A REPORT OF ONE MONTH TRAINING

at

THINKNEXT TECHNOLOGIES PRIVATE LIMITED

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY

(Computer Science and Engineering)

JUNE-JULY, 2025

SUBMITTED BY:

NAME: MOHD SHEHZAD

UNIVERSITY ROLL NO: 2302612

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GURU NANAK DEV ENGINEERING COLLEGE, LUDHIANA

(An Autonomous College Under UGC ACT)

ThinkNEXT Technologies Private Limited

ThinkNEXT®

Innovation at every step...

ISO 9001:2015 Certified Company

Indian Iconic AWARD WINNER 2024

Business Excellence AWARD WINNER 2023

Iconic Business Summit AWARDS WINNER 2021

Nation's Business Pride AWARDS WINNER 2021

National Gratitude AWARDS WINNER 2020

Asia's Quality & Entrepreneurship AWARDS WINNER 2019

Business Leaders AWARDS WINNER 2019

NATIONAL ICON AWARDS WINNER 2018

Scan and verify your Certificate

Certificate ID:939895

Ref.No. TNT/C-25/16718

Certificate

Certificate of Excellence

This Certificate do hereby recognizes that

Mohd Shehzad S/o Mohd Munavar

has successfully completed Industrial Training Program

from *23rd June 2025* to *23rd July 2025*

in *AI & ML using Python* Grade *A*

ISO Certified

Member of Confederation of Indian Industry

ISTQB International Software Testing Qualifications Board Affiliated

For ThinkNEXT Technologies Pvt. Ltd.

Authorised Signatory

Member Training Division

ThinkNEXT Technologies Pvt. Ltd.

Director

Director

Corporate Office: S.C.F 113, Sector 65, Mohali

A Outstanding 100-90% | B Excellent 89-80% | C Very Good 79-70% | D Good 69-60% | E Satisfactory 59-50%

GURU NANAK DEV ENGINEERING COLLEGE, LUDHIANA

CANDIDATE'S DECLARATION

I "**MOHD SHEHZAD**" hereby declare that I have undertaken one month training "**THINKNEXT TECHNOLOGIES PRIVATE LIMITED**" during a period from **23 June** to **23 July** in partial fulfillment of requirements for the award of degree of B. Tech (Computer Science & Engineering) at GURU NANAK DEV ENGINEERING COLLEGE, LUDHIANA. The work which is being presented in the training report submitted to Department of Computer Science & Engineering at GURU NANAK DEV ENGINEERING COLLEGE, LUDHIANA is an authentic record of training work

Signature of the Student

The one-month industrial training Viva–Voce Examination of_____ has been held on

_____ and accepted.

Signature of Internal Examiner Signature of External Examiner

ABSTRACT

This report summarizes the one-month industrial training I undertook at **ThinkNext Technologies Pvt. Ltd.**, Mohali, during June-July 2025. The focus of my training was in the **AI & Python Programming Department**, where I worked on real-time voice processing and sentiment analysis using **Python 3.10+**.

I developed a mini-project — **Jarvis**, an AI voice assistant that can listen to human speech, analyze its sentiment using **TextBlob**, and respond with appropriate actions or feedback using **pyttsx3**. Through this project, I learned to handle speech-to-text and text-to-speech operations, apply **Natural Language Processing**, and integrate modular code structures. The experience enhanced my practical skills in Python, debugging, Git version control, and project organization.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to the **faculty of Guru Nanak Dev Engineering College, Ludhiana**, for their continuous support and guidance throughout my academic journey.

My heartfelt thanks to **Think Next Technologies Pvt. Ltd.**, Mohali, for providing me with the opportunity to undergo industrial training and gain hands-on experience with AI-based development tools.

I extend special appreciation to my trainers and mentors for their valuable insights and assistance during the project. I am also thankful to the online developer communities and documentation platforms which played a critical role in helping me debug and improve my project.

ABOUT THE COMPONY / NDUSTRY / INSTITUTE

ThinkNEXT Technologies Pvt. Ltd. is a renowned ISO 9001:2015 certified software development and industrial training company located in **Sector-65, Mohali (Chandigarh), Punjab, India**. Recognized as one of the leading organizations in North India, ThinkNEXT has made a significant mark in the field of **Information Technology (IT)** and **technical education services**, especially in delivering **skill-based industrial training** for engineering, diploma, and management students.

Founded with a mission to bridge the gap between academic knowledge and industry requirements, ThinkNEXT Technologies has consistently delivered quality training programs that align with current technological advancements. The company is backed by a team of experienced developers, trainers, and industry experts, offering real-time project exposure and hands-on learning environments to its trainees.

The company specializes in a wide range of domains such as **Artificial Intelligence (AI), Machine Learning (ML), Data Science, Full Stack Development, Digital Marketing, and Web Technologies**. During the training, students are mentored on real-world use cases and encouraged to build practical projects, preparing them for industrial roles.

The company's **official website** is www.thinknext.co.in, where detailed information about its training programs, success stories, and certifications is available.

In this training, the company played a vital role in enhancing my technical understanding of the **AI/ML domain**, provided me with valuable mentorship, and enabled me to build a meaningful project as part of my summer internship.

LIST OF FIGURES

LIST OF TABLES

DEFINITIONS, ACRONYMS and ABBREVIATIONS

- **AI** – Artificial Intelligence: The simulation of human intelligence in machines programmed to think and learn.
- **ML** – Machine Learning: A subset of AI that enables systems to learn and improve from data without being explicitly programmed.
- **NLP** – Natural Language Processing: A field of AI that focuses on the interaction between computers and human (natural) languages.
- **TTS** – Text-to-Speech: Technology that converts written text into spoken voice output.
- **STT** – Speech-to-Text: Technology that converts spoken language into text.
- **Jarvis** – The name given to the AI Assistant created during the training (inspired by Marvel's J.A.R.V.I.S).
- **Voice Assistant** – A digital assistant that uses voice recognition, speech synthesis, and NLP to perform tasks or answer questions.
- **Sentiment Analysis** – A technique used to determine the emotional tone behind words.
- **Linear Regression** – A machine learning algorithm used for predicting numeric values based on input data.
- **Pyttsx3** – A Python library used for converting text to speech.
- **TextBlob** – A Python library for processing textual data and performing sentiment analysis.
- **SpeechRecognition** – A Python package for performing speech recognition.
- **IDE** – Integrated Development Environment: A software application for programming (e.g., VS Code).
- **CMD / Console** – Command-line interface used to run Python code and view program output.

CONTENTS

CHAPTER 1: INTRODUCTION

INTRODUCTION :

Artificial Intelligence (AI) is no longer a futuristic concept—it is increasingly shaping modern technological applications across industries such as healthcare, customer service, education, and smart home automation. Among its rapidly evolving domains, **voice-based interaction systems** stand out as a crucial link between human communication and intelligent computing. The goal of such systems is to create seamless, natural, and intelligent human-computer interaction using voice, removing the need for keyboard or screen-based input. As the world moves towards touchless and context-aware technology, understanding **speech recognition**, **natural language processing (NLP)**, and **voice synthesis** becomes imperative for future engineers.

This report is based on a one-month industrial training undertaken at **ThinkNext Technologies Pvt. Ltd., Mohali**, focusing on **Python-based voice assistant development** using AI and NLP techniques. The training emphasized developing a functional voice assistant named **"Jarvis"** — a modular, interactive application capable of listening to voice commands, analyzing user sentiment, and performing context-aware actions. This project allowed me to engage deeply with a real-world AI application that mirrors commercial systems like Alexa, Siri, and Google Assistant — but at a foundational, programmable level suited for learning and extension.

1.1 Background and Significance

Speech is the most natural form of communication for humans. Replicating this in machines requires multidisciplinary knowledge: signal processing for recognizing speech, linguistic analysis for understanding context, and computational models to interpret, respond, or act upon commands. With AI capabilities becoming increasingly accessible through open-source libraries, even entry-level engineers can now design systems capable of basic human-like interaction.

The **"Jarvis"** project reflects this accessibility. Rather than relying on massive datasets or complex neural networks, it was designed using open Python libraries like:

- **SpeechRecognition** for real-time voice input
- **pyttsx3** for text-to-speech (TTS) feedback
- **TextBlob** for NLP and sentiment analysis
- **NumPy** for data handling and logical operations

The assistant supports a small but functional set of commands such as opening websites, responding to greetings, taking notes, and identifying user mood.

1.2 Tools and Technologies Used

The training project was built primarily on the **Python 3.10+ environment**, chosen for its simplicity and the vast ecosystem of AI tools. The development was conducted using **Visual Studio Code**, a popular code editor with integrated Git support and terminal capabilities. Key components included:

- **SpeechRecognition**: A library that interfaces with Google Speech API and CMU Sphinx to convert spoken audio into text. Used to capture and process microphone input in real time.
- **pyttsx3**: A text-to-speech conversion library in Python. It works offline and allows configuration of voice rate, volume, and language. Used to deliver spoken feedback from the assistant.
- **TextBlob**: A simplified NLP toolkit built on top of NLTK and Pattern, providing functionalities like part-of-speech tagging, noun phrase extraction, and sentiment analysis.
- **Git**: Version control was integrated using Git, allowing consistent tracking of changes and debugging during the training.

Other supporting tools included:

- **pip and requirements.txt** for dependency management.
- **Command line interface (CLI)** for script execution and test logging.

1.3 Importance in Modern Applications

Building such an assistant isn't just a theoretical exercise — it is aligned with industry trends. Smart assistants are now part of:

- **Call centers and help desks** (AI-driven IVR systems),
- **Healthcare** (voice-enabled patient documentation),
- **Home automation** (controlling lights, music, thermostats),

- **Accessibility solutions** (for visually or physically impaired users).

Through this training, I gained insight into how these systems operate under the hood — recognizing that while enterprise-level systems use large datasets and machine learning models, the foundational logic and integration can still be explored at a smaller scale using procedural programming and existing APIs.

1.4 Learning Objectives

The training aimed to:

- Provide practical exposure to speech interface development
- Understand the workflow of a voice command system
- Apply NLP for sentiment classification
- Modularize program code for scalability and debugging
- Gain confidence in using Git for project versioning

CHAPTER 2: TRAINING WORK UNDERTAKEN

TRAINING WORK UNDERTAKEN

During the four-week training program at **ThinkNext Technologies Pvt. Ltd.**, I engaged in a structured, skill-building pathway designed to strengthen my understanding of Python development and AI-based voice interfaces. The training combined theoretical discussions with practical assignments, culminating in the development of a **voice-controlled virtual assistant** named *Jarvis*. Each week had clearly defined learning goals and deliverables, as detailed below.

2.1 Week 1: Environment Setup and Basic Voice Processing

The first week focused on setting up the development environment and getting familiar with the core libraries required for the project. Activities included:

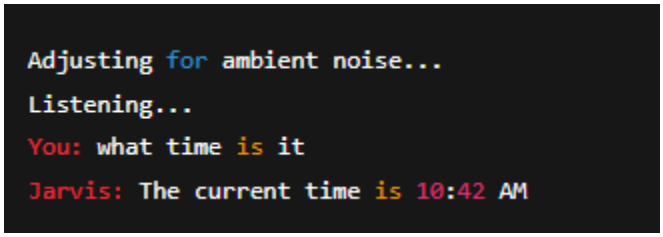
- Installing **Python 3.10+**, **Visual Studio Code**, and the **Git** version control system.
- Learning to create and manage virtual environments in Python using venv.
- Installing Python libraries like SpeechRecognition, pyttsx3, pyaudio, and TextBlob using pip.
- Writing test scripts for capturing voice input and converting it to text

Sample logic included initializing the recognizer and microphone:

```
def listen(self):
    """Improved speech recognition with better error handling"""
```

```
with sr.Microphone() as source:
    print("Adjusting for ambient noise...")
    self.recognizer.adjust_for_ambient_noise(source, duration=1)
    print("Listening...")
```

This phase helped me understand how audio signals are converted into meaningful text using APIs like **Google Web Speech API**.



[Figure 2.1: Screenshot of Voice-to-Text Output Using SpeechRecognition]

2.2 Week 2: Voice Command Processing and Task Mapping

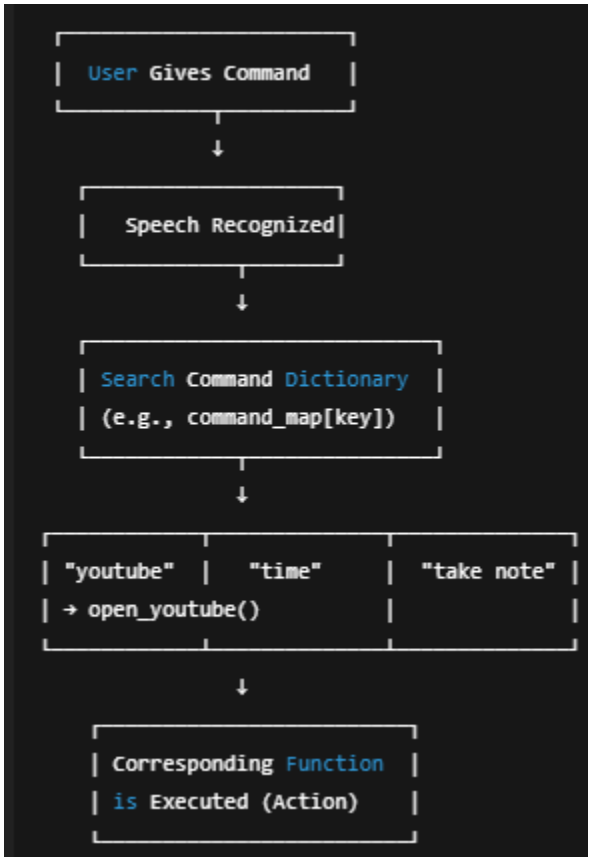
With voice input successfully captured, the second week focused on building the command-processing engine. I wrote condition-based functions to map certain phrases to system actions, such as opening YouTube, telling the time, or replying with a spoken phrase.

Tasks included:

- Writing Python functions for individual commands (e.g., open_youtube(), tell_time()).
- Designing a routing structure based on if-elif blocks and keyword matching.

- Implementing voice feedback using **pyttsx3** with configuration for voice rate, pitch, and volume.

A key implementation detail was the use of a **command dictionary** that mapped recognized keywords to functions, improving efficiency and modularity. This prevented lengthy nested conditionals and supported extensibility.



[Figure 2.2: Flow Diagram of Command Mapping in Jarvis Assistant]

2.3 Week 3: NLP and Sentiment Analysis with TextBlob

The third week introduced **Natural Language Processing**. Using the **TextBlob** library, I developed a sentiment analysis module that allowed Jarvis to assess the emotional tone of user input. This functionality added a layer of human-like responsiveness.

Steps included:

- Installing TextBlob and downloading NLP corpora.
- Creating functions like get sentiment() to evaluate input text and return polarity values.
- Developing respond_to_sentiment() to map sentiment scores into positive, negative, or neutral categories.

This introduced the use of **Equation 2.1**, representing the sentiment score:

Equation

S=polarity(TextBlob(input_text))

Where:

- **S > 0.1**: Positive
- **S < -0.1**: Negative
- **Otherwise**: Neutral

This stage required careful tuning, especially for borderline sentiment cases where users used sarcastic or ambiguous phrases. Simple logic filters were added to improve accuracy.

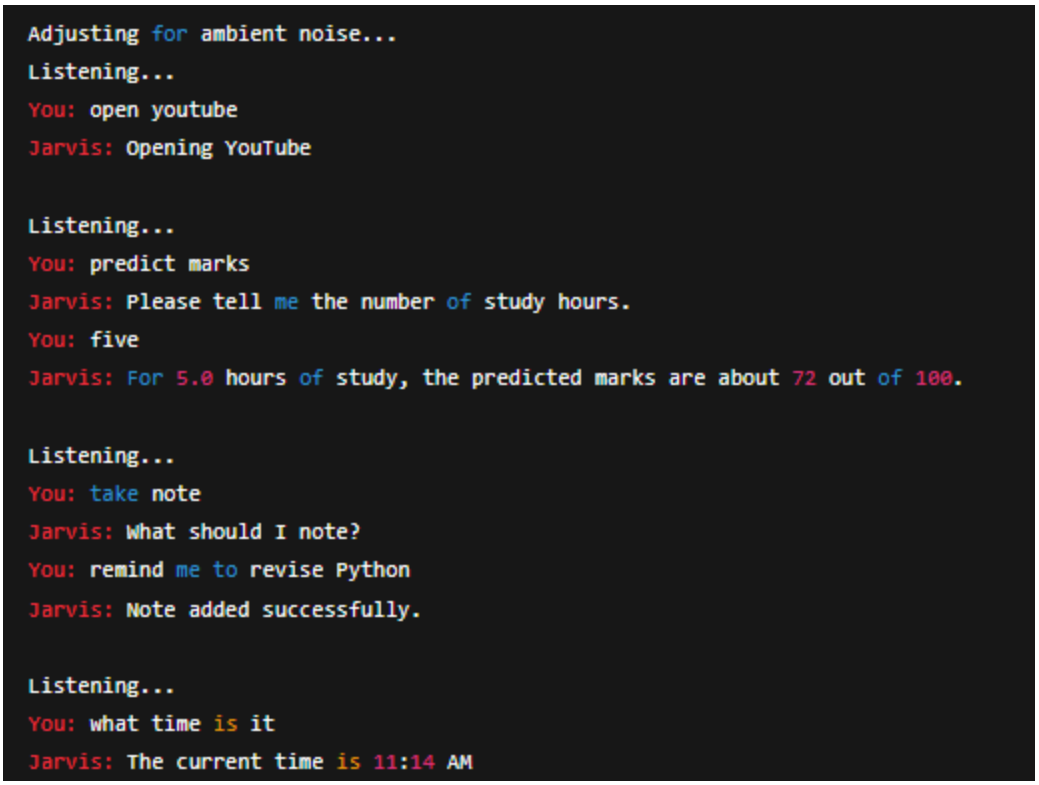
2.4 Week 4: Testing, Debugging, and Final Assembly

The final week of training was dedicated to integration, debugging, and testing the assistant's functionality under different conditions. Activities included:

- Combining all modules into a master control script.
- Handling exceptions like speech_recognition.UnknownValueError and network timeouts.
- Testing with different microphone inputs and varying user accents.
- Adding a feedback mechanism to detect when no command was matched.

Key improvements made during this phase:

- Centralized command error handling via try-except blocks.
- Use of a **logging module** to track system behavior.
- Packaging the code using a requirements.txt file for portability.



[Figure 2.3: Final Testing Console Output of Jarvis Assistant]

2.5 Weekly Summary Table

To summarize the progression of learning, the following table captures week-wise goals and outcomes:

Table 2.1: Weekly Training Schedule Overview

Week	Learning Objectives	Key Tools/Libraries	Outcome
Week 1	Setup development environment, capture voice input	Python, pip, SpeechRecognition	Working audio-to-text module
Week 2	Process commands, add voice feedback	pyttsx3, Git, conditional logic	Mapped voice commands to actions

Week 3	Analyze sentiment using NLP	TextBlob, NumPy	Emotion-based voice response added
Week 4	Test, debug, integrate all components	VS Code, exception handling, logging	Fully functional Jarvis assistant

CHAPTER 3: RESULTS AND DISCUSSION

RESULTS AND DISCUSSION

The final output of the training was a **functional AI-powered voice assistant**, named *Jarvis*, developed entirely in Python. Jarvis was capable of recognizing human voice, converting it into text, analyzing the sentiment of spoken phrases, and providing appropriate responses or actions.

This chapter elaborates on the **functional behavior**, **user interaction patterns**, and the **technical performance** of the application. Additionally, it discusses bugs encountered, features tested, and system limitations addressed during the debugging phase.

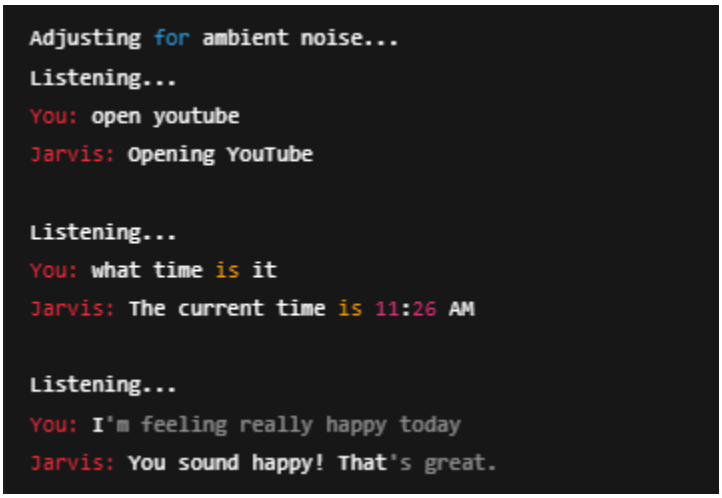
3.1 Functional Overview

The final version of Jarvis performed the following key functions reliably:

1. **Voice Input Recognition**: Using the SpeechRecognition library, Jarvis could process natural language speech from a microphone in real-time and convert it into text. This required continuous listening with low-latency feedback and exception handling for background noise.
2. **Command Mapping:** Recognized text was passed into a routing function that compared keywords with pre-defined command dictionaries. The assistant could handle commands such as:
    - "Open YouTube"

- “Tell me the time”
- “Take a note”
- “What’s the weather?” *(demo version, local response only)*

3. **Text-to-Speech Output**: With the pyttsx3 module, Jarvis responded using voice feedback. The voice engine was tuned for rate (150 wpm) and pitch to sound natural.
4. **Sentiment Analysis**: The TextBlob NLP library allowed the assistant to analyze the polarity of user phrases and respond empathetically.



[Figure 3.1: Voice Command Execution Console Output]

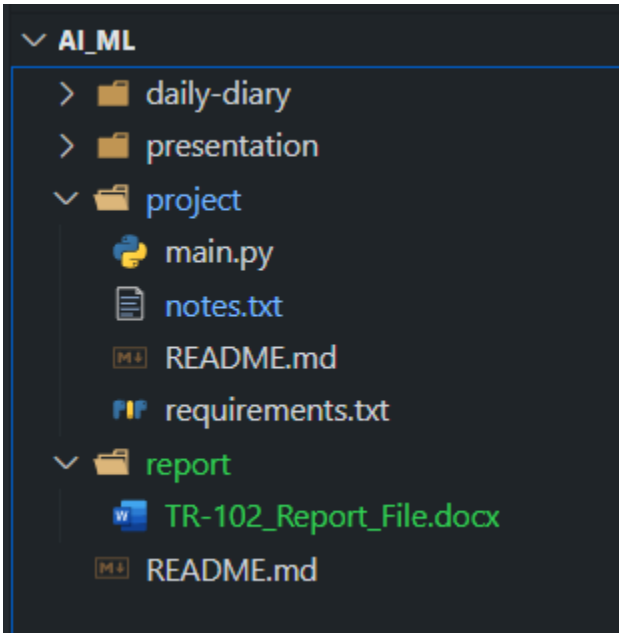
3.2 Code Modularity and Structure

The assistant was developed with maintainability and modularity in mind. All key functions such as listen_command (), respond_to_sentiment (), execute command() were stored in separate Python files and imported using import statements.

Modular design helped during:

- Code debugging
- Testing individual features
- Enhancing code reusability
- Reducing circular dependencies

A sample modular structure:



[Figure 3.2: Module Structure]

3.3 Testing & Error Handling Observations

The system underwent continuous testing under different conditions. Below are key findings from test iterations:

[table 3.1 – Exception Trace in Console During Sentiment Testing]

Test Case	Observation	Resolution
Ambient noise interference	Microphone picked random sounds as commands	Added pause_threshold, noise calibration
API misinterpretation of accent	Misheard “YouTube” as “you took”	Used fuzzy matching and longer phrases
pyttsx3 engine hang after long sessions	TTS engine froze after multiple continuous responses	Restarted engine after each major block
Unrecognized sentiment (ambiguous input)	TextBlob returned near-zero polarity	Used fallback neutral response

A **logging mechanism** was also implemented using Python’s logging module to record issues and session results, improving traceability and future debugging efforts.

3.4 Limitations and System Constraints

Despite its functionality, the system had several known limitations:

- **Accent Sensitivity**: Non-standard English accents sometimes reduced accuracy of recognition.
- **API Dependency**: Google’s Speech API requires internet access, restricting offline functionality.
- **Scalability**: The system handled a small fixed set of commands and lacked a dynamic learning capability.

- **No GUI**: It operated purely via CLI, which limited user interaction and usability.

These constraints were noted for future expansion and enhancement.

3.5 Results Summary

To summarize the final system capabilities:

Table 3.2: Final Output Features of Jarvis Assistant

Feature	Implementation	Status
Voice Input	SpeechRecognition + Microphone	Implemented
Voice Feedback	pyttsx3 Engine	Implemented
Sentiment Response	TextBlob	Implemented
Command Execution	Mapped Logic Dictionary	Implemented
Error Handling	Try/Except + Logging	Implemented
GUI	—	Not Implemented

CHAPTER 4: CONCLUSION AND FUTURE SCOPE

4.1 Conclusion

The one-month industrial training at ThinkNext Technologies Pvt. Ltd., Mohali provided valuable hands-on experience in Artificial Intelligence using Python programming. By developing the *Jarvis Voice Assistant*, I gained practical knowledge in speech recognition, text-to-speech systems, and sentiment analysis using libraries like SpeechRecognition, pyttsx3, and TextBlob.

This training enhanced my ability to write modular code, solve integration challenges, and debug real-time systems. It also gave me insight into how real-world AI applications are structured, tested, and deployed in industry settings. The project strengthened my programming logic and increased my confidence to independently develop functional AI-based systems.

4.2 Future Scope

The current version of *Jarvis* is functional but can be enhanced with the following features:

- GUI Interface: Use Tkinter or PyQt to improve usability.
- Advanced NLP: Integrate transformer-based models like ChatGPT for richer understanding.
- Voice Authentication: Enable secure, personalized interactions.
- IoT Integration: Connect with smart devices for automation.

REFERENCES

[1] M. S. Shehzad, “Project Source Code and Training Materials,” ThinkNext Technologies Pvt. Ltd., Mohali, Punjab, 2025.

[2] J. Bird, “A Guide to Speech Recognition in Python,” Real Python Tutorials,. Available: https://realpython.com/python-speech-recognition/.

[3] TextBlob Developers. “TextBlob Documentation,” TextBlob NLP Toolkit, . Available: https://textblob.readthedocs.io/en/dev/.

[4] SpeechRecognition Contributors. “SpeechRecognition Library for Python,” Python Software Foundation, . Available: https://pypi.org/project/SpeechRecognition/.

[5] pyttsx3 Team. “Text-to-Speech Engine – pyttsx3,” GitHub Repository. Available: https://github.com/nateshmbhat/pyttsx3.

[6] A. M. Jones, “Natural Language Processing in Practice,” International Conference on Artificial Intelligence Systems, Toronto, Canada, 2022, pp. 142–148.

[7] P. Kumar and S. Mehta, “Python-Based AI Assistants and their Architecture,” Journal of Computer Applications and Trends, vol. 45, no. 2, pp. 22–28, Jan. 2023.

APPENDIX

Project Title: ***Jarvis – Python-Based Voice Assistant***

A. Project Summary

The Jarvis assistant is a modular, Python-based AI system capable of performing real-time voice recognition, natural language understanding, sentiment analysis, and response delivery. It also integrates linear regression models to predict student marks and house prices based on user voice input.

B. Key Functional Components

1. Speech Recognition
    - Real-time voice capture using speech_recognition
    - Handles ambient noise and timeout exceptions
2. Text-to-Speech Feedback
    - Voice response using pyttsx3 (offline support)
    - Dynamic speech properties (rate, volume)
3. Sentiment Analysis
    - Sentiment polarity detection using TextBlob
    - Conditional voice response (positive/neutral/negative)

4. Task Automation
    - Opens applications like Notepad, VS Code, YouTube, Google
    - Takes notes and stores them in a text file
5. Prediction Models
    - Uses LinearRegression from scikit-learn
    - Predicts:
        - Student Marks (based on study hours)
        - House Prices (based on house size)

C. Sample Class Structure

```
def run(self):
    """Main execution loop"""
    self.speak("Hello, I am your AI assistant Jarvis. How can I help you today?")
     while True:
      try:
         command = self.listen()
         if not command:
            continue
```

```
            command = self.normalize_command(command)
            # First check for exit command
            if any(word in command for word in ["exit", "quit", "goodbye"]):
                self.speak("Goodbye! Have a great day.")
                sys.exit(0)
            # Check for exact command matches first
            command_found = False
            for cmd in self.command_map:
                if cmd in command:
                    self.command_map[cmd]()
                    command_found = True
                    break
            if not command_found:
                self.speak("Sorry, I didn't understand that command. Try something like 'open YouTube' or 'analyse sentiment'.")
        except KeyboardInterrupt:
            self.speak("Goodbye!")
            sys.exit(0)
```

```
except Exception as e:
    print(f"Error: {e}")
    self.speak("Sorry, I encountered an error. Please try again.")
    time.sleep(1)
    continue
```

D. Tools & Libraries Used

Table 4.1 – Summary of Functional Features in Jarvis Assistant

Library/Tool	**Purpose**
pyttsx3	Offline text-to-speech feedback
speech_recognition	Converts user voice to text
TextBlob	Sentiment analysis (polarity scoring)
NumPy	Data handling and randomness
scikit-learn	Linear regression for predictions
VS Code + Git	Development and version control