Use cases: A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.
Actors: An actor is a person, organization, or external system that plays a role in one or more interactions with the system.
Include: In one form of interaction, a given use case may include another. "Include is a Directed Relationship between two use cases, implying that the behavior of the included use case is inserted into the behavior of the including use case.The first use

case often depends on the outcome of the included use case. This is useful for extracting truly common behaviors from multiple use cases into a single description. The notation is a dashed arrow from the including to the included use case, with the label "«include "»". There are no parameters or return values. To specify the location in a flow of events in which the base use case includes the behavior of another, you simply write include followed by the name of the use case you want to include, as in the following flow for track order.

Extend: In another form of interaction, a given use case (the extension) may extend another. This relationship indicates that the behavior of the extension use case may be inserted in the extended use case under some conditions. The notation is a dashed arrow from the extension to the extended use case, with the label "«extend»". Modelers use the «extend» relationship to indicate use cases that are "optional" to the base use case.

Generalization : A given use case may have common behaviors, requirements, constraints, and assumptions with a more general use case. In this case, describe them once, and deal with it in the same way, describing any differences in the specialized cases. The notation is a solid line ending in a hollow triangle drawn from the specialized to the more general use case (following the standard generalization notation.

Associations : It’s between actors and use cases are indicated in use case diagrams by solid lines. An association exists whenever an actor is involved with an interaction described by a use case. Associations are modeled as lines connecting use cases and actors to one another, with an optional arrowhead on one end of the line. The arrowhead is often used to indicate the direction of the initial invocation of the relationship or to indicate the primary actor within the use case.

Identified Use Cases : The ―user model view‖ encompasses a problem and solution from the preservative of those individuals whose problem the solution addresses. The view presents the goals and objectives of the problem owners and their requirements of the solution. This view is composed of ―use case diagrams‖. These diagrams describe the functionality provided by a system to external integrators. These diagrams contain actors, use cases, and their relationships.

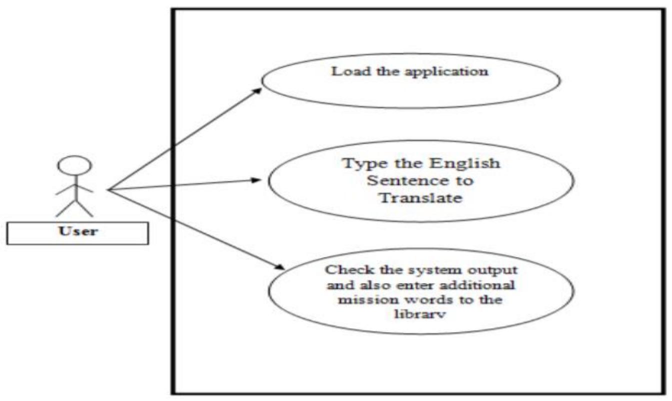

Load the application
Type the English Sentence to Translate
Check the system output and also enter additional mission words to the librarv
User

Fig 5.3.1 Use Case Diagram

5.3.2 Class Diagram

Class-based Modeling, or more commonly class-orientation, refers to the style of object-oriented programming in which inheritance is achieved by defining classes of objects; as opposed to the objects themselves (compare Prototype-based programming).
The most popular and developed model of OOP is a class-based model, as opposed to an object-based model. In this model, objects are entities that combine state (i.e., data), behavior (i.e., procedures, or methods) and identity (unique existence among all other objects). The structure and behavior of an object are defined by a class, which is a definition, or blueprint, of all objects of a specific type. An object must be explicitly created based on a class and an object thus created is considered to be an instance of that class. An object is similar to a structure, with the addition of method pointers, member access control, and an implicit data member which locates instances of the class (i.e. actual objects of that class) in the class hierarchy (essential for runtime inheritance features).

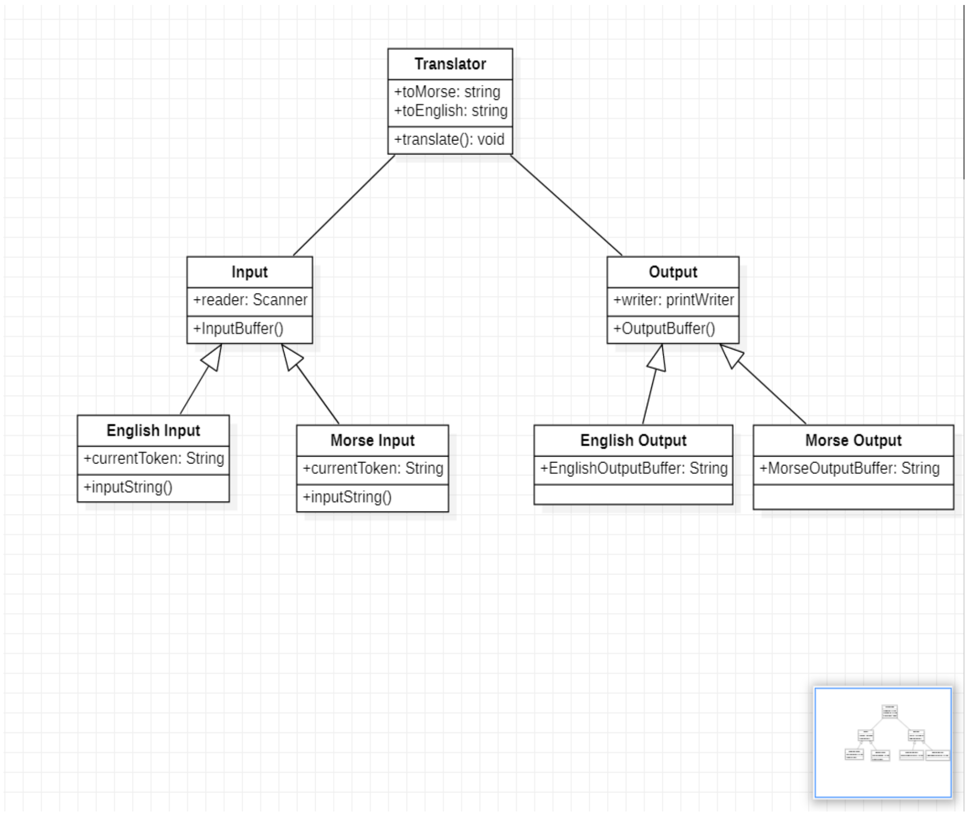


Fig 5.3.2 Class Diagram

5.4 Detailed Design

5.4.1 Sequence Diagram

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams. A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner. If the lifeline is that of an object, it demonstrates a role. Note that leaving the instance name blank can represent anonymous and unnamed instances. In order to display interaction, messages are used. These are horizontal arrows with the message name written above them. Solid arrows with full heads are synchronous calls, solid arrows with stick heads are asynchronous calls and dashed arrows with stick heads are return messages.
Activation boxes, or method-call boxes, are opaque rectangles drawn on top of lifelines to represent that processes are being performed in response to the message (Execution Specifications in UML). Objects calling methods on themselves use messages and add new activation boxes on top of any others to indicate a further level of processing. When an object is destroyed (removed from memory), an X is drawn on top of the lifeline, and the dashed line ceases to be drawn below it (this is not the case in the first example though). It should be the result of a message, either from the object itself, or another. A message sent from outside the diagram can be represented by a message originating from a filled-in circle (found message in UML) or from a border of sequence diagram (gate in UML)

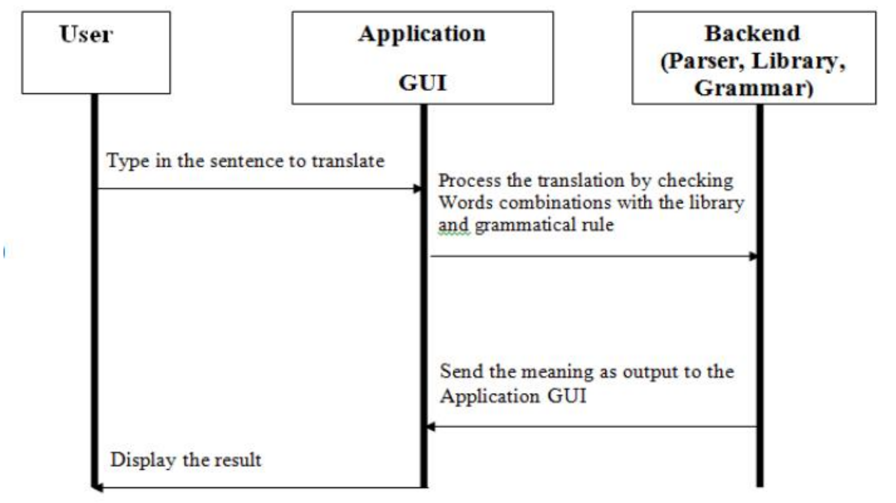

User
Application GUI
Backend (Parser, Library, Grammar)
Type in the sentence to translate
Process the translation by checking Words combinations with the library and grammatical rule
Send the meaning as output to the Application GUI
Display the result

Fig 5.4.1 Sequence Diagram

5.4.2 Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.
Activity diagrams are constructed from a limited repertoire of shapes, connected with arrows. The most important shape types:

- rounded rectangles represent activities;
- diamonds represent decisions;
- bars represent the start (split) or end (join) of concurrent activities;
- a black circle represents the start (initial state) of the workflow;
- An encircled black circle represents the end (final state).

Arrows run from the start towards the end and represent the order in which activities happen. However, the join and split symbols in activity diagrams only resolve this for simple cases; the meaning of the model is not clear when they are arbitrarily combined with the decisions or loops.

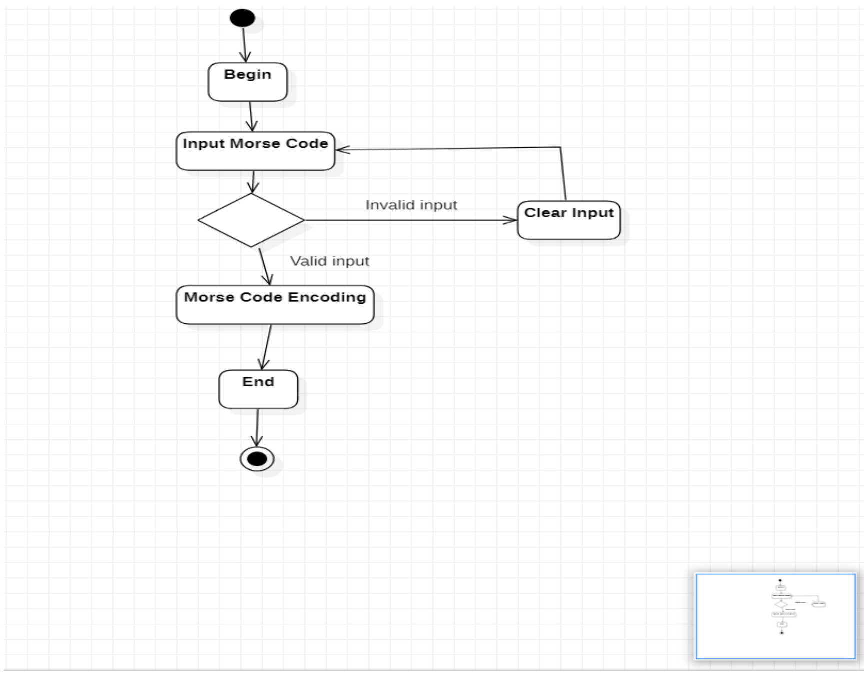


Fig 5.4.2 Activity Diagram for Morse Code

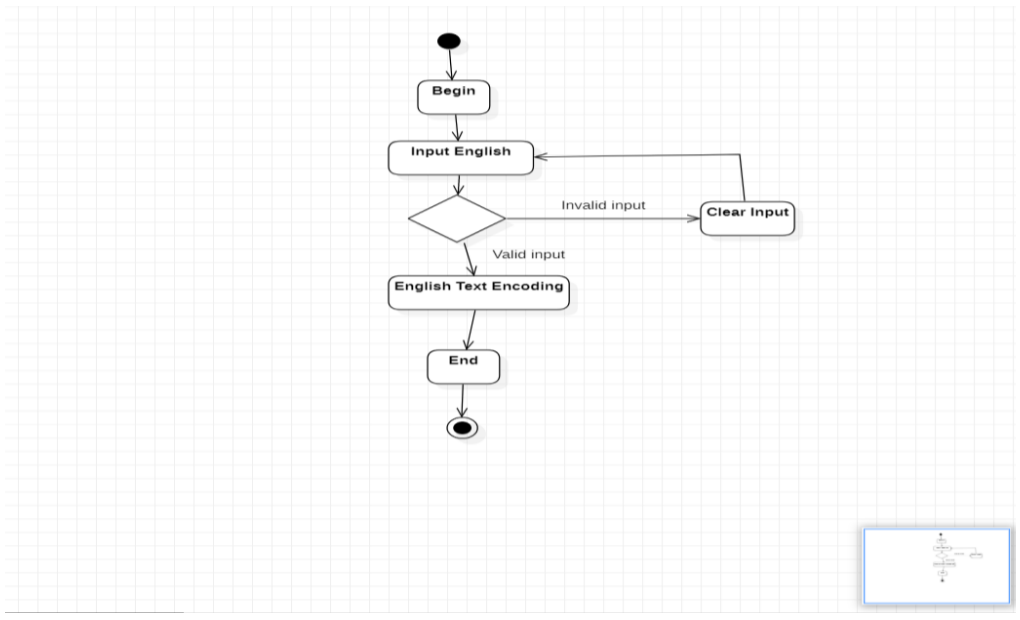


Fig 5.4.2.1 Activity Diagram for English Text

5.4.3 State Chart Diagram:

Objects have behaviors and states. The state of an object depends on its current activity or condition. A state chart diagram shows the possible states of the object and the transitions that cause a change in state. A state diagram, also called a state machine diagram or state chart diagram, is an illustration of the states an object can attain as well as the transitions between those states in the Unified Modeling Language. A state diagram resembles a flowchart in which the initial state is represented by a large black dot and subsequent states are portrayed as boxes with rounded corners. There may be one or two horizontal lines through a box, dividing it into stacked sections. In that case, the upper section contains the name of the state, the middle section (if any) contains the state variables and the lower section contains the actions performed in that state. If there are no horizontal lines through a box, only the name of the state is written inside it. External straight lines, each with an arrow at one end, connect various pairs of boxes. These lines define the transitions between states. The final state is portrayed as a large black dot with a circle around it. Historical states are denoted as circles with the letter H inside.

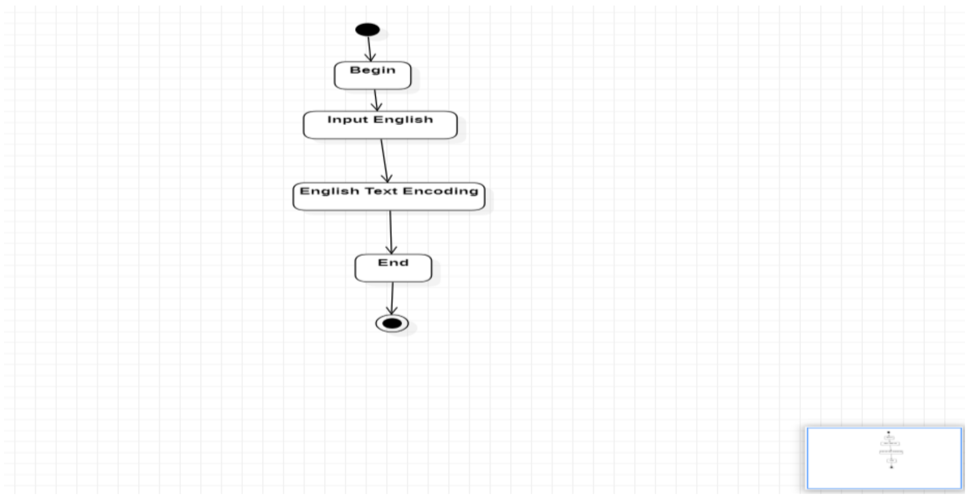


Fig 5.4.3 State Chart Diagram for English Text

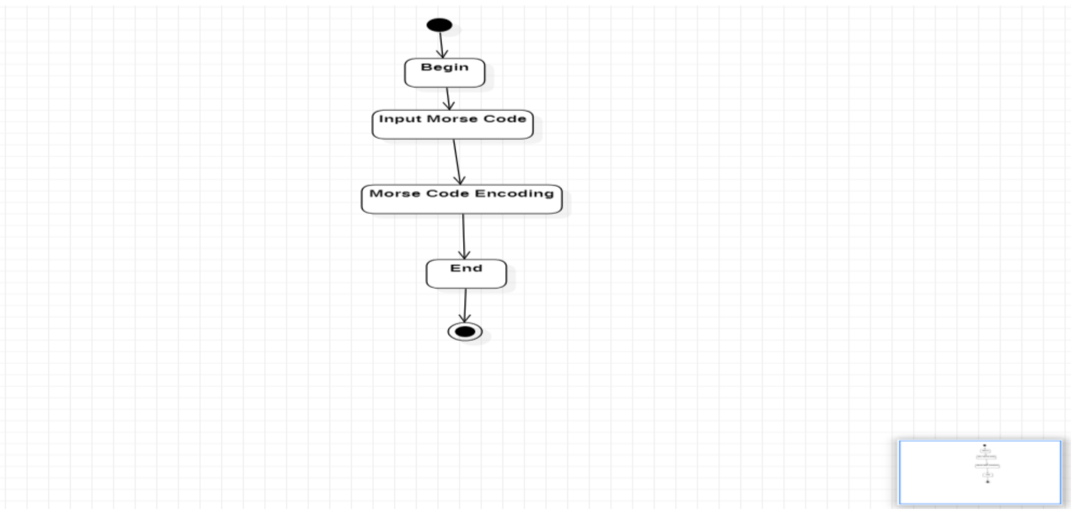


Fig 5.4.3.1 State Chart Diagram for Morse Code

CHAPTER 6

IMPLEMENTATION

6.1 Introduction to Python

Python is an interpreted high-level general-purpose programming language. It is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming.

Features

There are many features in Python, some of them are mentioned below –

- Easy to Code
- Free and open source
- Object-Oriented Language
- GUI Programming Support
- Portable
- High Level Language
- Interpreted Language
- Dynamically Typed Language
- Integrated Language

The Python Platform

A platform is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and MacOS.

The Python is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets.

Python Program Execution:

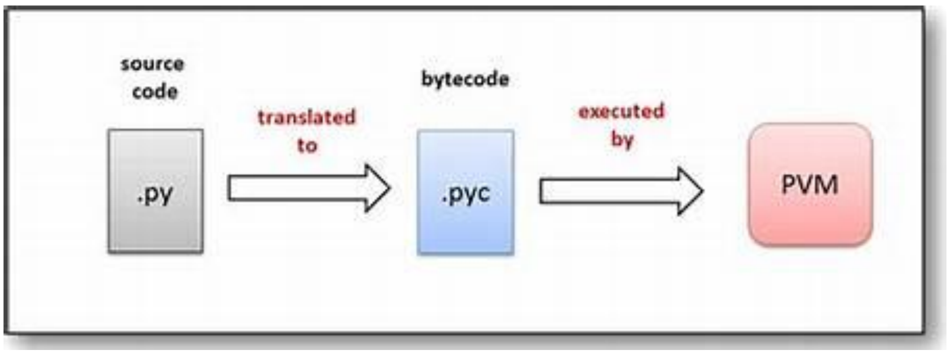


Fig 6.1 Program Execution

6.2 Introduction to Tkinter

Tkinter is the inbuilt python module that is used to create GUI applications. It is one of the most commonly used modules for creating GUI applications in Python as it is simple and easy to work with. You don't need to worry about the installation of the Tkinter module separately as it comes with Python already. It gives an object-oriented interface to the Tk GUI toolkit.

Graphical User Interface(GUI) is a form of user interface which allows users to interact with computers through visual indicators using items such as icons, menus, windows, etc. It has advantages over the Command Line Interface(CLI) where users interact with computers by writing commands using keyboard only and whose usage is more difficult than GUI.

Some other Python Libraries available for creating our own GUI applications are

- Kivy
- Python Qt
- wxPython

Among all Tkinter is most widely used.

Widgets in Tkinter are the elements of GUI application which provides various controls (such as Labels, Buttons, ComboBoxes, CheckBoxes, MenuBars, RadioButtons and many more) to users to interact with the application

Fundamental structure of tkinter program

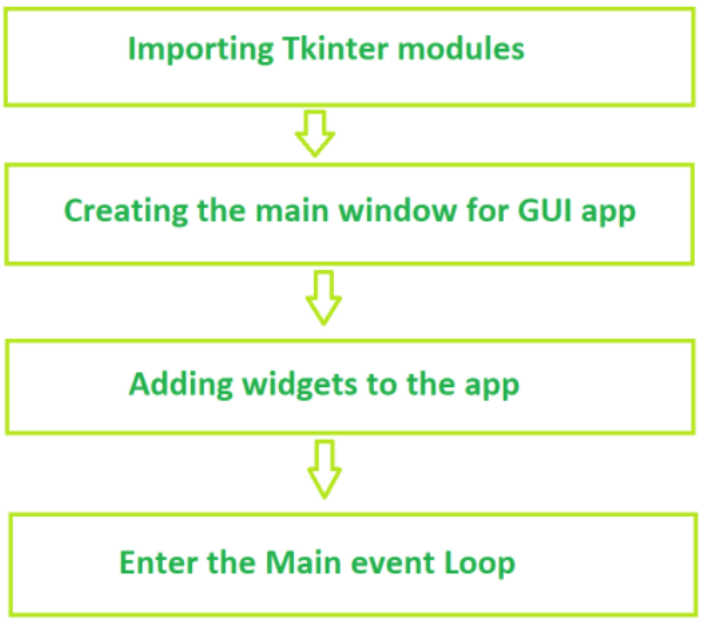

Importing Tkinter modules
Creating the main window for GUI app
Adding widgets to the app
Enter the Main event Loop

Fig 6.2 Structure of Tkinter

Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

To create a Tkinter application:

1. Importing the module – tkinter
2. Create the main window (container)
3. Add any number of widgets to the main window
4. Apply the event Trigger on the widgets.

Importing tkinter is same as importing any other module in the Python code.

Note that the name of the module in Python 2.x is 'Tkinter' and in Python 3.x it is 'tkinter'.

For importing the module, we have to do like this

import tkinter

These are two main methods used which the user needs to remember while creating the Python application with GUI.

1) Tk(screenName=None, baseName=None, className=’Tk’, useTk=1):

To create a main window, tkinter offers a method ‘Tk(screenName=None, baseName=None, className=’Tk’, useTk=1)’. To change the name of the window, you can change the className to the desired one. The basic code used to create the main window of the application is:

m=tkinter.Tk() where m is the name of the main window object

2) mainloop():

There is a method known by the name mainloop() is used when your application is ready to run. mainloop() is an infinite loop used to run the application, wait for an event to occur and process the event as long as the window is not closed.

m.mainloop()

Tkinter also offers access to the geometric configuration of the widgets which can organize the widgets in the parent windows. There are mainly three geometry manager classes class.

1. **pack() method:**It organizes the widgets in blocks before placing in the parent widget.
2. **grid() method:**It organizes the widgets in grid (table-like structure) before placing in the parent widget.
3. **place() method:**It organizes the widgets by placing them on specific positions directed by the programmer.

There are a number of widgets which you can put in your tkinter application. Some of the major widgets are explained below:

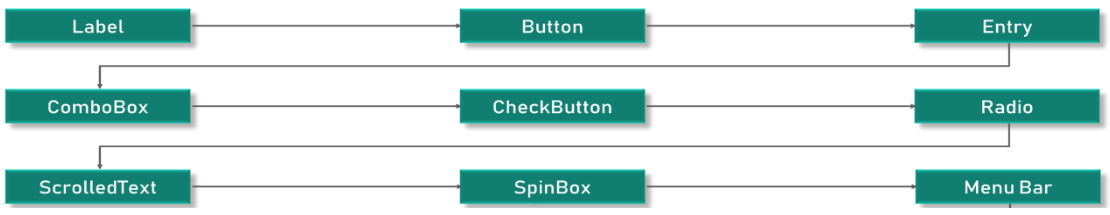


Fig 6.2.1 Widgets

Button:To add a button in your application, this widget is used.
The general syntax is:

w=Button(master, option=value)

master is the parameter used to represent the parent window.
There are number of options which are used to change the format of the Buttons. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **activebackground**: to set the background color when button is under the cursor.
- **activeforeground**: to set the foreground color when button is under the cursor.
- **bg**: to set he normal background color.
- **command**: to call a function.
- **font**: to set the font on the button label.
- **image**: to set the image on the button.
- **width**: to set the width of the button.
- **height**: to set the height of the button.

Message: It refers to the multi-line and non-editable text. It works same as that of Label.
The general syntax is:

w = Message(master, option=value)

master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **bd**: to set the border around the indicator.
- **bg**: to set he normal background color.
- **font**: to set the font on the button label.
- **image**: to set the image on the widget.
- **width**: to set the width of the widget.
- **height**: to set the height of the widget.

Canvas: It is used to draw pictures and other complex layout like graphics, text and widgets.
The general syntax is:

w = Canvas(master, option=value)

master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **bd**: to set the border width in pixels.
- **bg**: to set the normal background color.
- **cursor**: to set the cursor used in the canvas.
- **highlightcolor**: to set the color shown in the focus highlight.
- **width**: to set the width of the widget.
- **height**: to set the height of the widget

Entry: It is used to input the single line text entry from the user.. For multi-line text input, Text widget is used.

The general syntax is:

w=Entry(master, option=value)

master is the parameter used to represent the parent window.
There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **bd**: to set the border width in pixels.
- **bg**: to set the normal background color.
- **cursor**: to set the cursor used.
- **command**: to call a function.
- **highlightcolor**: to set the color shown in the focus highlight.
- **width**: to set the width of the button.
- **height**: to set the height of the button.

CheckButton: To select any number of options by displaying a number of options to a user as toggle buttons.
The general syntax is:

w = CheckButton(master, option=value)

There are number of options which are used to change the format of this widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **Title**: To set the title of the widget.
- **activebackground**: to set the background color when widget is under the cursor.
- **activeforeground**: to set the foreground color when widget is under the cursor.
- **bg**: to set he normal backgrouSteganography
- **font**: to set the font on the button label.

image: to set the image on the widget.

Frame: It acts as a container to hold the widgets. It is used for grouping and organizing the widgets.
The general syntax is:

w = Frame(master, option=value)

master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **highlightcolor**: To set the color of the focus highlight when widget has to be focused.
- **bd**: to set the border width in pixels.

- **bg**: to set the normal background color.
- **cursor**: to set the cursor used.
- **width**: to set the width of the widget.
- **height**: to set the height of the widget.

Label: It refers to the display box where you can put any text or image which can be updated any time as per the code.
The general syntax is:

w=Label(master, option=value)

master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **bg**: to set he normal background color.
- **bg** to set he normal background color.
- **command**: to call a function.
- **font**: to set the font on the button label.
- **image**: to set the image on the button.
- **width**: to set the width of the button.
- **height**" to set the height of the button.

RadioButton: It is used to offer multi-choice option to the user. It offers several options to the user and the user has to choose one option.
The general syntax is:

w = RadioButton(master, option=value)

There are number of options which are used to change the format of this widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **activebackground**: to set the background color when widget is under the cursor.
- **activeforeground**: to set the foreground color when widget is under the cursor.
- **bg**: to set he normal background color.
- **command**: to call a function.
- **font**: to set the font on the button label.
- **image**: to set the image on the widget.
- **width**: to set the width of the label in characters.
- **height**: to set the height of the label in characters.

Text: To edit a multi-line text and format the way it has to be displayed.
The general syntax is:

w =Text(master, option=value)

There are number of options which are used to change the format of the text. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **highlightcolor**: To set the color of the focus highlight when widget has to be focused.
- **insertbackground**: To set the background of the widget.
- **bg**: to set he normal background color.
- **font**: to set the font on the button label.
- **image**: to set the image on the widget.
- **width**: to set the width of the widget.
- **height**: to set the height of the widget.

Listbox: It offers a list to the user from which the user can accept any number of options.
The general syntax is:

w = Listbox(master, option=value)

master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **highlightcolor**: To set the color of the focus highlight when widget has to be focused.
- **bg**: to set he normal background color.
- **bd**: to set the border width in pixels.
- **font**: to set the font on the button label.
- **image**: to set the image on the widget.
- **width**: to set the width of the widget.
- **height**: to set the height of the widget.

MenuButton: It is a part of top-down menu which stays on the window all the time. Every menubutton has its own functionality.
The general syntax is:

w = MenuButton(master, option=value)

master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **activebackground**: To set the background when mouse is over the widget.
- **activeforeground**: To set the foreground when mouse is over the widget.
- **bg**: to set he normal background color.
- **bd**: to set the size of border around the indicator.
- **cursor**: To appear the cursor when the mouse over the menubutton.
- **image**: to set the image on the widget.
- **width**: to set the width of the widget.
- **height**: to set the height of the widget.
- **highlightcolor**: To set the color of the focus highlight when widget has to be focused.

6.3 Modules

Play sound : Run the following command to install the package

```
pip install playsound
```

- The playsound module contains only a single function named **playsound()**.
- It requires one argument: the path to the file with the sound we have to play. It can be a local file, or a URL.
- There's an optional second argument, **block**, which is set to True by default. We can set it to False for making the function run **asynchronously**.
- It works with both **WAV** and **MP3** files.

Images: In order to do various operations and manipulations on images, we require Python Pillow package. If the Pillow package is not present in the system then it can be installed using the below command.

In Command prompt : **pip install Pillow**

In Tkinter, some widgets can display an image such as Label and Button. These widgets take an image argument that allows them to display an image.However, you can't just simply pass the path to an image file to the image argument. Instead, you need to create a PhotoImage object and pass it the image argument.It's important to note that you keep the reference to the PhotoImage object in scope for as long as the image will be shown. Otherwise, the image won't appear.

CHAPTER 7

SYSTEM TESTING

7.1 Introduction to Testing

Testing is a process, which reveals errors in the program. It is the major quality measure employed during software development. During testing, the program is executed with a set of test cases and the output of the program for the test cases is evaluated to determine if the program is performing as it is expected to perform.

7.2 Testing Strategies

In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development are:

7.2.1 Unit Testing

Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements. Each module can be tested using the following two Strategies:

7.2.1.1 Black Box Testing

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been uses to find errors in the following categories:

- Incorrect or missing functions
- Interface errors
- Errors in data structure or external database access
- Performance errors
- Initialization and termination errors.

In this testing only the output is checked for correctness. The logical flow of the data is not checked.

7.2.1.2 White Box Testing

In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been uses to generate the test cases in the following cases:

- Guarantee that all independent paths have been Executed.
- Execute all logical decisions on their true and false Sides.
- Execute all loops at their boundaries and within their operational bounds
- Execute internal data structures to ensure their validity.

7.2.2 Integrating Testing

Integration testing ensures that software and subsystems work together a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated together. In this case the communication between the device and Google Translator Service.

7.2.3 System Testing

Involves in-house testing in an emulator of the entire system before delivery to the user. It's aim is to satisfy the user the system meets all requirements of the client's specifications.

7.2.4 Acceptance Testing

It is a pre-delivery testing in which entire system is tested in a real android device on real world data and usage to find errors.

7.3 Test Approach

Testing can be done in two ways:

- Bottom up approach
- Top down approach

A) Bottom up Approach

Testing can be performed starting from smallest and lowest level modules and proceeding one at a time. For each module in bottom up testing a short program executes the module and provides the needed data so that the module is asked to perform the way it will when embedded within the larger system. When bottom level modules are tested attention turns to those on the next level that use the lower level ones they are tested individually and then linked with the previously examined lower level modules.

B) Top down approach

This type of testing starts from upper level modules. Since the detailed activities usually performed in the lower level routines are not provided stubs are written. A stub is a module shell called by upper level module and that when reached properly will return a message to the calling module indicating that proper interaction occurred. No attempt is made to verify the correctness of the lower level module.

7.4 Validation

The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed.

CHAPTER – 8

OUTPUTS

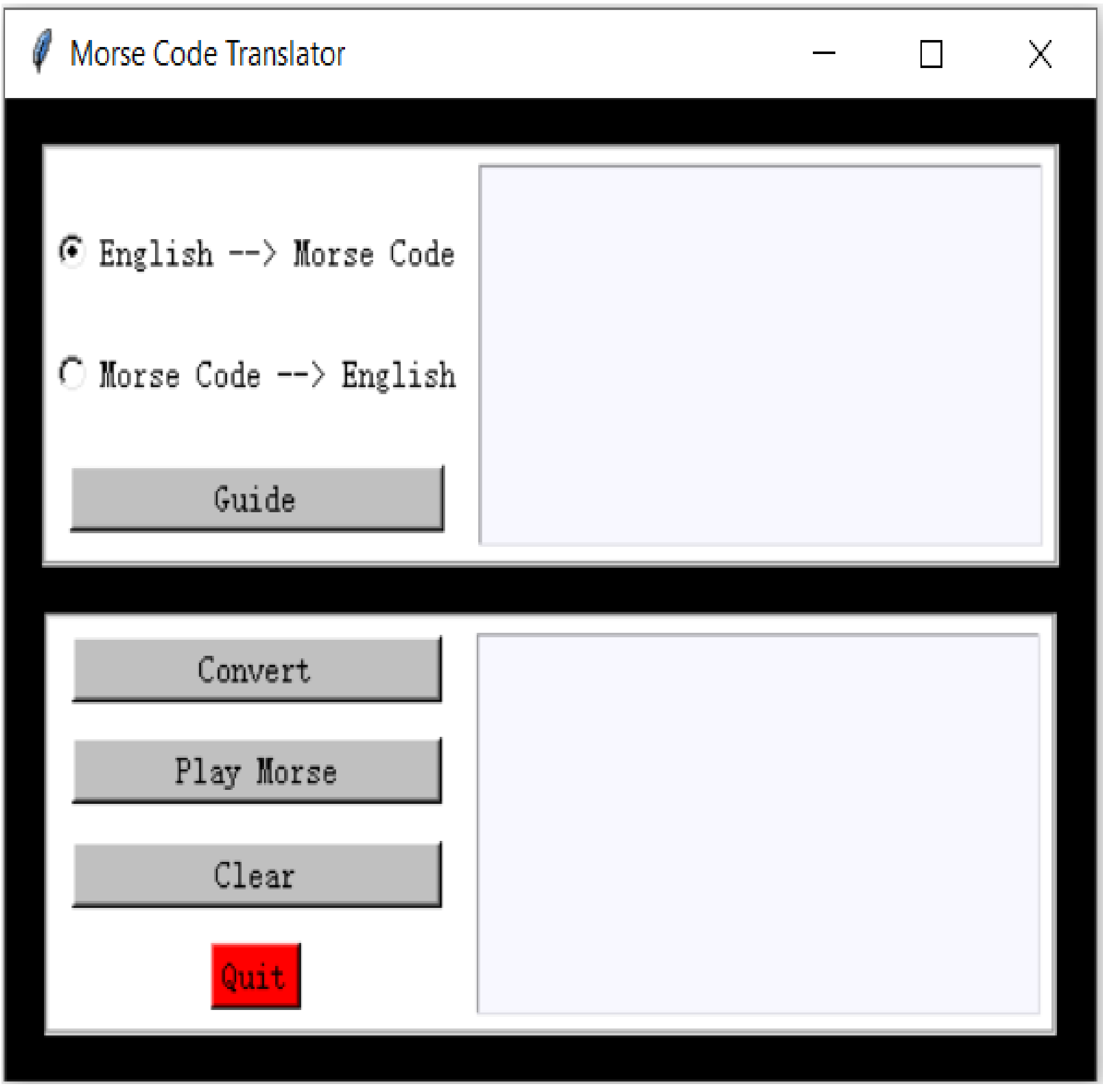


Fig 8.1 GUI Interface

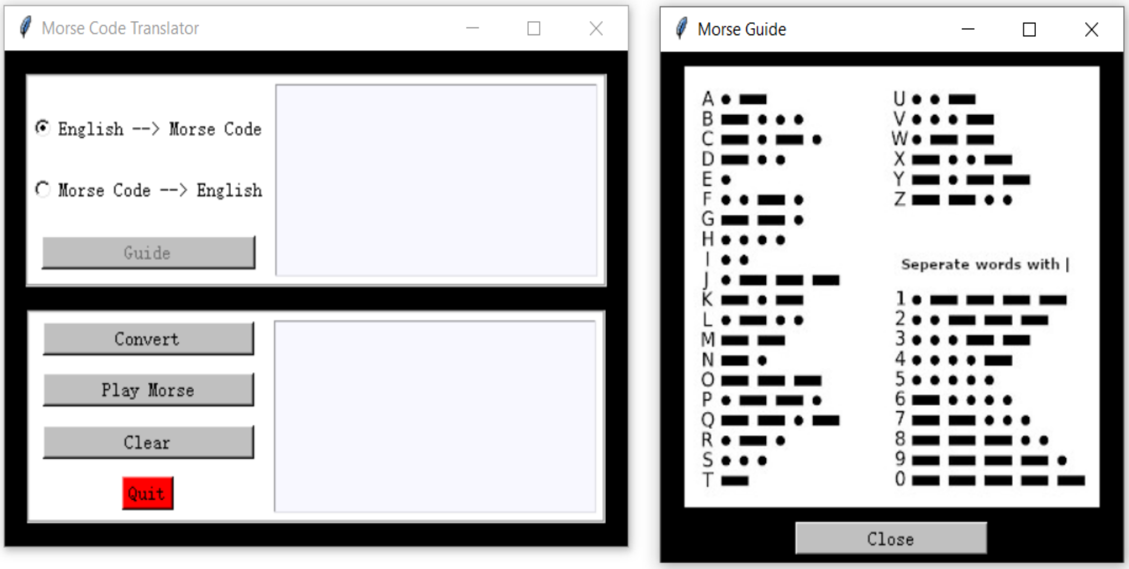
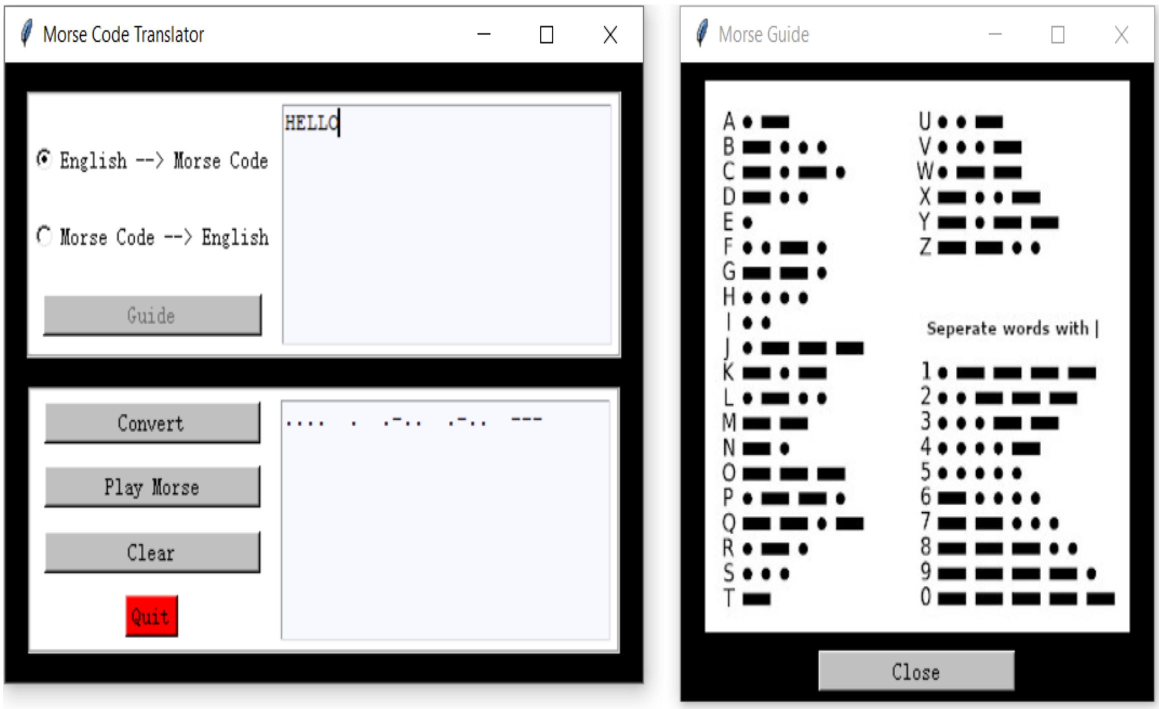


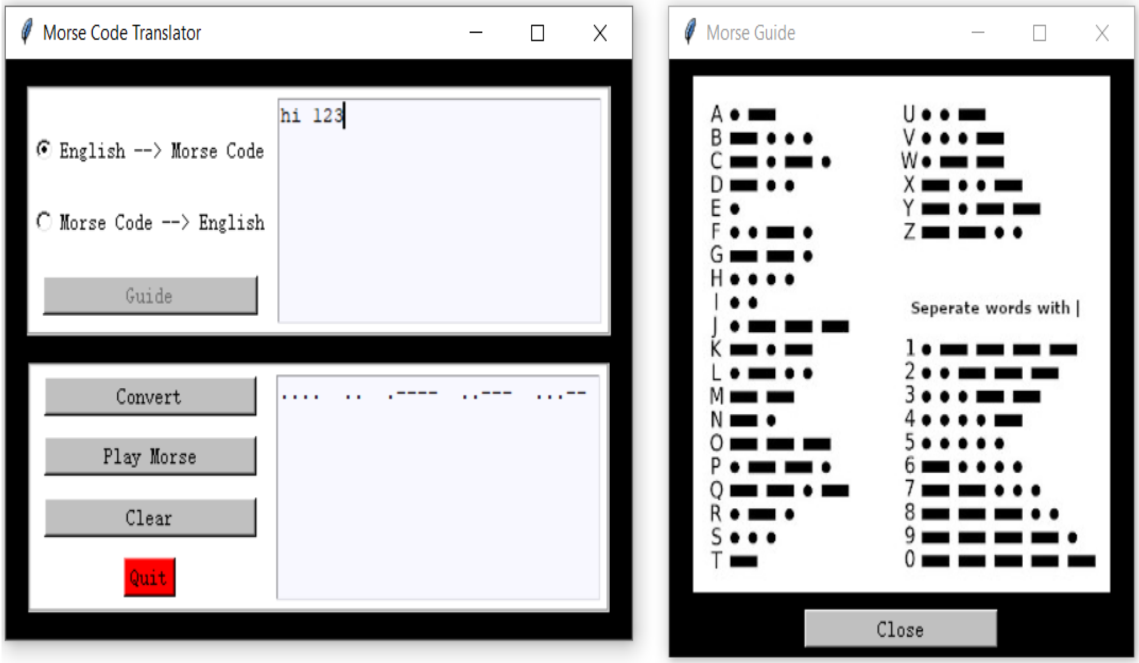
Fig 8.2 GUI Interface along with Guide



```
HELLO
```

```
.... . .-.. .-.. ---
```

Fig 8.3 English Text as Input



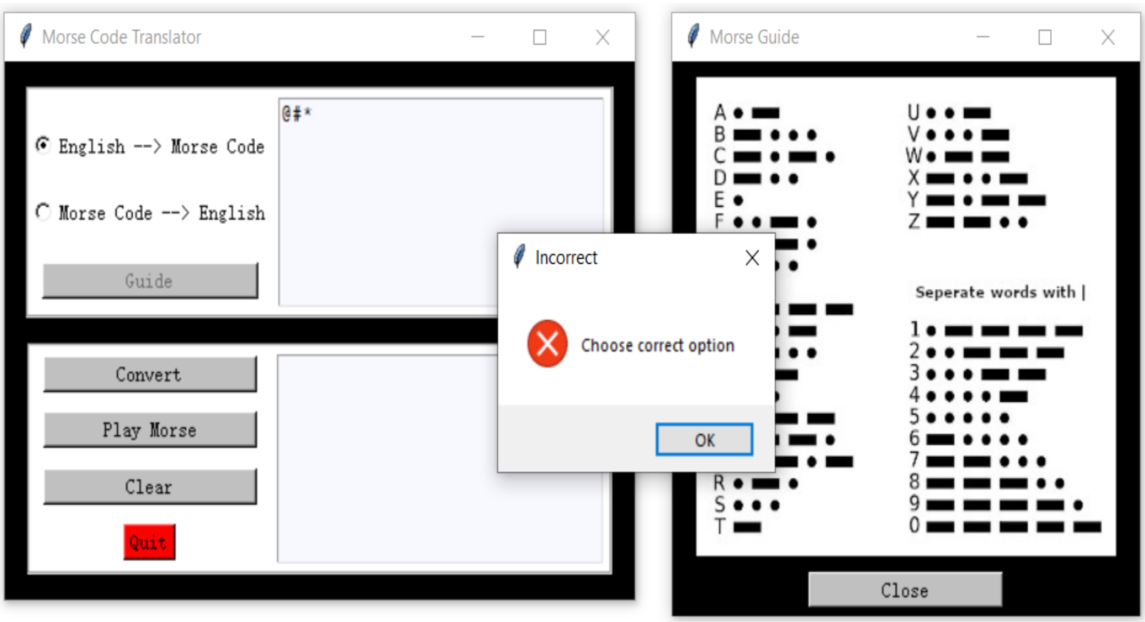
[✓ English --> Morse Code] Morse Code --> English

```
hi 123
```

```
.... .. .---- ..--- ...--
```

Seperate words with |

Fig 8.4 Alpha Numeric as Input



[✓ English --> Morse Code] Morse Code --> English

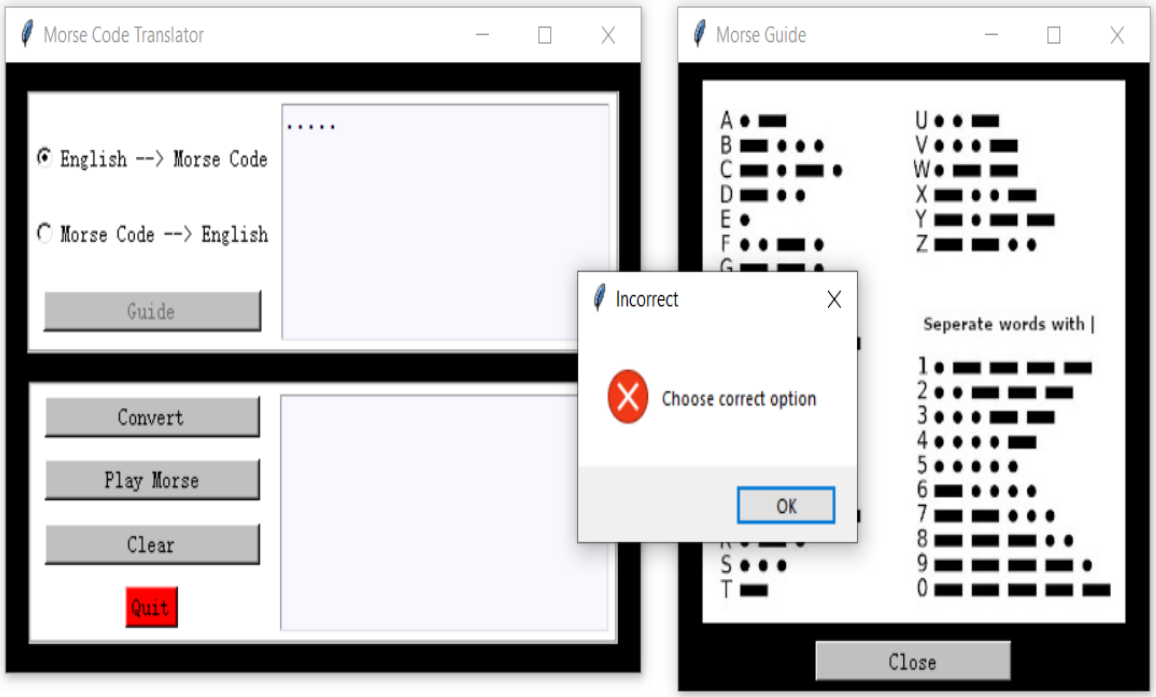
```
@#*
```

Incorrect

Choose correct option

Seperate words with |

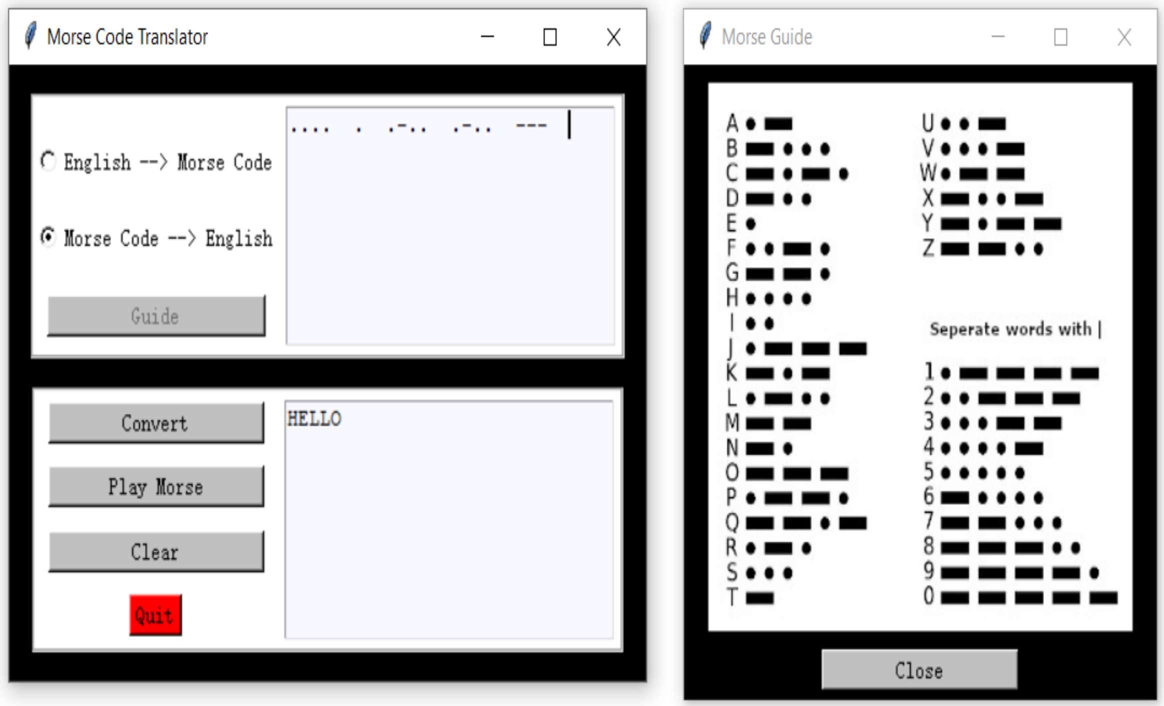
Fig 8.5 Special Characters as Input



.....

Choose correct option

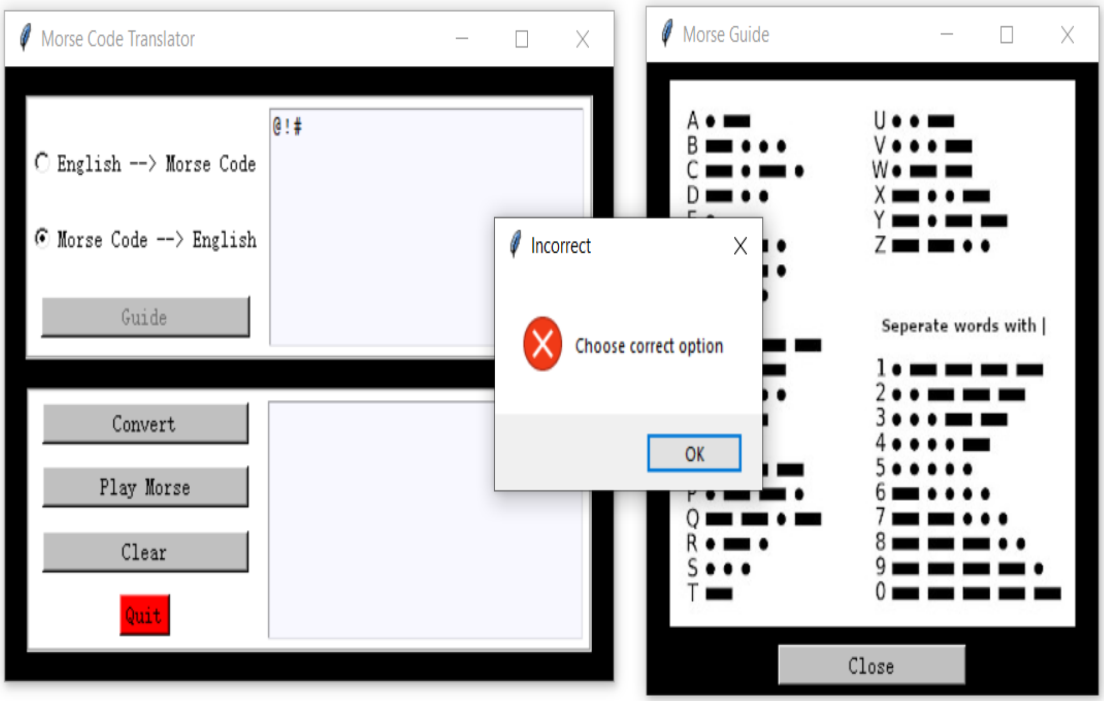
Fig 8.6 Incorrect Input is given



.... . .-.. .-.. ---

HELLO

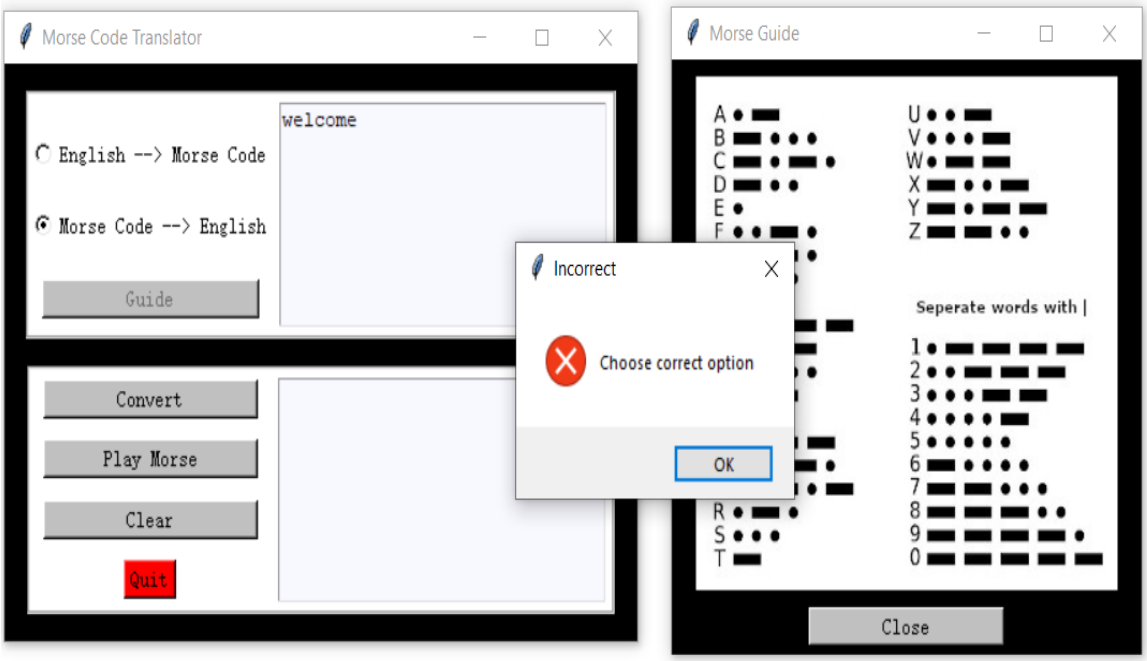
Fig 8.7 Morse Code as Input



@!#

Choose correct option

Fig 8.8 Special Character as Input



welcome

Choose correct option

Fig 8.9 Incorrect Input is given

CHAPTER – 9

CONCLUSION AND FUTURE SCOPE

9.1 Conclusion

This system was designed to transmit message securely to long distance.This design can be used in different areas like long communication, military, external affairs etc. By using this system there is no need of tapping device for transmission.So this overcomes the security problems.

9.2 Future Scope

The future scope of this project is Two-factor authentication is basically providing a two-layer of security to protect an account or system. Concerning the future enhancement, can implement facial recognition for each user, there will be no need to enter the password at all. Also trying to deploy this model in government sectors, with a fewer number of steps required for authentication.

CHAPTER – 10

REFERENCES

[1] Chethana Prasad K, Disha S, Divya TM, Sunil Kumar GR**,** Morse Code based secured Authentication system through AI**,** IARJSET, Vol. 8, Issue 5, May 2021.

[2] Katie A. Hart, Jennifer Chapin, and Dr. Kyle B. Reed, "Haptics Morse Code Communication for Deaf and Blind Individuals" 38th Annual
International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2016.

[3] Cheng-Hong Yang, Ching-Hsing Luo, Yuan-Long Jeang, Gwo-Jia Jon, A novel approach to adaptive Morse code recognition for disabled
persons, In Mathematics and Computers in Simulation, Volume 54, Issues 1–3, 2000.

[4] Sourya Dey, Keith M. Chugg and Peter A. Beerel presented a paper on Morse code datasets for machine learning university of south California Los Angeles.

[5] Paparao Nalajala, Bhavana Godavarth , M Lakshmi Raviteja, Deepthi Simhadri presented a paper on Morse code generator using Microcontroller using Alphanumeric keyboard, Department of electronics institute of Aeronautical, Hyderabad.

[6] Dr A.Murugan, R.Thilagavathy Associate Professor, PG and Research Department of Computer Science, Affiliated to University of Madras, Chennai, India.

[7] Zhimeng Yin, Wenchao Jiang, Song Min Kim, Tian, Department of Computer Science and Engineering, University of Minnesota, U.S.