**BASAVARAJESWARI GROUP OF INSTITUTIONS**

**Ballari Institute of Technology & Management**

**AUTONOMOUS INSTITUTE UNDER VISVESVARAYA TECHNOLOGICAL UNIVERSITYJNANA SANGAMA, BELAGAVI 590018**

**INTERNSHIP**

**Report On**

# NETWORK LATENCY TESTER

Submitted in partial fulfillment of the requirements for the award of degree of

**Bachelor of Engineering**

**In**

**COMPUTER SCIENCE – DATA SCIENCE**

## Submitted by

SHREYAS

22BI24DS404-T

## Internship Carried Out By

**EZ TRAININGS & TECHNOLOGIES PVT.LTD HYDERABAD**

**Internal Guide External Guide**

**PARVATHI Vishal Kumar**

**Asst.prof , CSE (DS) Technical Trainer**

ANUSHYA

**Asst. prof,CSE (DS)**

### BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT

NACC Accredited Institution\*

**(Recognized by Govt. of Karnataka, approved by AICTE, New Delhi & Affiliated to Visvesvaraya Technological University, Belagavi)**

**"Jnana Gangotri" Campus, No.873/2, Ballari-Hospet Road, Allipur, Ballar1-583 104 (Karnataka) (India) Ph: 08392 – 237100 / 237190, Fax: 08392 – 237197**

**2024-2025**

**BASAVARAJESWARI GROUP OF INSTITUTIONS**

## BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT

**Autonomous institute under VISVESVARAYA TECHNOLOGICAL UNIVERSITYJNANA SANGAMA,**

**BELAGAVI 590018**

NACC Accredited Institution\*

**(Recognized by Govt. of Karnataka, approved by AICTE, New Delhi & Affiliated to**

**Visvesvaraya Technological University, Belagavi)**

### "JnanaGangotri" Campus, No.873/2,Ballari-HospetRoad,Allipur,

**Ballar1-583 104 (Karnataka)(India)**

**Ph: 08392 – 237100 / 237190, Fax: 08392 –237197**

**DEPARTMENT OF COMPUTER SCIENCE – DATA SCIENCE**

# CERTIFICATE

This is to certify that the Internship entitled **“NETWORK LATENCY TESTER”** has been successfully completed by **SHREYAS** bearing USN **22BI24DS404-T** a bonafide student of Ballari Institute of Technology and Management, Ballari. For the partial fulfillment of the requirementsfor the **Bachelor’s Degree in Computer Science and Engineering (DS)** of the VISVESVARAYA TECHNOLOGICAL UNIVERSITY,

Belagavi during the academic year 2024-2025.

**Signature of Internship Signature of HOD**

**Co-ordinator**

**PARVATHI Dr. ARADHANA**

**Asst. prof, CSE (DS) Prof. and HOD CSE(DS)**

**ANUSHYA**

**Asst. prof, CSE (DS)**

**DECLARATION**

I, SHREYAS second year student of Computer Science and Engineering (DS) Ballari Institute of Technology, Ballari, declare that Internship entitled **NETWORK LATENCY TESTER** is a part of Internship Training successfully carried out by **EZ TECHNOLOGIES & TRAININGS PVT. LTD, Hyderabad** at “**BITM, BALLARI”.** This report is submitted in partial fulfillment of the requirements for the award of the degree, Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belagavi.

**Date : 28 Sep 2024 Signature of the Student**

**Place : Bellary**

**ACKNOWLEDGEMENT**

The satisfactions that a company the successful completion of my internship on “NETWORK LATENCY TESTER” would be incomplete without the mention of people who made it possible, whose noble gesture, affection, guidance, encouragement and support crowned my efforts with success. It is my privilege to express my gratitude and respect to all those who inspired me in the completion of my internship.

I am grateful to our respective coordinator **“PARVATHI (Asst.prof,CSE(DS)), ANUSHYA**

**(Asst. prof, CSE (DS))”** for his noble gesture, support co-ordination and valuable suggestions given to me in the completion of Internship.

I also thank **ARADHANA,** H.O.D. Department of **Computer science – Data Science Engineering** for extending all his valuable support and encouragement.

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Chapter Name** | **Page No.** |
| **1** | **College Profile** | **01** |
| **2** | **Certificate** | **02** |
| **3** | **Declaration** | **03** |
| **4** | **Acknowledgement** | **04** |
| **5** | **Internship Report** | **06** |
| **6** | **Introduction** | **07** |
| **7** | **Code Implementation** | **09-10** |
| **8** | **Output** | **10-11** |
| **9** | **Conclusion** | **11** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Day** | **Date** | **Content Covered** | **Signature of the** |
| **faculty in-charge** |
| **1** | **09.09.24** | **Introduction to Python, Setup & Installation, First Python Program, Variables, Data Types,**  **`and Basic I/O** |  |
| **2** | **10.09.24** | **Control Structures: If-else, Loops, Functions and Modules** |  |
| **3** | **11.09.24** | **Lists, Tuples, and Dictionaries, File Handling** |  |
| **4** | **12.09.24** | **Exception Handling, Practice exercises on Python basics** |  |
| **5** | **13.09.24** | **Introduction to OOP, Classes, and Objects** |  |
| **6** | **14.09.24** | **Inheritance, Polymorphism, and Encapsulation** |  |
| **7** | **15.09.24** | **Abstract Classes and Interfaces** |  |
| **8** | **17.09.24** | **Practice exercises on OOP concepts** |  |
| **9** | **18.09.24** | **Introduction to DSA, Arrays, and Linked Lists** |  |
| **10** | **19.09.24** | **Introduction to Stacks and Queues** |  |
| **11** | **20.09.24** | **Practice exercise on basic concept(Reduce, Lambda function, List comprehension)** |  |
| **12** | **21.09.24** | **Introduction to Tree Data Structure** |  |
| **13** | **23.09.24** | **Introduction to Graph Data Structure** |  |
| **14** | **24.09.24** | **Searching Algorithm project building and presentation** |  |
| **15** | **25.09.24** | **Project Building & Presentations** |  |
| **16** | **26.09.24** | **Project Building & Presentations** |  |
| **17** | **27.09.24** | **Project Building & Presentations** |  |
| **18** | **28.09.24** | **Project Building & Presentations** |  |

**INTRODUCTION**

The Network Latency Tester is a simple tool designed to measure and simulate the time delay, or latency, in network connections. By allowing users to test and analyze how quickly data travels from one point to another, it helps identify slowdowns in the network. With features to add, update, and track latency records, the tool is useful for improving network performance, making it ideal for both beginners and professionals. It helps ensure smoother, faster communication for online activities like video calls, gaming, and cloud-based applications.

1. **Problem Statement**

High network latency causes delays, slow data transfer, and poor user experiences. This affects businesses, gamers, and remote workers who rely on fast connections. There is a need for a simple tool to measure and monitor latency, helping users quickly identify and fix network issues for better performance.

1. **Objectives**
2. Measure Latency: Provide accurate and real-time measurements of network latency.
3. Monitor Performance: Track latency over time to detect patterns and identify issues.
4. Simplify Troubleshooting: Help users easily find and resolve network delays.
5. User-Friendly Interface: Offer a simple, intuitive tool for non-technical and technical users alike.
6. **Technologies and Tools Used**
7. Python: Core programming language for building the application logic.
8. Colorama: For adding color to terminal outputs, enhancing user interaction.
9. unittest: Python's built-in testing framework to ensure functionality and reliability.
10. Random & Time Modules: To simulate latency and introduce realistic delay in tests.

**CODE IMPLEMENTATION:**

import time

import random

import unittest

from datetime import datetime

from colorama import Fore, Style, init

# Initialize colorama

init(autoreset=True)

class LatencyTestRecord:

    def \_\_init\_\_(self, source, latency, test\_time=None):

        """

        Initializes a latency test record with provided details.

        :param source: The source location for the latency test.

        :param latency: The measured latency for the source.

        :param test\_time: Optional time the test was performed.

        """

        self.source = source

        self.latency = latency

        self.test\_time = test\_time if test\_time else datetime.now()

    def \_\_str\_\_(self):

        return f"{Fore.CYAN}Latency for {Fore.YELLOW}{self.source}{Fore.CYAN}: {Fore.GREEN}{self.latency}ms{Fore.CYAN}, tested at {Fore.MAGENTA}{self.test\_time}{Style.RESET\_ALL}"

class DataStorage:

    """ Class for handling storage of latency test records. """

    def \_\_init\_\_(self):

        self.records = []

    def add\_record(self, record):

        """Adds a latency record to the storage."""

        self.records.append(record)

    def get\_record(self, idx):

        """Retrieves a latency record by its index."""

        return self.records[idx] if 0 <= idx < len(self.records) else f"{Fore.RED}Record not found{Style.RESET\_ALL}"

    def modify\_record(self, idx, new\_record):

        """Modifies a specific latency record at the given index."""

        if 0 <= idx < len(self.records):

            self.records[idx] = new\_record

            return True

        return False

    def remove\_record(self, idx):

        """Removes a latency record at the given index."""

        if 0 <= idx < len(self.records):

            self.records.pop(idx)

            return True

        return False

    def display\_all(self):

        """Displays all latency records."""

        if not self.records:

            print(f"{Fore.RED}No records available.{Style.RESET\_ALL}")

        else:

            print(f"{Fore.GREEN}Displaying all records:{Style.RESET\_ALL}")

            for idx, record in enumerate(self.records):

                print(f"{Fore.CYAN}[{idx}] {record}")

def run\_latency\_simulation(source):

    """

    Simulates latency for a given source location.

    :param source: The source location for the latency test.

    :return: Simulated latency in milliseconds.

    """

    latency = random.randint(1, 100)  # Random simulated latency in ms

    print(f"{Fore.YELLOW}Simulating latency for {source}...{Style.RESET\_ALL}")

    time.sleep(1)  # Simulate a delay

    print(f"{Fore.GREEN}Simulated latency: {latency} ms for {source}{Style.RESET\_ALL}")

    return LatencyTestRecord(source, latency)

class LatencyTestCase(unittest.TestCase):

    def setUp(self):

        self.storage = DataStorage()

    def test\_add\_and\_get\_record(self):

        test\_record = run\_latency\_simulation("New York")

        self.storage.add\_record(test\_record)

        self.assertEqual(self.storage.get\_record(0), test\_record, "Error in adding and retrieving record")

    def test\_modify\_record(self):

        self.storage.add\_record(run\_latency\_simulation("New York"))

        modified\_record = LatencyTestRecord("New York", 75)

        self.assertTrue(self.storage.modify\_record(0, modified\_record))

        self.assertEqual(self.storage.get\_record(0), modified\_record, "Error in modifying the record")

    def test\_remove\_record(self):

        self.storage.add\_record(run\_latency\_simulation("New York"))

        self.assertTrue(self.storage.remove\_record(0))

        self.assertEqual(self.storage.get\_record(0), "Record not found", "Error in deleting the record")

def print\_stylish\_header():

    """Prints a stylish header with ASCII art."""

    print(f"{Fore.CYAN}========================================")

    print(f"        {Fore.YELLOW}⚡ Network Latency Tester ⚡")

    print(f"{Fore.CYAN}========================================{Style.RESET\_ALL}")

def main\_menu():

    """Displays the main menu and handles user interaction."""

    storage = DataStorage()

    while True:

        print\_stylish\_header()

        print(f"{Fore.GREEN}1. {Fore.CYAN}Add new latency record")

        print(f"{Fore.GREEN}2. {Fore.CYAN}Measure latency")

        print(f"{Fore.GREEN}3. {Fore.CYAN}Update latency record")

        print(f"{Fore.GREEN}4. {Fore.CYAN}Delete latency record")

        print(f"{Fore.GREEN}5. {Fore.CYAN}Show all records")

        print(f"{Fore.GREEN}6. {Fore.CYAN}Quit")

        print(f"{Fore.CYAN}========================================{Style.RESET\_ALL}")

        choice = input(f"{Fore.YELLOW}Choose an option (1-6): {Style.RESET\_ALL}")

        if choice == '1':

            source = input(f"{Fore.CYAN}Enter the source location: {Style.RESET\_ALL}")

            test\_record = run\_latency\_simulation(source)

            storage.add\_record(test\_record)

            print(f"\n{Fore.GREEN}Record added: {test\_record}{Style.RESET\_ALL}")

        elif choice == '2':

            source = input(f"{Fore.CYAN}Enter the source location: {Style.RESET\_ALL}")

            test\_record = run\_latency\_simulation(source)

            print(f"\n{Fore.GREEN}Latency measured: {test\_record}{Style.RESET\_ALL}")

        elif choice == '3':

            if not storage.records:

                print(f"{Fore.RED}No records to update. Add a record first.{Style.RESET\_ALL}")

                continue

            try:

                idx = int(input(f"{Fore.CYAN}Enter the index of the record to update: {Style.RESET\_ALL}"))

                source = input(f"{Fore.CYAN}Enter the source location: {Style.RESET\_ALL}")

                latency = int(input(f"{Fore.CYAN}Enter the new latency (ms): {Style.RESET\_ALL}"))

                new\_record = LatencyTestRecord(source, latency)

                if storage.modify\_record(idx, new\_record):

                    print(f"{Fore.GREEN}Record at index {idx} updated successfully.{Style.RESET\_ALL}")

                else:

                    print(f"{Fore.RED}Error: Invalid index.{Style.RESET\_ALL}")

            except ValueError:

                print(f"{Fore.RED}Error: Invalid input. Please enter numeric values for index and latency.{Style.RESET\_ALL}")

        elif choice == '4':

            if not storage.records:

                print(f"{Fore.RED}No records to delete. Add a record first.{Style.RESET\_ALL}")

                continue

            try:

                idx = int(input(f"{Fore.CYAN}Enter the index of the record to delete: {Style.RESET\_ALL}"))

                if storage.remove\_record(idx):

                    print(f"{Fore.GREEN}Record at index {idx} deleted successfully.{Style.RESET\_ALL}")

                else:

                    print(f"{Fore.RED}Error: Invalid index.{Style.RESET\_ALL}")

            except ValueError:

                print(f"{Fore.RED}Error: Invalid input. Please enter a numeric index.{Style.RESET\_ALL}")

        elif choice == '5':

            print(f"\n{Fore.CYAN}Displaying all records:{Style.RESET\_ALL}")

            storage.display\_all()

        elif choice == '6':

            print(f"{Fore.CYAN}Exiting Network Latency Tester. Goodbye!{Style.RESET\_ALL}")

            break

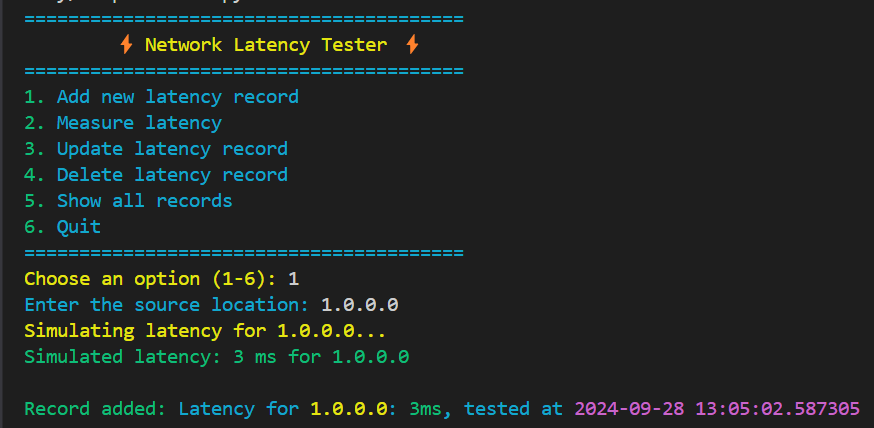
        else:

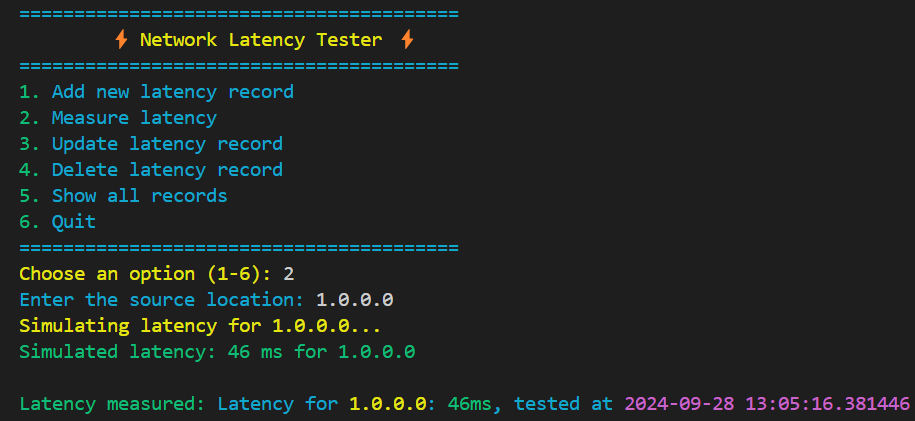
            print(f"{Fore.RED}Invalid choice. Please choose an option between 1 and 6.{Style.RESET\_ALL}")

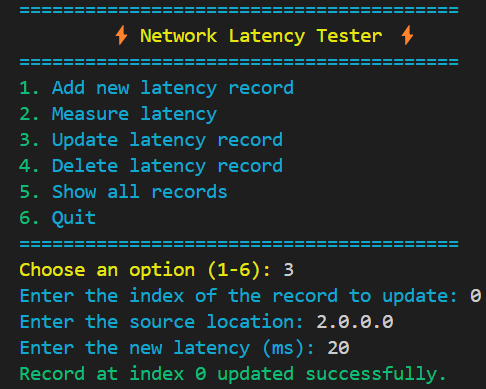
if \_\_name\_\_ == '\_\_main\_\_':

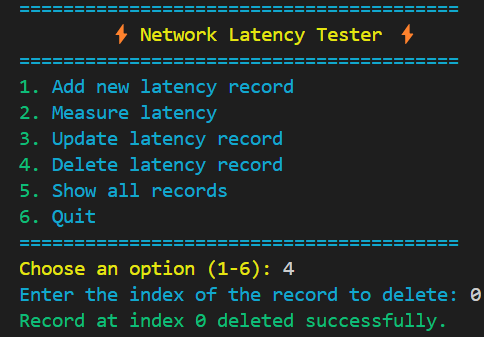
    main\_menu()

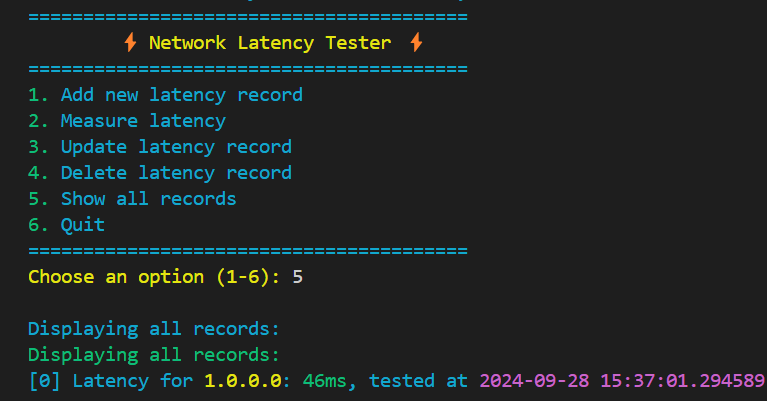
Output:

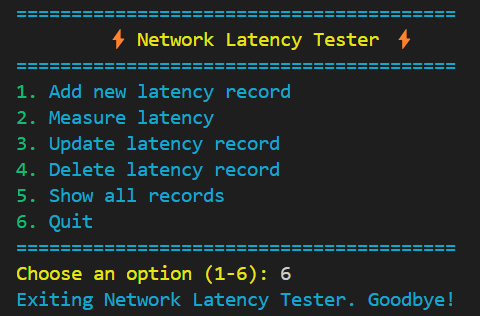












**CONCLUSION :**

The Network Latency Tester provides a practical and user-friendly solution for measuring, recording, and managing network latency data. With its intuitive interface and enhanced visual feedback, users can easily simulate and monitor latency performance across various locations. This tool helps identify network inefficiencies and ensures smoother network operations by enabling timely insights into latency issues.

1. **Practical Tool**: The Network Latency Tester serves as a valuable resource for assessing network performance, making it essential for network administrators and users alike.
2. **User-Friendly Interface**: The intuitive design allows for easy navigation and interaction, enabling users to efficiently conduct latency tests without technical expertise.
3. **Real-Time Insights**: The tool provides immediate feedback on latency measurements, helping to identify network inefficiencies and optimize performance.
4. **Data Management**: Users can conveniently record, update, and manage latency records, enhancing their ability to track and analyze network performance over time.
5. **Improved Network Efficiency**: By facilitating timely insights into latency issues, the tool contributes to smoother network operations and improved overall performance.

In summary, the Network Latency Tester is a valuable tool for measuring network performance and identifying latency issues. Its user-friendly interface allows for easy recording and updating of latency data, facilitating efficient network management. By optimizing latency, this tool helps enhance overall network reliability and performance.

**REFERENCE**:

<https://chatgpt.com/>

<https://www.google.co.in/>