CODE NOW

Main Project Report

Submitted by

DILNA P S

Reg No:FIT20MCA-2045

In partial fulfillment of the requirements for award of the degree of

Master of Computer Applications

Of

A P J Abdul Kalam Technological University

FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)®
ANGAMALY-683577, ERNAKULAM(DIST)

JULY 2022

DECLARATION

I, **Dilna P S** hereby declare that the report of this project work, submitted to the Department of Computer Applications, Federal Institute of Science and Technology (**FISAT**), Angamaly in partial fulfillment of the award of the degree of Master of Computer Applications is an authentic record of my original work.

The report has not been submitted for the award of any degree of this university or any other university.

Date:

Place : Angamaly **DILNA P S**

FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)®
ANGAMALY, ERNAKULAM-683577

DEPARTMENT OF COMPUTER APPLICATIONS

CERTIFICATE

This is to certify that the project report titled **"CODE NOW "** submitted by **Dilna P S [Reg No: FIT20MCA-2045]** towards partial fulfillment of the requirements for the award of the degree of Master of Computer Applications is a record of bonafide work carried out by her during the year 2022.

Project Guide
Mrs. Rose Mary Mathew

Head of the Department
Dr. Deepa Mary Mathews

Submitted for the viva-voce held on ..... at .....

Examiner :

ACKNOWLEDGEMENT

I am deeply grateful to **Dr.Manoj George** , Principal, FISAT, Angamaly for providing me with adequate facilities, way and means by which I am able to complete my main project work and I express my thanks to **Dr. C. Sheela** ,Vice Principal FISAT, Angamaly.

My sincere thanks to **Dr.Deepa Mary Mathews**, Head of the department of MCA, FISAT, who had been a source of inspiration. I express heartiest thanks to **Mrs.Rose Mary Mathew** my project guide for her encouragement and valuable suggestion.I express my gratitude to my scrum master **Dr.Santhosh Kottam** and the faculty members in my department for their constant encouragement and never ending support throughout the project.

Finally I express my thanks to all my friends who gave me wealth of suggestion for successful completion of this project.

ABSTRACT

When technology advances, the challenge of competition becomes more difficult. It is an overhead for developers to remember each and every kind of syntax and cause or write every line of code. It's very feverish and time consuming. Syntactical and structural ways of programming Programmers are divided into three groups: beginner users, information intermittent users, and professional users. It is still a challenge for inexperienced users to write code that is free of typographical errors, even though they have theoretical knowledge of the programming language, its structure and syntax, as well as the logic of the software. So the aim of this project is to devise a system for writing or developing software code with less stress and time. This system allows developers to start coding with a single press

In this digital era there are various digitization is happening like the AI is deploying so fast, various automations are happening and taking place of human and reducing the efforts of human. And all this things are happening because of programmers/developers they are doing a very hard work for the development of technology. As effect of this this work many developers/programmers are facing various physical / mental disorders like stress, carpel tunnel syndrome and more problems. So here I am developing a system which will reduce their injuries or problems like this by creating solution i.e. they have to just speak they want code the system will take it and create proper code

Contents

1 Introduction **8**

1.1 Existing System And Proposed System 10

2 PROOF OF CONCEPT **11**

2.1 Objectives 11

3 IMPLEMENTATION **12**

3.1 System Architecture 13

3.2 Modules 15

3.2.1 UI designing 15

3.2.2 Voice recognition 15

3.2.3 Code Converter 15

3.2.4 Voice translator 15

3.3 Algorithm 16

3.3.1 Speech Recognition 16

3.3.2 Natural Language Processing 17

3.4 Tools 19

3.4.1 Front end 19

3.4.2 Back end 19

4 RESULT ANALYSIS **21**

5 VALIDATION AND TESTING 22
5.1 Test Plan . . . 22
5.1.1 White Box Testing . . . 24
5.1.2 Black Box Testing . . . 24
5.1.3 Integration Testing . . . 24
5.2 Classes of Tests Formal Test Case . . . 25
5.2.1 Informal Test Case . . . 25

6 REQUIREMENT ANALYSIS AND SPECIFICATION 26
6.1 REQUIREMENT ANALYSIS/LITERATURE VIEW . . . 26
6.2 REQUIREMENT SPECIFICATION . . . 27
6.2.1 FUNCTIONAL REQUIREMENT . . . 27
6.2.2 NON-FUNCTIONAL REQUIREMENT . . . 28
6.3 FEASIBILITY STUDY . . . 29
6.3.1 TECHNICAL FEASIBILITY . . . 29
6.3.2 ECONOMICAL FEASIBILITY . . . 29
6.3.3 OPERATIONAL FEASIBILITY . . . 30

7 CONCLUSION AND FUTURE SCOPE 31
7.1 Conclusion . . . 31
7.2 Future Scope . . . 32

8 APPENDIX 33
8.1 Source Code . . . 33
8.1.1 routes.py . . . 33
8.2 Screen Shots . . . 45

9 REFERENCES 49

Chapter 1

Introduction

Programming involves human efforts, hardware - due to manual typing through keyboard, there are many chances to meet an error while typing. As programming is important for students and innovation, as manual typing is time consuming there is a need for an advanced system, which would reduce programmer's effort and promote smart work. The system, which I am going to develop, is easier than this manual typing of code. Voice coding needs two kinds of software: a speech-recognition engine and a platform for voice coding commands. In this I am creating a platform where one can just give a command gets the total code done by platform. For this I am creating Grammar for it which will contain set of instructions which are required for it. I am developing a solution which will take an instruction from user but there's a no need to give it with syntax. There is also a real-time voice translator that can translate voice input and give translated voice output generated from it.

The technology used for this project is Artificial Intelligence. Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to the natural intelligence displayed by animals including humans. AI research has been defined as the field of study of intelligent agents, which refers to any system that perceives its environment and takes actions that maximize its chance of achieving its goals. The term "artificial intelligence" had previously been used to describe machines

that mimic and display ”human” cognitive skills that are associated with the human mind, such as ”learning” and ”problem-solving”. This definition has since been rejected by major AI researchers who now describe AI in terms of rationality and acting rationally, which does not limit how intelligence can be articulated.

AI applications include advanced web search engines (e.g., Google), recommendation systems (used by YouTube, Amazon and Netflix), understanding human speech (such as Siri and Alexa), self-driving cars (e.g., Tesla), automated decision-making and competing at the highest level in strategic game systems (such as chess and Go). As machines become increasingly capable, tasks considered to require ”intelligence” are often removed from the definition of AI, a phenomenon known as the AI effect. For instance, optical character recognition is frequently excluded from things considered to be AI, having become a routine technology.

The various sub-fields of AI research are centered around particular goals and the use of particular tools. The traditional goals of AI research include reasoning, knowledge representation, planning, learning, natural language processing, perception, and the ability to move and manipulate objects.[c] General intelligence (the ability to solve an arbitrary problem) is among the field’s long-term goals. To solve these problems, AI researchers have adapted and integrated a wide range of problem-solving techniques—including search and mathematical optimization, formal logic, artificial neural networks, and methods based on statistics, probability and economics. AI also draws upon computer science, psychology, linguistics, philosophy, and many other fields.

1.1 Existing System And Proposed System

There are lots of AI based speech recognition applications like Google Alexa, Apple Siri etc are available. But there are not many applications that convert users speech to code. This is not just a speech to text convert application. The user just need to give the instruction. Users are not necessary to know the syntax. Users have to just speak the code they want the system will take it and create proper python code. And also there is a translator available in the application that the any language can be converted to any other language. The proposed system can be used by programmers with beginner level effectively. The users or programmers just need to know the logic they do not need to know the syntax. There will have lots of bugs in a program due syntax errors. This system will help us to create programs with least syntax error or zero syntax error. In this system there's no need to specify the required syntax the system will automatically identify the required syntax.

Chapter 2

PROOF OF CONCEPT

This project aims to display the python code according to the users instruction through voice and translate major languages to other languages. Here using machine learning concept in which voice will be recognized by using the natural language processing and will be printed on the editor. Firstly, users speech convert to text and code according to the text and print on editor. Existing systems like Google assistant recognizes the voice and will perform the operate according to the voice instructions, but in our system we are developing the same approach for programming purpose. In this system there's no need to specify the required syntax the system will automatically identify the required syntax.

2.1 Objectives

- Users do not need to know the syntax.
- Programmers with beginner level can effectively use.
- No need to type the code.
- Users just need to know the logic.
- Can translate any language.

Chapter 3

IMPLEMENTATION

Implementation is the stage in the project where the theoretical design is turned in to a working system and is giving confidence and effectiveness. It involves careful planning investigation of the current system and its constraints on implementation design of methods to achieve the change, over an evaluation of change over methods. Apart from planning major tasks of preparing the implementation is education and training of the users. The more complex system is being implemented the more evolved will be the system analysis and design efforts required for the implementation.

The implementation process begins with repairing a plan for the implementation of the system. According to this plan the activities are to be carried out, discussions made regarding the equipment must be acquired to implement the system. Implementation is the final and the most important phase. The most critical stage in achieving a successful new system and in giving the users confidence is only after through testing is done and if it is found to work according to the specification. This method also offers the greatest security since the old system can take over if the errors are found or inability to handle certain type of transactions while using the new system.

The versatility and ease of use of Python, as well as the availability of APIs powered by AI, have opened up myriad programming possibilities in the realm of automation.
Steps to generate requested code by recognizing voice input:-

- Set up voice recognition so we can tell the AI what to write for us
- Make sure the AI can parse the voice input
- Code the code – write the routines that the system to be able to handle for you

Steps to translate to any language by recognizing voice input:-

- Set up voice recognition so we can tell the AI what to translate and to in which language
- A tuple of all the languages mapped with their code
- Mapping user input with the language code

3.1 System Architecture

The architecture diagram for the voice converter module is given in figure 3.1. User's voice is accepted through a microphone. The input speech is then convert to text using SpeechRecognition library. Using that parse and interpret the input string in order to write the code.

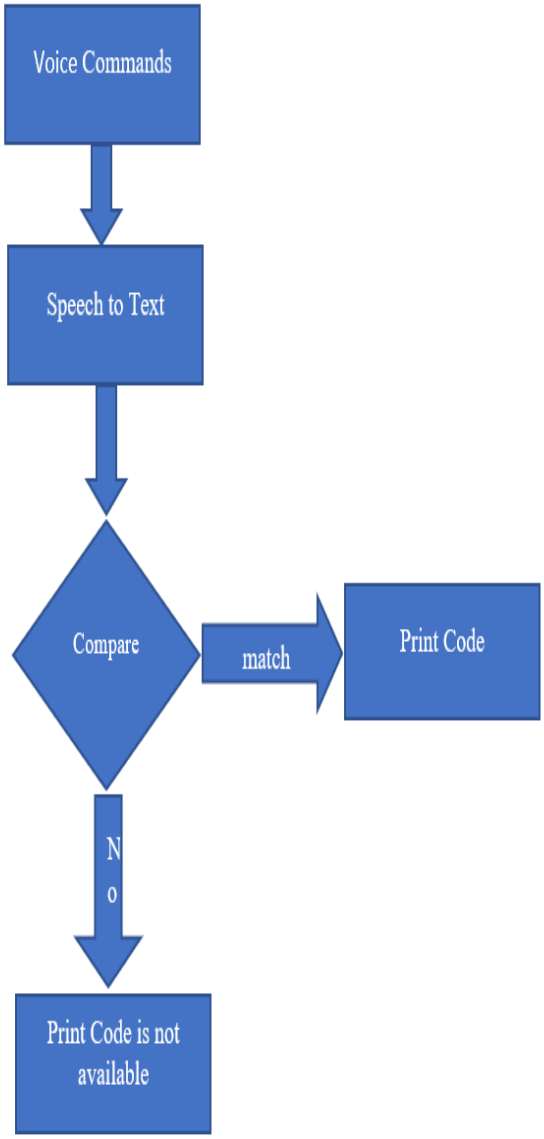


Figure 3.1: Architecture

3.2 Modules

3.2.1 UI designing

In this module, I am designing an Online platform/plugin which will just have to start and after that all the work can be done by voice instruction. UI contains two pages, one is the main page and the other is the voice converter page. Main page consists a button which redirect to the voice converter page. Voice converter page consists a button which enables the microphone and listen to what the user say.

3.2.2 Voice recognition

In this module the proposed system will recognize the voice of user through the microphone and print that on terminal. Speech recognition is a machine's ability to listen to spoken words and identify them. Convert the speech to text.

3.2.3 Code Converter

This module helps to convert voice input to its python code. It is created with help of some user defined functions. The input is compared with some if statements and if it is a match then the required syntax is displayed on the editor.

3.2.4 Voice translator

The translator that can translate voice input and give translated output generated from it. It is created using google's googleTrans API and speech_recognition library of python. It converts text from one language to another language.

3.3 Algorithm

Speech recognition is the ability of a computer or software to analyze and convert spoken language into a master-readable format. Mechanical telephones and medical dictation systems were among the earliest applications for speech recognition. It is also used in specialized vocabulary careers for transcript, query index, and commanding computer systems. It is also personally liable for automobiles and tablets, such as Apple's Siri.

3.3.1 Speech Recognition

Speech recognition is a machine's ability to listen to spoken words and identify them. You can then use speech recognition in Python to convert the spoken words into text, make a query or give a reply. You can even program some devices to respond to these spoken words. You can do speech recognition in python with the help of computer programs that take in input from the microphone, process it, and convert it into a suitable form.

Speech recognition seems highly futuristic, but it is present all around you. Automated phone calls allow you to speak out your query or the query you wish to be assisted on; your virtual assistants like Siri or Alexa also use speech recognition to talk to you seamlessly.

How Does Speech Recognition work?
Speech recognition in Python works with algorithms that perform linguistic and acoustic modeling. Acoustic modeling is used to recognize phenones/phonetics in our speech to get the more significant part of speech, as words and sentences.

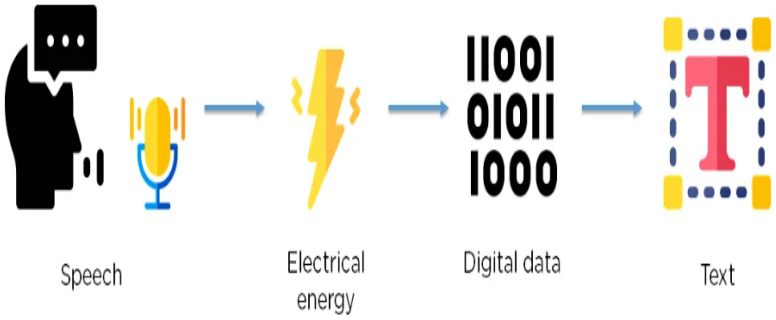


Figure 3.2: Working of Speech Recognition

Speech recognition starts by taking the sound energy produced by the person speaking and converting it into electrical energy with the help of a microphone. It then converts this electrical energy from analog to digital, and finally to text.

It breaks the audio data down into sounds, and it analyzes the sounds using algorithms to find the most probable word that fits that audio. All of this is done using Natural Language Processing and Neural Networks. Hidden Markov models can be used to find temporal patterns in speech and improve accuracy

3.3.2 Natural Language Processing

Natural language processing (NLP) is a subfield of Artificial Intelligence (AI). This is a widely used technology for personal assistants that are used in various business fields/areas. This technology works on the speech provided by the user, breaks it down for proper understanding and processes accordingly. This is a very recent and effective approach due to which it has a really high demand in today's market. Natural Language Processing is an upcoming field where already many transitions such as compatibility with smart devices, interactive talks with a human have been made possible. Knowledge representation, logical reasoning, and constraint satisfaction were the emphasis of AI applications in NLP. Here first it was applied to semantics and later to the grammar. In the last decade, a sig-

nificant change in NLP research has resulted in the widespread use of statistical approaches such as machine learning and data mining on a massive scale. The need for automation is never ending courtesy of the amount of work required to be done these days. NLP is a very favourable, but aspect when it comes to automated applications. The applications of NLP have led it to be one of the most sought-after methods of implementing machine learning. Natural Language Processing (NLP) is a field that combines computer science, linguistics, and machine learning to study how computers and humans communicate in natural language. The goal of NLP is for computers to be able to interpret and generate human language. This not only improves the efficiency of work done by humans but also helps in interacting with the machine. NLP bridges the gap of interaction between humans and electronic devices.

Working

The field is divided into three different parts:

- Speech Recognition—The translation of spoken language into text.
- Natural Language Understanding (NLU) —The computer's ability to understand what we say.
- Natural Language Generation (NLG) —The generation of natural language by a computer.

3.4 Tools

3.4.1 Front end

The use of different kinds of languages is vital for web developers and web designers. Different styles can lead to a higher quality of the site as a whole. One of the most commonly used languages for web development and web designing is Python. Python is a powerful programming language ideal for scripting and rapid application development.

Python is an interpreted, high-level, general-purpose programming language. Python was developed by Guido van Rossum in early 1990's and its latest version is 3.7.1, we can simply call it as Python3. Python 3.0 was released in 2008. and is interpreted language i.e. it's not compiled, and the interpreter will check the code line by line.

Features of python 3 are Python 3 makes coding more obvious and intuitive by removing duplicate constructs and modules. It simplifies multilingual support, with its core string type based on Unicode by default. It makes it easier to swap in any print function, now that Print() is built-in (rather than a statement);Python 3's integer division now automatically promotes to a floating-point result.
Python is also commonly referred to as the "language of the future." It is because of the growing uses of Python each passing day. It has already been in the market for over 25 years now. It is slowly dominating the use of different programming languages such as C++, Java, and C.

3.4.2 Back end

Anaconda offers the easiest way to perform Python/R data science and machine learning on a single machine.Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. The distribution includes data-

science packages suitable for Windows, Linux, and macOS.

Package versions in Anaconda are managed by the package management system conda. This package manager was spun out as a separate open-source package as it ended up being useful on its own and for things other than Python. There is also a small, bootstrap version of Anaconda called Miniconda, which includes only conda, Python, the packages they depend on, and a small number of other packages.

Anaconda is an open-source software that contains Jupyter, spyder, etc that are used for large data processing, data analytics, heavy scientific computing. Anaconda works for R and python programming language. Spyder(sub-application of Anaconda) is used for python. Opencv for python will work in spyder. Package versions are managed by the package management system called conda.

Chapter 4

RESULT ANALYSIS

The system was successfully completed in time. Python code for the given input was generated . A real-time voice translator that can translate voice input and give translated output from it. This technology would be simple to use, requiring less human efforts and hardware resources. It would undoubtedly be beneficial to blind, inexperienced, and infrequent consumers.

Chapter 5

VALIDATION AND TESTING

5.1 Test Plan

Test plan is the project plan for the testing work to be done. It is not a test design specification. A collection of test cases or a set of test procedures: in fact, most of our test plans do not address that level of detail. Many people have different definitions for test plans. First, by writing a test plan it guides our thinking. Writing a test plan forces us to confront the challenges that await us and focus our thinking on important topics. Fred Brooks explains the importance of careful estimation and planning for testing in one of his books as follows: Failure to allow enough time for system test, in particular, is peculiarly disastrous. Since the delay comes at the end of the schedule, no one is aware of schedule trouble until almost the delivery date delay at this point has unusually severe financial repercussions. It is therefore very important to allow enough system test time in the original schedule by using a template for writing test plans helps us remember the important challenges.

Second, the test planning process and the plan itself serve as the means of communication with other members of the project team, testers, peers, managers and other stakeholders. This communication allows the test plan to influence the project team and the project team to influence the test plan, especially in the ar-

eas of organization-wide testing policies and motivations; test scope, objectives and critical areas to test; project and product risks, resource considerations and constraints; and the testability of the item under test. We can complete this communication by circulating one or two test plan drafts and through review meetings. As we keep note or document the answers to these kinds of questions, the test plan becomes a record of previous discussions and agreements between the testers and the rest of the project team.

Third, the test plan helps us to manage change. During early phases of the project, as we gather more information, we revise our plans. As the project evolves and situations change, we adapt our plans. By updating the plan at major milestone helps us to keep testing aligned with project needs. As we run the tests, we make final adjustments to our plans based on the results. You might not have the time or the energy to update your test plans every time a change is made in the project, as some projects can be quite dynamic. At times it is better to write multiple test plans in some situations. For example, when we manage both integration and system test levels, those two test execution periods occur at different points in time and have different objectives. For some systems projects, a hardware test plan and a software test plan will address different techniques and tools as well as different audiences. However, there are chances that these test plans can get overlapped, hence, a master test plan should be made that addresses the common elements of both the test plans can reduce the amount of redundant documentation.

5.1.1 White Box Testing

White box testing is a way of testing the external functionality of the code by examining and testing the program code that realizes the external functionality. Thus, it is also known as clear box testing, glass box testing or open box testing. White box testing considers the program code, code structure, internal design flow. White box testing is classified into static and structural testing. Static testing is a type of testing which requires only the source code of the product, not the binaries or executables. Structural testing considers the program code, code structure, internal design flow and how they coded.

5.1.2 Black Box Testing

Black box testing involves looking into the specification and does not require examining the code of a program. Black box testing is done from the customer's point of view. The test engineer involved in the testing only know a set of inputs and outputs and is unaware of how these inputs are transformed into outputs Black box testing is convenient to administer because it uses the complete finished product and do not require any knowledge of its construction.

5.1.3 Integration Testing

It is the schematic technique for constructing the program structure while at the same time conducting tests to see uncovered errors associated with interfacing. It also tests to find discrepancies between the system and its original objectives, current specification and systems documentation .The primary concern in compatibility of individual modules. Integration testing is a systematic testing to discover errors associated within the interface. The objective is to take unit tested modules and build a program structure. All the modules are combining and tested as a while. We followed bottom-up integration testing .Bottom up integration testing as its name implies begins construction and testing with atomic modules. Because

components are integrated from the bottom up, processing required for component subordinate to give level is always available and the need for stubs is estimated.

5.2 Classes of Tests Formal Test Case

In order to fully test that all the requirements of an application are met, there must be at least two test cases for each requirement: one positive test and one negative test. If a requirement has sub-requirements, each sub-requirement must have at least two test cases. Keeping track of the link between the requirement and the test is frequently done using a traceability matrix. Written test cases should include a description of the functionality to be tested, and the preparation required to ensure that the test can be conducted. A formal written test-case is characterized by a known input and by an expected output, which is worked out before the test is executed. The known input should test a precondition and the expected output should test a post condition.

5.2.1 Informal Test Case

For applications or systems without formal requirements, test cases can be written based on the accepted normal operation of programs of a similar class. In some schools of testing, test cases are not written at all, but the activities and results are reported after the tests have been run. In scenario testing, hypothetical stories are used to help the tester think through a complex problem or system. They can be as simple as a diagram for a testing environment or they could be a description written in prose. The ideal scenario test is a story that is motivating, credible, complex, and easy to evaluate. They are usually different from test cases in that test cases are single steps while scenarios cover several steps of the key.

Chapter 6

REQUIREMENT ANALYSIS AND SPECIFICATION

6.1 REQUIREMENT ANALYSIS/LITERATURE VIEW

Requirement analysis results in the specification of software's operational characteristics; indicates software's interface with other system elements and establishes constraints the software must meet. Requirement analysis allows the software engineer to elaborate on basic requirements established during earlier requirement engineering tasks and build models that depict user scenarios, functional activities, problem classes and their relationships, system and class behavior and flow of data as it is transformed. Requirement analysis provides the software designer with representation of information, function and behavior that can be translated to architectural, interface and component level designs. Finally, the analysis model and the requirement specification provide the development and the customs with the means to access quality once the software is built.

6.2 REQUIREMENT SPECIFICATION

6.2.1 FUNCTIONAL REQUIREMENT

In software engineering (and systems engineering), a functional requirement defines a function of a system and its components. A function is described as a set of inputs, the behavior, and outputs.

Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases. Functional requirements are supported by non-functional requirements (also known as quality requirements), which impose constraints on the design or implementation (such as performance requirements, security, or reliability). Generally, functional requirements are expressed in the form ”system must do ¡requirement¿”, while non-functional requirements are ”system shall be ¡requirement¿”. The plan for implementing functional requirements is detailed in the system design. The plan for implementing non-functional requirements is detailed in the system architecture.

As defined in requirements engineering, functional requirements specify particular results of a system. This should be contrasted with non-functional requirements which specify overall characteristics such as cost and reliability. Functional requirements drive the application architecture of a system, while non-functional requirements drive the technical architecture of a system.

In some cases, a requirements analyst generates use cases after gathering and validating a set of functional requirements. The hierarchy of functional requirements is: user/stakeholder request feature use case business rule. Each use case illustrates behavioral scenarios through one or more functional requirements. Often, though, an analyst will begin by eliciting a set of use cases, from which the analyst can derive the functional requirements that must be implemented to allow a user to perform each use case.

6.2.2 NON-FUNCTIONAL REQUIREMENT

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are contrasted with functional requirements that define specific behavior or functions. The plan for implementing functional requirements is detailed in the system design. The plan for implementing non-functional requirements is detailed in the system architecture.

Broadly, functional requirements define what a system is supposed to do and non-functional requirements define how a system is supposed to be. Functional requirements are usually in the form of ”system shall do ¡requirement¿”, an individual action of part of the system, perhaps explicitly in the sense of a mathematical function, a box description input, output, process and control functional model or IPO Model. In contrast, non-functional requirements are in the form of” system shall be requirement ”an overall property of the system as a whole or of a particular aspect and not a specific function. The system’s overall properties commonly mark the difference between whether the development project has succeeded or failed.

Non-functional requirements are often called ”quality attributes” of a system. Other terms for non-functional requirements are ”qualities”, ”quality goals”, ”quality of service requirements”, ”constraints” and ”non-behavioral requirements”. Informally these are sometimes called the ”ileitis”, from attributes like stability and portability. Qualities–that is non-functional requirements–can be divided into two main categories:

- Execution qualities, such as security and usability, which are observable at run time.
- Evolution qualities, such as testability, maintainability, extensibility and scalability, which are embodied in the static structure of the software system.

6.3 FEASIBILITY STUDY

The most difficult part of feasibility analysis is the identification of the candidate systems and the evaluation of their performance. A feasibility study is conducted to select the best system that meets performance requirements. This entails an identification description, an evaluation of candidate systems, and the selection of the best system for the job. The new system has advantages such as we can easily doing transactions in the shop and this application is more user- friendly for employees.

There are three key considerations are involved in feasibility analysis. They are:

- Technical Feasibility
- Economic Feasibility
- Operational Feasibility

6.3.1 TECHNICAL FEASIBILITY

Technical Feasibility determines whether the technology needed for the proposed system is available and how it can be integrated with in the organization. It centers on the existing computer system and to what extent it can support the proposed addition. It also involves financial considerations to accommodate technical enhancements. The proposed system **Code Now** can be done with the current software technology and available personal.

6.3.2 ECONOMICAL FEASIBILITY

Economic analysis is the most frequently used method for evaluating the effectiveness of a candidate system. More commonly known as cost/benefit analysis, the procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs. If benefits out-weight costs, then

the decision is made to design and implement the system. Otherwise, further justification or alterations in the proposed system will have to be made if it is to have a chance of being approved. This is an ongoing effort that improves in accuracy at each phase of the system life cycle.

6.3.3 OPERATIONAL FEASIBILITY

People are inherently resistant to change, and computers have been known to facilitate change. An estimate should be made of how strong a reaction the user staff is likely have toward the development of a computerized system. It is common knowledge that computer installed have something to do with turnover, transfers, retraining and changes in employee job status. Therefore, it is understandable that the introduction of a candidate system requires special effort to educate, sell, and train the staff on new ways of conducting business. The current staff can be very much used to run the new system by giving proper training. Most of the staff has basic knowledge of a computer and thus training them is easy. Operational feasibility ensures the operational integrity of the system.

Chapter 7

CONCLUSION AND FUTURE SCOPE

7.1 Conclusion

The goal of this project is to develop a model that can generate python code by converting user speech. And also this system translate users speech to any language as per the users choice.

Less typing that results in more coding can't be a bad thing. Developing a computer program is a difficult process that necessitates the use of hardware resources that the user must manage. There is a chance that the user's fingertips will be injured as they continue to type the message. To prevent these issues, we're creating a technology that allows a computer program to be created using voice commands

A voice translator that helps to translate voice commands to any language.

7.2 Future Scope

Many different adaptations, and improvements can be done on this report. Future work concerns more on code conversion. That is it can be developed for multiple programming languages. And also can implement a python terminal to run the code.

.

Chapter 8

APPENDIX

8.1 Source Code

8.1.1 routes.py

```
from msilib.schema import File
from flask import Flask, render_template, request, redirect,send_file, flash, abort,
url_for
from pose import app,db,mail
from pose import app,db,mail
from pose import app
from pose.models import *
# from flask_login import login_user, current_user, logout_user, login_required
from random import randint
import os
from PIL import Image
# from flask_mail import Message
from io import BytesIO
import glob
@app.route('/')
```

```
def index1():
return render_template(”index1.html”)
@app.route(’/det’,methods=[’GET’,’POST’])
def det():
if request.method==”POST”:
import sys
import speech_recognition as sr
import os
import readline
#from sys import exit
import time
import numpy as np
def listen():
r = sr.Recognizer()
with sr.Microphone() as source:
print(”I am listening...”)
audio = r.listen(source)
data = ””
try:
data = r.recognize_google(audio)
print(”You said: ” + data)
except sr.UnknownValueError:
print(”Google Speech Recognition did not understand audio”)
except sr.RequestError as e:
print(”Request Failed; 0”.format(e))
return data
def text2float(textnum, numwords=):
if not numwords:
units = [ ”zero”, ”one”, ”two”, ”three”, ”four”, ”five”, ”six”, ”seven”, ”eight”,
”nine”, ”ten”, ”eleven”, ”twelve”, ”thirteen”, ”fourteen”, ”fifteen”, ”sixteen”, ”sev-
```

enteen”, ”eighteen”, ”nineteen”,]
tens = [””, ””, ”twenty”, ”thirty”, ”forty”, ”fifty”, ”sixty”, ”seventy”, ”eighty”, ”ninety”]
scales = [”hundred”, ”thousand”, ”million”, ”billion”, ”trillion”]
numwords[”and”] = (1, 0)
for idx, word in enumerate(units): numwords[word] = (1, idx)
for idx, word in enumerate(tens): numwords[word] = (1, idx * 10)
for idx, word in enumerate(scales): numwords[word] = (10 ** (idx * 3 or 2), 0)
current = result = 0
for word in textnum.split():
if word not in numwords:
raise Exception(”Illegal word: ” + word)
scale, increment = numwords[word]
current = current * scale + increment
if scale ¿ 100:
result += current
current = 0
return float(result + current)
def write_code(data):
listening = ”The System did not understand audio”
global output
data = data.split(” ”)
data = [x.lower() for x in data]
if ”program” in data:
output = create_pgm(data)
return output
if ”function” in data:
output = create_fun(data)
return output
if ”list” in data:

```
output = create_list(data)
return output
if ”set” in data:
output = create_set(data)
return output
if ”import” in data:
output = create_lib(data)
return output
if ”print” in data:
listening = ”Cannot recognize voice”
output = create_print(data)
return output
if ”accept” in data:
output = accept_value(data)
return output
if ”class” in data:
output=create_cls(data)
return output
if ”syntax” in data:
output=dis_syn(data)
return output
if ”stop listening” in data:
output=’Listening stopped’
return output
return listening
def create_cls(data):
if ”class” in data:
indices = [i+1 for i, s in enumerate(data) if ’class’ in s]
for i in indices:
var=data[i]
```

```
d=”class ”+var+”:”
return d
def create_pgm(data):
if ”add” in data:
if ”two” in data:
with open(’add_two.txt’,’r’) as f:
pgm=f.read()
return pgm
if ”three” in data:
with open(’add_three.txt’,’r’) as f:
pgm=f.read()
return pgm
if ”area” in data:
if ”triangle” in data:
with open(’tri_area.txt’,’r’) as f:
pgm=f.read()
return pgm
if ”swap” in data:
with open(’swap.txt’,’r’) as f:
pgm=f.read()
return pgm
def dis_syn(data):
if ”if” in data:
if ”simple” in data:
with open(’simple_if.txt’,’r’) as f:
pgm=f.read()
return pgm
if ”else” in data:
with open(’else_if.txt’,’r’) as f:
pgm=f.read()
```

```
return pgm
if ”loop” in data:
if ”for” in data:
with open(’forloop.txt’,’r’) as f:
pgm=f.read()
return pgm
if ”while” in data:
with open(’whileloop.txt’,’r’) as f:
pgm=f.read()
return pgm
def create_fun(data):
if ’function’ in data:
indices = [i+1 for i, s in enumerate(data) if ’function’ in s]
for i in indices:
var=data[i]
d=”def ”+var+”():”
return d
def accept_value(data):
if ’integer’ in data:
indices = [i+1 for i, s in enumerate(data) if ’variable’ in s]
for i in indices:
var=data[i]
d= var+ ’ = int(input(”Enter the number : ”))’
return d
if ’string’ in data:
indices = [i+1 for i, s in enumerate(data) if ’variable’ in s]
for i in indices:
var=data[i]
d= var+ ’ = input(”Enter the string : ”)’
return d
```

```
if ’float’ in data:
indices = [i+1 for i, s in enumerate(data) if ’variable’ in s]
for i in indices:
var=data[i]
d= var+ ’ = float(input(”Enter the string : ”))’
return d
def create_list(data):
l = list()
if ’empty’ in data:
indices = [i+1 for i, s in enumerate(data) if ’list’ in s]
for i in indices:
var=data[i]
if ’variable’ in data:
indices = [i+1 for i, s in enumerate(data) if ’variable’ in s]
for i in indices:
var=data[i]
if ’integer’ in data:
indices = [i+1 for i, s in enumerate(data) if ’integer’ in s]
for i in indices:
try:
l.append(int(data[i]))
except ValueError:
l.append(text2float(data[i]))
if ’float’ in data:
indices = [i+1 for i, s in enumerate(data) if ’float’ in s]
for i in indices:
l.append(float(data[i]))
if ’string’ in data:
indices = [i+1 for i, s in enumerate(data) if ’string’ in s]
for i in indices:
```

```
l.append(data[i])
# print(var,”=”,l)
d=var+”=”+str(l)
return d
def create_set(data):
s=[]
if ’empty’ in data:
indices = [i+1 for i, s in enumerate(data) if ’tuple’ in s]
for i in indices:
var=data[i]
if ’variable’ in data:
indices = [i+1 for i, s in enumerate(data) if ’variable’ in s]
for i in indices:
var=data[i]
if ’integer’ in data:
indices = [i+1 for i, s in enumerate(data) if ’integer’ in s]
for i in indices:
try:
s.append(int(data[i]))
except ValueError:
s.append(text2float(data[i]))
if ’float’ in data:
indices = [i+1 for i, s in enumerate(data) if ’float’ in s]
for i in indices:
s.append(float(data[i]))
if ’string’ in data:
indices = [i+1 for i, s in enumerate(data) if ’string’ in s]
for i in indices:
s.append(data[i])
# print(var,”=”,l)
```

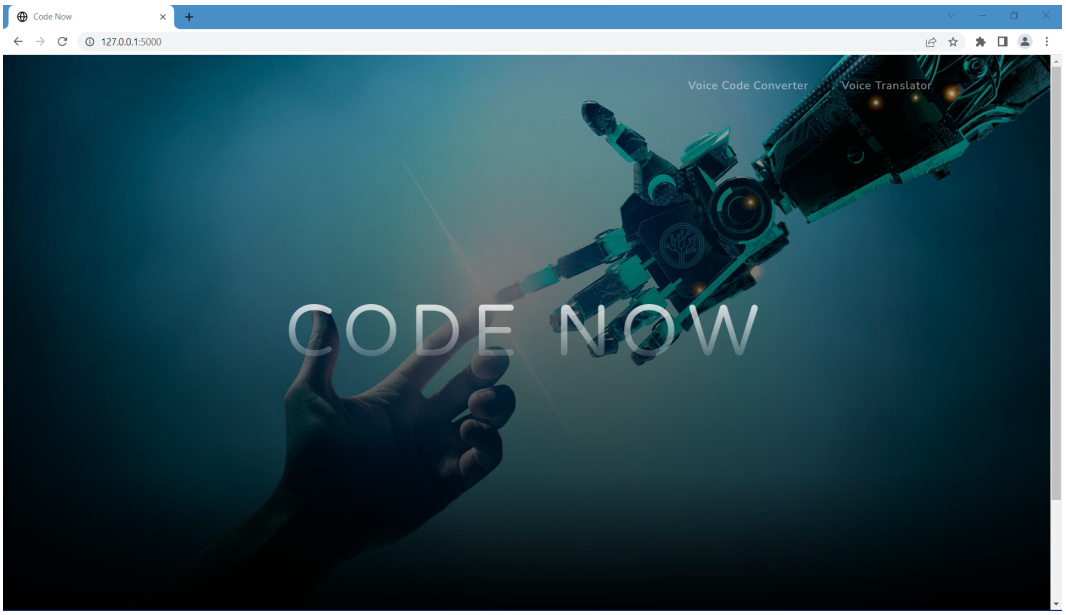
```
d=var+”= set(”+str(s)+”)”
return d
def create_lib(data):
global lib
if ’import’ in data:
indices = [i+1 for i, s in enumerate(data) if ’library’ in s]
for i in indices:
lib=data[i]
d= ”import” +” ”+lib
return d
def create_print(data):
if ’print’ in data:
indices = [i+1 for i, s in enumerate(data) if ’print’ in s]
for i in indices:
word=data[i]
d=”print(”+” ’”+word+”’)”
return d
time.sleep(2)
data = listen()
out=write_code(data)
return render_template(”output.html”,out=out,data=data)
return render_template(”up_image1.html”)

    @app.route(’/trans’,methods=[’GET’,’POST’])
def trans():
if request.method==”POST”:
# from playsound import playsound
import speech_recognition as sr
from googletrans import Translator
from gtts import gTTS
```

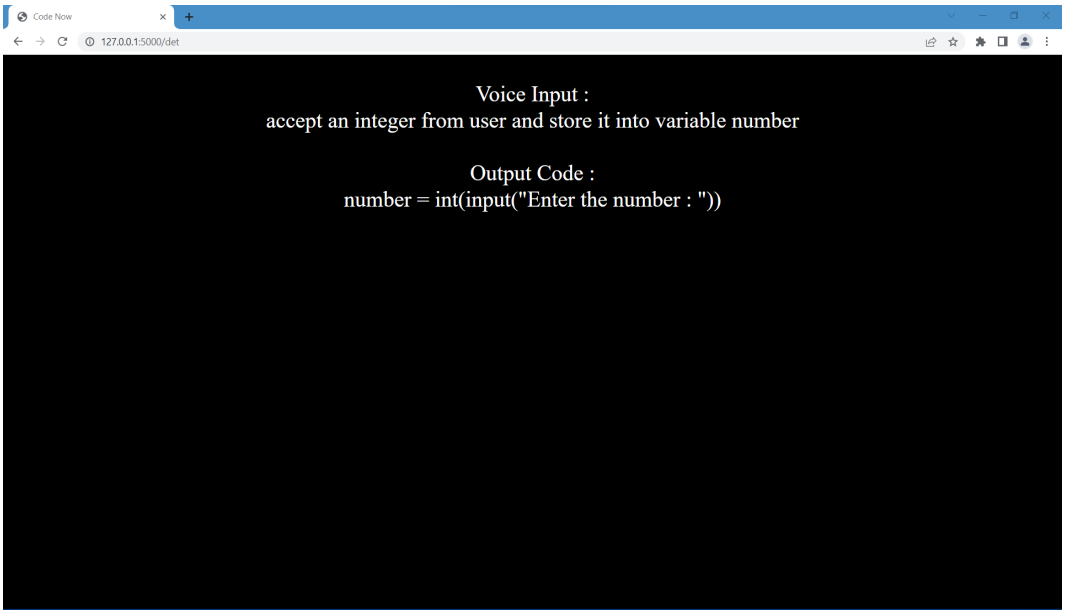
import os
dic=('afrikaans', 'af', 'albanian', 'sq', 'amharic', 'am','arabic', 'ar', 'armenian',
'hy', 'azerbaijani', 'az', 'basque', 'eu', 'belarusian', 'be', 'bengali', 'bn', 'bosnian','bs',
'bulgarian', 'bg', 'catalan', 'ca','cebuano', 'ceb', 'chichewa', 'ny', 'chinese (simplified)', 'zh-cn', 'chinese (traditional)', 'zh-tw','corsican', 'co', 'croatian', 'hr',
'czech', 'cs', 'danish', 'da', 'dutch', 'nl', 'english', 'en', 'esperanto','eo', 'estonian', 'et', 'filipino', 'tl', 'finnish', 'fi', 'french', 'fr', 'frisian', 'fy', 'galician',
'gl','georgian', 'ka', 'german', 'de', 'greek', 'el', 'gujarati', 'gu', 'haitian creole', 'ht', 'hausa', 'ha', 'hawaiian', 'haw', 'hebrew', 'he', 'hindi', 'hi', 'hmong',
'hmn', 'hungarian', 'hu', 'icelandic', 'is', 'igbo', 'ig', 'indonesian', 'id', 'irish',
'ga', 'italian', 'it', 'japanese', 'ja', 'javanese', 'jw', 'kannada', 'kn', 'kazakh',
'kk', 'khmer', 'km', 'korean', 'ko', 'kurdish (kurmanji)', 'ku', 'kyrgyz', 'ky',
'lao', 'lo', 'latin', 'la', 'latvian', 'lv', 'lithuanian', 'lt', 'luxembourgish', 'lb',
'macedonian', 'mk', 'malagasy', 'mg', 'malay', 'ms', 'malayalam', 'ml', 'maltese', 'mt', 'maori', 'mi', 'marathi', 'mr', 'mongolian', 'mn', 'myanmar (burmese)',
'my', 'nepali', 'ne', 'norwegian', 'no', 'odia', 'or', 'pashto', 'ps', 'persian', 'fa',
'polish', 'pl', 'portuguese', 'pt', 'punjabi', 'pa', 'romanian', 'ro', 'russian', 'ru',
'samoan', 'sm', 'scots gaelic', 'gd', 'serbian', 'sr', 'sesotho', 'st', 'shona', 'sn',
'sindhi', 'sd', 'sinhala', 'si', 'slovak', 'sk', 'slovenian', 'sl', 'somali', 'so', 'spanish', 'es', 'sundanese', 'su', 'swahili', 'sw', 'swedish', 'sv', 'tajik', 'tg', 'tamil',
'ta', 'telugu', 'te', 'thai', 'th', 'turkish', 'tr', 'ukrainian', 'uk', 'urdu', 'ur', 'uyghur',
'ug', 'uzbek', 'uz', 'vietnamese', 'vi', 'welsh', 'cy', 'xhosa', 'xh', 'yiddish', 'yi',
'yoruba', 'yo', 'zulu', 'zu')
def takecommand():
r = sr.Recognizer()
with sr.Microphone() as source:
print("listening.....")
r.pause_threshold = 1
audio = r.listen(source)
try:

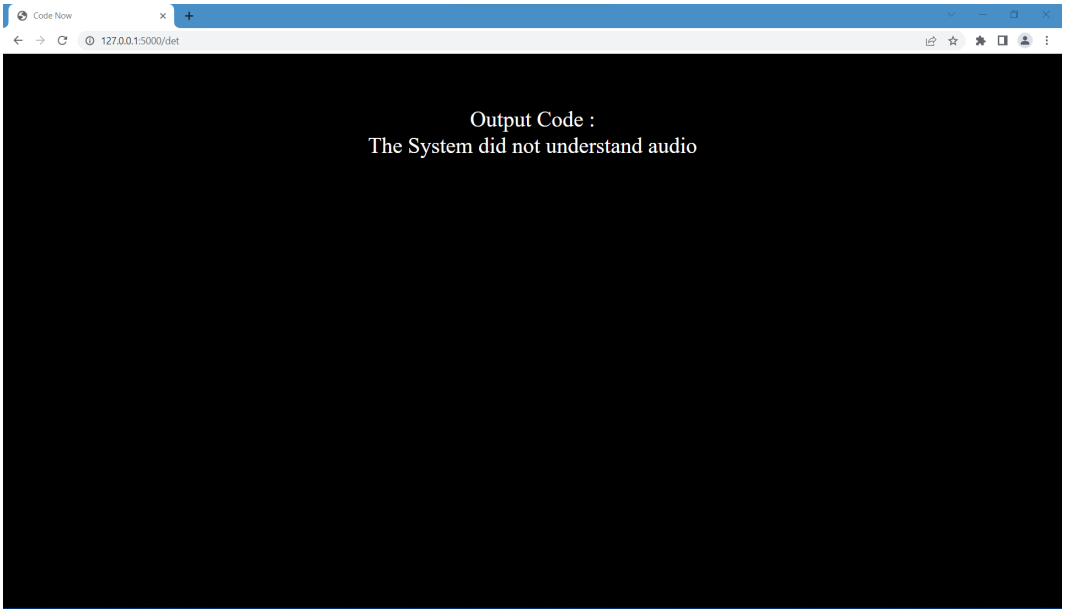
```
print(”Recognizing.....”)
query = r.recognize_google(audio, language=’en-in’)
print(f”user said query”)
except Exception as e:
print(”say that again please.....”)
return ”None”
return query
query = takecommand()
while (query == ”None”):
query = takecommand()
def destination_language():
global text
print(”Enter the language in which you want to convert  Ex. Hindi , English , etc.”)
print()
Input destination language in which the user
wants to translate
to_lang = takecommand()
while (to_lang == ”None”):
to_lang = takecommand()
to_lang = to_lang.lower()
return to_lang
to_lang = destination_language()
lang=to_lang
# Mapping it with the code
while (to_lang not in dic):
print(”Language in which you are trying to convertcurrently not available ,please input some other language”)
print()
to_lang = destination_language()
```

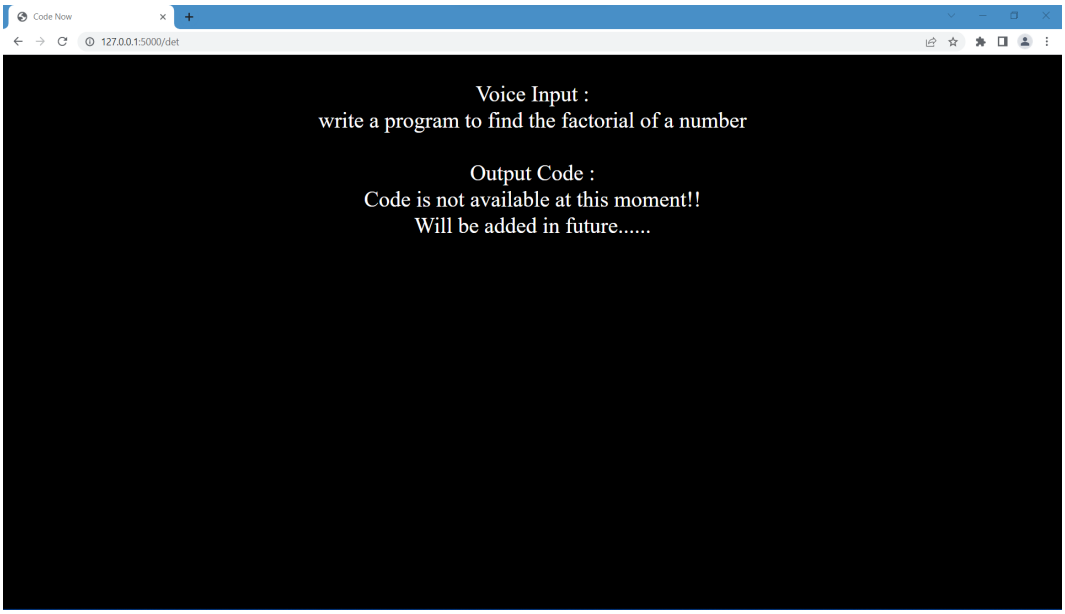
```
to_lang = dic[dic.index(to_lang)+1]
# invoking Translator
translator = Translator()
# Translating from src to dest
text_to_translate = translator.translate(query, dest=to_lang)
text = text_to_translate.text
return render_template(”output1.html”,query=query,text=text,lang=lang)
return render_template(”trans.html”)
```

8.2 Screen Shots

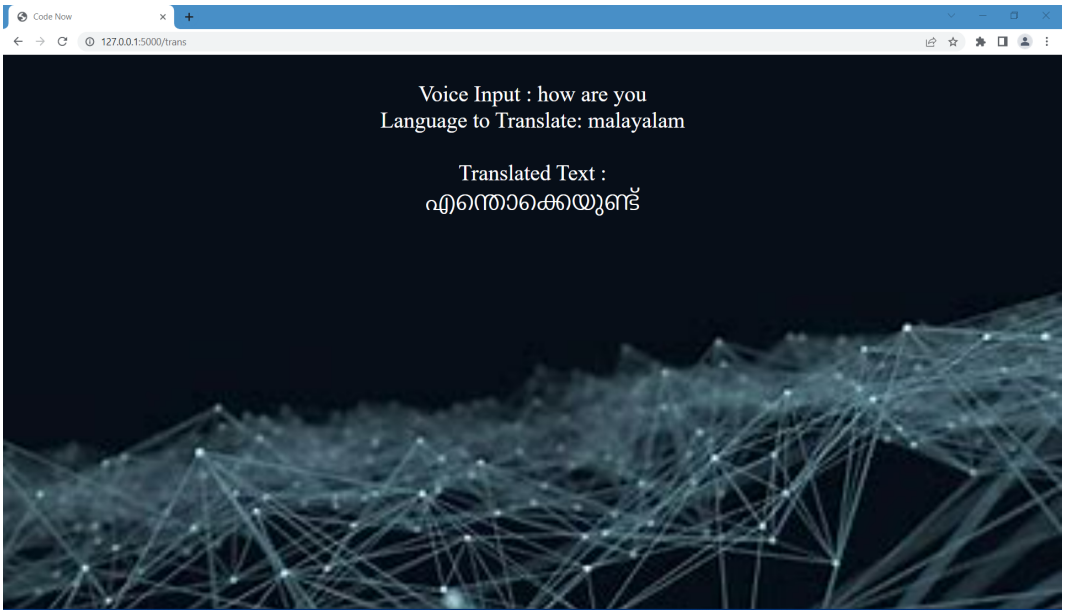



Code Now
127.0.0.1:5000/det
Voice Input :
accept an integer from user and store it into variable number
Output Code :
number = int(input("Enter the number : "))


Code Now
127.0.0.1:5000/det
Output Code :
The System did not understand audio


Code Now
127.0.0.1:5000/det
Voice Input :
write a program to find the factorial of a number
Output Code :
Code is not available at this moment!!
Will be added in future.....


Code Now
127.0.0.1:5000/det
Voice Input :
stop listening
Listening Stopped.....!!


Code Now
127.0.0.1:5000/trans
Voice Input : how are you
Language to Translate: malayalam
Translated Text :
എന്തൊക്കെയുണ്ട്

Chapter 9

REFERENCES

- https://machinelearningknowledge.ai/create-ai-voice-assistant-with-speech-recognition-python-project-source-code/
- https://www.activestate.com/blog/how-to-use-ai-to-write-code-for-you
- https://www.geeksforgeeks.org/create-a-real-time-voice-translator-using-python
- https://www.irjmets.com/uploadedfiles/paper/ volume3/issue_1_january_2021/5762/1628083235.pd
- https://www.semanticscholar.org/paper/Voice-Coding-Using-AI-Devshatwar/8abd8c61f9f0c6b62618b9c4e805916b6d10a672
- https://en.wikipedia.org/wiki/Artificial_intelligence