BASAVARAJESWARI GROUP OF INSTITUTIONS

Ballari Institute of Technology & Management

AUTONOMOUS INSTITUTE UNDER VISVESVARAYA TECHNOLOGICAL UNIVERSITYJNANA SANGAMA, BELAGAVI 590018

INTERNSHIP

Report On

BASIC CONFIGURATION MANAGEMENT TOOL

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

Bachelor of Engineering

In

COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

Submitted by

RIDA ARSHAD

3BR23CD076

Internship Carried Out By

EZ TRAININGS & TECHNOLOGIES PVT.LTD HYDERABAD

Internal Guide

V P ANUSHYA

Asst.prof,CSE-DS

KAMALAPADU VARSHA

Teaching Asst.CSE-DS

External Guide

VISHAL KUMAR

Technical Trainer

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 AND ENGINEERING

CERTIFICATE

This is to certify that the Internship entitled **" BASIC CONFIGURATION MANAGEMENT TOOL"** has been successfully completed by **RIDA ARSHAD** bearing USN **3BR23CD076** 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-Data science** of the VISVESVARAYA TECHNOLOGICAL UNIVERSITY, Belagavi during the academic year 2024-2025.

Signature of Internship Co-ordinator

V P ANUSHYA

Asst.prof,CSE-DS

KAMALAPADU VARSHA

Asst. prof,CSE-DS

Signature of HOD

DR. ARADHANA

Prof. and HOD(CSE-DS)

DECLARATION

I, **RIDA ARSHAD,** second year student of Computer Science and Engineering(Data Science), Ballari Institute of Technology, Ballari, declare that Internship entitled **BASIC CONFIGURATION MANAGEMENT TOOL** 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 (Data Science) of the Visvesvaraya Technological University, Belagavi.

Date : 28/09/2024 **Signature of the Student**
Place : BALLARI

ACKNOWLEDGEMENT

The satisfactions that a company the successful completion of my internship on " Basic Configuration Management Tool " 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 **"V P Anushya (Asst.prof,CSE-DS) , Kamalapadu Varsha (Teaching.Asst.CSE-DS)"** for his noble gesture, support co-ordination and valuable suggestions given to me in the completion of Internship.

I also thank **DR. AARADHANA,** H.O.D. Department of **Computer science and engineering-Data Science** for extending all her valuable support and encouragement.

Table of Contents

CHAPTER-1

DAY TO DAY ACTIVITIES

Basavarajeshwari Group of Institutions
BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT
Autonomous Institute under VTU - Belagavi
"Jnana Gangotri" Campus, Bellary-Hospet Road, Near Allipura Village,
BALLARI - 583 104 (Karnataka)
Ph: 08392-237167/237153 Fax: 237197, e-mail: bitmbly@gmail.com
Website: www.bitm.edu.in

Internship Program on Python for BE-3rd Sem students
From 9th September to 28th September (During 3rd semester vacations).

Student Name: Rida Arshad USN No: 3BR23CD076 Branch: CSE-DS

INDEX PAGE			
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 DSA, Arrays, and Linked Lists	
11	20.09.24	Introduction to Stacks and Queues	
12	21.09.24	Practice Exercise on Basic Concepts (Reduce, Lambda Function, List Comprehensions)	
13	23.09.24	Introduction To Tree Data Structure	

14	**24.09.24**	**Introduction To Graph Data Structures**	
15	**25.09.24**	**Searching Algorithms & Project Building & Presentations**	
16	**26.09.24**	**Project Building & Presentations**	
17	**27.09.24**	**Project Building & Presentations**	
18	**28.09.24**	**Project Building & Presentations**	

CHAPTER-2

COMPANY PROFILE

Company Name: EZ Trainings and Technologies Pvt. Ltd.

Introduction:

EZ Trainings and Technologies Pvt. Ltd. is a dynamic and innovative organization dedicated to providing comprehensive training solutions and expert development services. Established with a vision to bridge the gap between academic learning and industry requirements, we specialize in college trainings for students, focusing on preparing them for successful placements. Additionally, we excel in undertaking development projects, leveraging cutting-edge technologies to bring ideas to life.

Mission:

Our mission is to empower the next generation of professionals by imparting relevant skills and knowledge through specialized training programs. We strive to be a catalyst in the career growth of students and contribute to the technological advancement of businesses through our development projects.

Services:

College Trainings:

- Tailored training programs designed to enhance the employability of students.
- Industry-aligned curriculum covering technical and soft skills.
- Placement assistance and career guidance.

Development Projects:

- End-to-end development services, from ideation to execution.
- Expertise in diverse technologies and frameworks.
- Custom solutions to meet specific business needs.

Locations: Hyderabad | Delhi NCR

At EZ Trainings and Technologies Pvt. Ltd., we believe in transforming potential into excellence

CHAPTER-3

ABSTRACT

This project presents the development of a Basic Configuration Management Tool, designed to manage and track configuration changes in a simulated environment using Object-Oriented Programming (OOP) principles in Python. The primary objective of the tool is to facilitate efficient management of configuration files while maintaining an audit trail of modifications, without the complexity of interacting with real servers or databases. Instead, operations are simulated using in-memory data structures such as dictionaries and lists.

The system is built around four core components:

1. ConfigurationManager: This class is responsible for handling CRUD (Create, Read, Update, Delete) operations on configuration files. It enables users to manage configurations in an organized and systematic manner, providing the basic functionality required for efficient configuration management.

2. Configuration: Each configuration file is represented by an instance of this class. The Configuration class stores the configuration data and metadata, encapsulating the structure of a single configuration file. This allows for easy manipulation and retrieval of configuration details when needed.

3. ChangeLogger: This component plays a crucial role in tracking and maintaining logs of all configuration changes. Every modification—whether a file is created, updated, or deleted—is recorded with relevant details (e.g., timestamps, user actions) to ensure accountability and transparency in the management process.

4. ServerSimulator: To simulate real-world scenarios, the tool includes a ServerSimulator class, which mimics the process of deploying configuration changes to servers. While no actual server interaction occurs, this component provides a realistic simulation of the deployment process, demonstrating how the tool would function in a live environment.

Together, these components form a cohesive system that allows users to manage configuration files, deploy changes, and track modifications in a controlled, simulated environment. By utilizing OOP design patterns, the tool remains flexible, maintainable, and easily extensible for future enhancements or integration with actual infrastructure systems. The project provides a foundational solution for configuration management, demonstrating core principles and functionality that can be expanded into more complex systems if needed.

CHAPTER-4

INTRODUCTION OF THE PROJECT

- The Basic Configuration Management Tool is a Python-based application designed to streamline the management, modification, and tracking of configuration files, which are essential for the proper functioning of servers, applications, and network systems. Configuration management is a critical task for IT administrators, DevOps teams, and system engineers, ensuring that system settings are maintained consistently across servers and that any changes made to these configurations are logged for future reference.

- In real-world scenarios, configuration files control a wide array of parameters, such as network settings, security policies, database connections, and application-specific settings. Proper management of these configurations is crucial for ensuring system stability, security, and operational efficiency. Without an effective system for managing and tracking changes, errors may occur, and troubleshooting becomes more complex and time-consuming.
- The goal of this project is to create a basic yet functional tool that allows users to perform essential operations on configuration files.

- The tool is implemented using Object-Oriented Programming (OOP) principles in Python, ensuring the code is modular, maintainable, and extendable. It also incorporates a role-based access control system, where two distinct roles User and Admin—are defined. These roles come with specific permissions to ensure that only authorized individuals can make critical changes to configurations.
    - *Users:* Have limited privileges, including the ability to create and read configuration entries. This role is suitable for team members who only need to view or suggest configurations but should not be able to alter or delete critical settings.
    - *Admins:* Have full control over the configuration management system. In addition to the capabilities of a user, admins can update, delete configurations, view all stored configurations, and access a detailed history log of all changes made. Admin access is protected by a password to prevent unauthorized modifications.

- The configuration management tool also logs every operation performed, whether it's creating a new configuration, reading an existing one, or updating/deleting a configuration entry. This change log or audit trail ensures that any changes to the system can be traced back to the user who performed the action, making it easier to troubleshoot issues and providing a layer of accountability for all configuration changes.

- While the project simulates configuration management using in-memory data structures like dictionaries (to store configuration settings) and lists (to track history), it demonstrates how such tools function in a real-world IT environment. With future enhancements, the tool could be integrated into live systems, interacting with actual servers and databases to manage configurations in production environments

- Overall, the Basic Configuration Management Tool offers a simple yet powerful solution to manage, deploy, and audit system configurations, demonstrating core principles of configuration management and laying the groundwork for future scalability and integration with real-world infrastructure.

CHAPTER-5

MODULE DESCRIPTION

The Basic Configuration Management Tool is composed of several modules, each responsible for different functionalities within the system. The primary modules include User, Admin, and History & Logging, all working together to facilitate configuration management and tracking.

***User Module*:**

The User class allows basic interactions with the configuration system, enabling users to perform essential operations without administrative privileges. This role is suited for users who need to create or read configuration entries but do not have permission to modify or delete them.

Key Features:

1. Create Configuration:

- Users can add new configuration entries by specifying key-value pairs. For example, adding settings for network or security policies.

- Each operation is logged with a timestamp, allowing the system to track when a configuration was created.

2. Read Configuration:

- Users can view existing configurations by querying keys. If the key exists, the corresponding value is returned; otherwise, an error message is shown.

- All read operations are recorded in the system history for future reference.

Role Limitation:

- Users cannot modify or delete configurations, nor can they access the change history. This ensures only authorized personnel can perform sensitive operations.

Admin Module:

The Admin class extends the User class, granting additional privileges, such as the ability to update, delete, and view the entire configuration history. Admin access is password-protected to ensure security.

Key Features:

1. Update Configuration:

- Admins can modify existing configurations by updating values associated with keys. This is useful for adjusting settings such as database connections or server policies.

- Update operations are logged with timestamps to track changes.

2. Delete Configuration:

- Admins can remove outdated or irrelevant configurations. This ensures that only necessary data is maintained in the system.

- All delete operations are logged, maintaining a clear record of changes.

3. View History:

- Admins can access the history log to review all create, update, and delete actions, helping them track changes and diagnose issues.

4. View All Configurations:

- Admins can see the full list of configurations at any time, providing a complete view of the system's current state.

Authentication:

- Admin access is protected by a password. Only admins who authenticate successfully can access advanced features, ensuring secure control over sensitive operations.

History and Logging Module:

The logging system ensures that all operations—whether by users or admins—are recorded for future reference. This helps maintain transparency and provides an audit trail of changes.

Key Features:

1. Action Logging:

- Every operation, including creating, reading, updating, and deleting configurations, is logged with a timestamp. The system also records which user performed the action.

2. Accountability:

- The logs help establish accountability, ensuring that any changes made to the system can be traced back to the user or admin who performed the operation.

3. History Review:

- Admins can view the entire action history at any time, allowing them to monitor system activity and troubleshoot issues as needed.

Simulated Configuration Deployment (Future Scope):

While not part of the current implementation, a future module could simulate the deployment of configuration changes to servers. This would allow admins to push configuration updates to multiple servers simultaneously, reducing the chance of human error and ensuring consistency across systems.

In summary, the Basic Configuration Management Tool offers a modular and structured approach to managing configurations, providing essential CRUD operations, role-based access control, and comprehensive logging for accountability. The modular design ensures that future extensions, such as server interaction and deployment automation, can be integrated seamlessly into the system.

CHAPTER-6

ALGORITHM

1. **Start**

2. Initialize Variables

- Initialize configurations as an empty dictionary to store key-value pairs.
- Initialize history as an empty list to log all actions performed.

3. Define Classes

- **Class User**
    - **Method create_config()**
        1. Prompt user for the number of attributes.
        2. For each attribute:
            - Prompt for key and value.
            - Add the key-value pair to configurations.
        3. Log the creation action with a timestamp in history.
    - **Method read_config()**
        1. Continuously prompt for a key to read:
            - If key is found, display its value.
            - If not found, display an error message.
        2. Log the read operation with a timestamp.
- **Class Admin (inherits from User)**
    - **Method update_config()**
        1. Continuously prompt for a key to update:
            - If key exists, prompt for a new value and update it.
            - If not found, display an error message.
        2. Log the update action with a timestamp.
    - **Method delete_config()**
        1. Continuously prompt for a key to delete:
            - If key exists, remove it from configurations.
            - If not found, display an error message.
        2. Log the delete action with a timestamp.
    - **Method view_history()**
        1. Display all entries in history.
    - **Method view_config()**
        1. Display all entries in configurations.

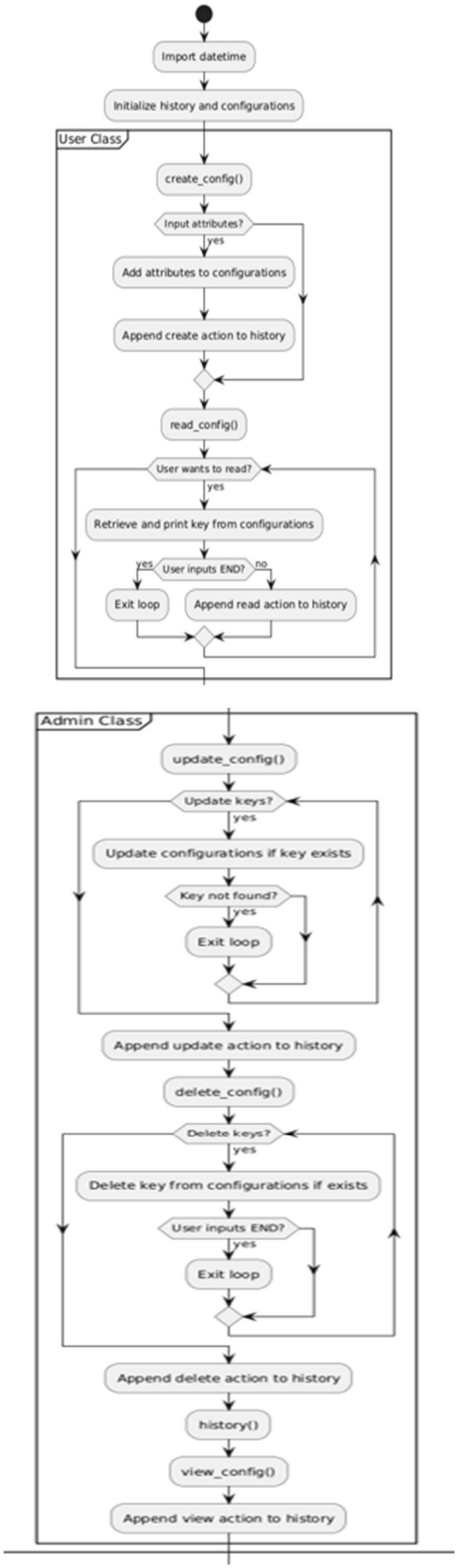
4. Main Function

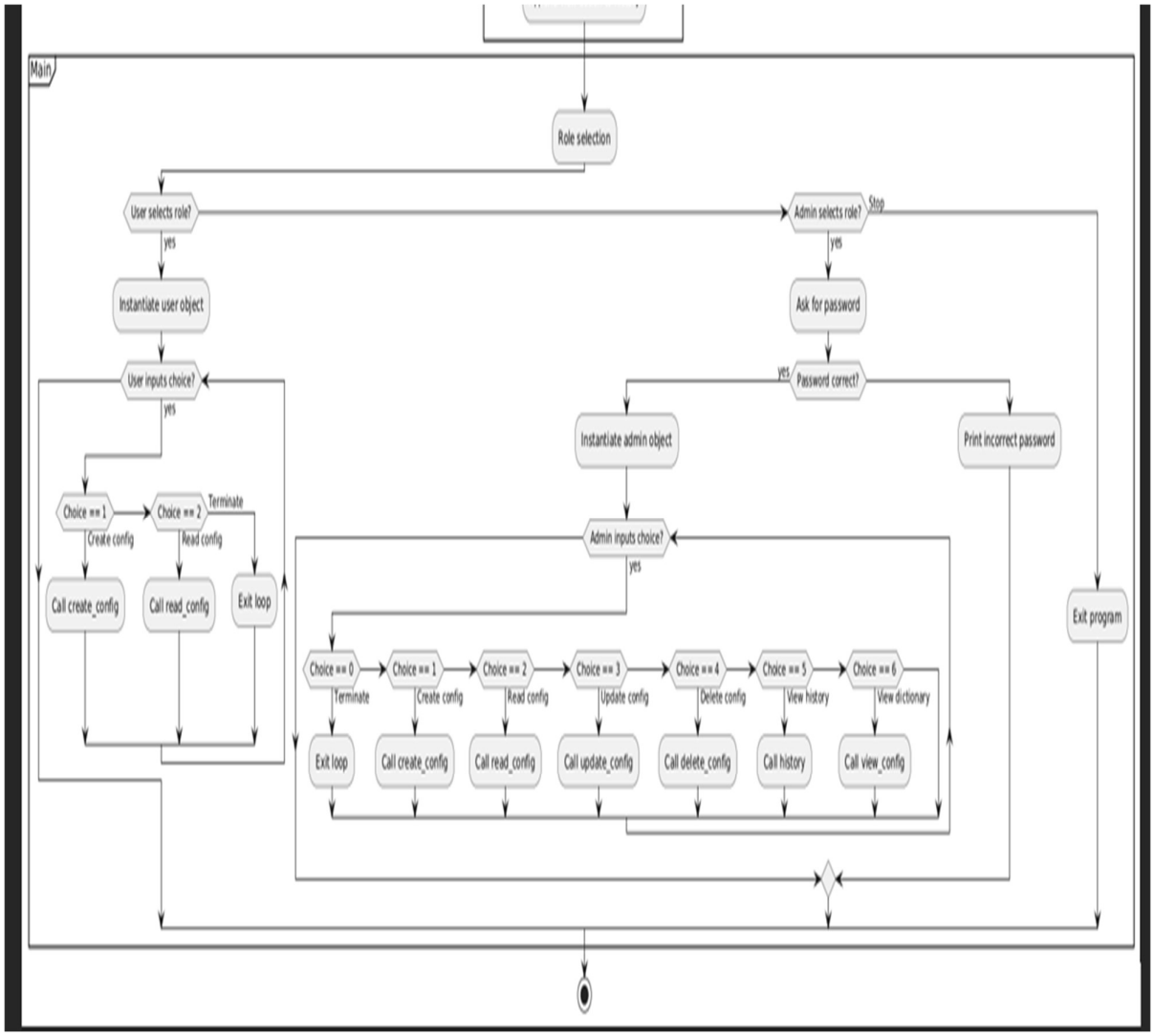
1. Loop indefinitely to handle user input:
    - Prompt for role (user or admin):
        - If role is "user":
            1. Create a User object.
            2. Present menu for operations (CREATE, READ):
                - If CREATE, call create_config().
                - If READ, call read_config().
                - If exit, break the loop.
        - If role is "admin":
            1. Prompt for password:
                - If correct, create an Admin object.
                2. Present menu for admin operations (CREATE, READ, UPDATE, DELETE, VIEW HISTORY, VIEW CONFIG):
                    - If CREATE, call create_config().
                    - If READ, call read_config().
                    - If UPDATE, call update_config().
                    - If DELETE, call delete_config().
                    - If VIEW HISTORY, call view_history().
                    - If VIEW CONFIG, call view_config().
                    - If exit, break the loop.
        - If role is invalid, display an error message.

5. End

CHAPTER-7

FLOWCHART




Main
Role selection
User selects role?
Admin selects role?
Stop
yes
yes
Instantiate user object
Ask for password
User inputs choice?
yes
Password correct?
yes
Instantiate admin object
Print incorrect password
Choice == 1
Choice == 2
Terminate
Create config
Read config
Call create_config
Call read_config
Exit loop
Admin inputs choice?
yes
Exit program
Choice == 0
Choice == 1
Choice == 2
Choice == 3
Choice == 4
Choice == 5
Choice == 6
Terminate
Create config
Read config
Update config
Delete config
View history
View dictionary
Exit loop
Call create_config
Call read_config
Call update_config
Call delete_config
Call history
Call view_config

CHAPTER-8

SOURCE CODE

```
from datetime import datetime

history = []
configurations = {}

class user:

    def create_config(self):
        no_of_attributes = int(input('\nEnter the number of attributes you want\t'))
        for i in range(no_of_attributes):
            key = input('\nEnter the key\t')
            value = input('\nEnter the value\t')
            configurations[key] = value

        print('\nYour current list of dictionary is:')
        currenttime = datetime.now()
        print(configurations)
        history.append(f'Attribute has been created! (Created at {currenttime})')

    def read_config(self):
        while(True):
            print('\nEnter the key you want to read. (Enter END to terminate the operation)\t')
            user_key = input()
            attribute = configurations.get(user_key, 'The input key was not found')
            print(attribute)
            if(user_key == 'END'):
                print('\nLoop has been exited\t')
                break
            currenttime = datetime.now()

        history.append(f'Read operation has been performed!(Read at {currenttime})')

class admin(user):

    def update_config(self):
        while(True):
            print('\nEnter the key to be updated (Type END to kill the process)\t')
            user_key = input()
            if(user_key == 'END'):
                break
            if user_key in configurations:
                print('\nEnter the value to be updated at key\t')
                user_value = input()
                configurations[user_key] = user_value
            else:
                print('\nThe entered key was not found!\t')
                break
```

```
        currenttime = datetime.now()

        print('\nYour updated values are: ')
        print(configurations)
        history.append(f'The dictionary has been updated! (Updated at: {currenttime})')

    def delete_config(self):
        while(True):
            print('\nEnter the key to be deleted from the values (Enter END to terminate the loop)\t')
            user_key = input()
            if(user_key == 'END'):
                break
            if user_key in configurations:
                del configurations[user_key]
                print('\nThe given key is successfully deleted. Your current values are:\t')
                print(configurations)
            else:
                print('\nThe entered key was not found\t')
        currenttime = datetime.now()

        history.append(f'Delete operation has been performed. (Deleted at {currenttime})')

    def history(self):
        print(history)

    def view_config(self):
        print(configurations)
        currenttime = datetime.now()

        history.append(f'The dictionary has been viewed. (Viewed at {currenttime})')


def main():
    while(True):
        print('Enter your role (enter STOP to terminate the loop!): ', end = " ")
        role = input()
        role = role.lower()
        if(role == 'user'):
            userobj = user()
            while(True):
                print('\n ')
                print('''Enter your choice!
                Press 1 for CREATE
                Press 2 to READ
                Press 0 to Terminate loop.''')
                choice = int(input())
                if(choice == 1):
                    userobj.create_config()
                elif(choice == 2):
                    userobj.read_config()
                elif(choice==0):
```

```
            break
        else:
            print('Enter valid input')

    elif(role=='admin'):
        print('enter the password.')
        password=input()
        if(password=='admin123'):
            adminobj=admin()
            while(True):
                print('\n')
                print('''Enter your choice!
                Press 1 for CREATE
                Press 2 to READ
                Press 3 to UPDATE
                Press 4 to DELETE
                Press 5 to view history
                Press 6 to view the dictionary
                Press 0 to terminate the loop.''')

                choice=int(input('\nChoose an option from the menu\t'))
                if(choice==0):
                    break
                elif(choice==1):
                    adminobj.create_config()
                elif(choice==2):
                    adminobj.read_config()
                elif(choice==3):
                    adminobj.update_config()
                elif(choice==4):
                    adminobj.delete_config()
                elif(choice==5):
                    adminobj.history()
                elif(choice==6):
                    adminobj.view_config()
                else:
                    print('\nInvalid choice! Please enter a valid input.\t')

        else:
            print('Incorrect password.')

    elif(role=='stop'):
            break
    else:
        print('Enter the valid role!')

main()
```

CHAPTER-9

OUTPUT

```
Enter your role (enter STOP to terminate the loop!):  invalid_input
Enter the valid role!
Enter your role (enter STOP to terminate the loop!):  user
Enter your choice!
        Press 1 for CREATE
        Press 2 to READ
        Press 0 to Terminate loop.
1
Enter the number of attributes you want         1
Enter the key     firewall_status
Enter the value  Active
Your current list of dictionary is:
{'firewall_status': 'Active'}

Enter your choice!
        Press 1 for CREATE
        Press 2 to READ
        Press 0 to Terminate loop.
2
Enter the key you want to read. (Enter END to terminate the operation)
firewall_status
Active
Enter the key you want to read. (Enter END to terminate the operation)
END
The input key was not found
Loop has been exited

Enter your choice!
        Press 1 for CREATE
```

Press 2 to READ

Press 0 to Terminate loop.

0

Enter your role (enter STOP to terminate the loop!): admin

enter the password.

wrong_password

Incorrect password.

Enter your role (enter STOP to terminate the loop!): admin123

Enter the valid role!

Enter your role (enter STOP to terminate the loop!): admin

enter the password.

admin123

Enter your choice!

Press 1 for CREATE

Press 2 to READ

Press 3 to UPDATE

Press 4 to DELETE

Press 5 to view history

Press 6 to view the dictionary

Press 0 to terminate the loop.

Choose an option from the menu 3

Enter the key to be updated (Type END to kill the process)

firewall_status

Enter the value to be updated at key

Inactive

Enter the key to be updated (Type END to kill the process)

END

Your updated values are:

{'firewall_status': 'Inactive'}

Enter your choice!

Press 1 for CREATE

Press 2 to READ

Press 3 to UPDATE

Press 4 to DELETE

Press 5 to view history

Press 6 to view the dictionary

Press 0 to terminate the loop.

Choose an option from the menu 6

{'firewall_status': 'Inactive'}

Enter your choice!

Press 1 for CREATE

Press 2 to READ

Press 3 to UPDATE

Press 4 to DELETE

Press 5 to view history

Press 6 to view the dictionary

Press 0 to terminate the loop.

Choose an option from the menu 4

Enter the key to be deleted from the values (Enter END to terminate the loop)

firewall_status

The given key is successfully deleted. Your current values are:

{}

Enter the key to be deleted from the values (Enter END to terminate the loop)

5

The entered key was not found

Enter the key to be deleted from the values (Enter END to terminate the loop)

END

Enter your choice!

Press 1 for CREATE

Press 2 to READ

Press 3 to UPDATE

Press 4 to DELETE

Press 5 to view history

Press 6 to view the dictionary

Press 0 to terminate the loop.

Choose an option from the menu 5

['Attribute has been created! (Created at 2024-09-27 19:07:14.817238)', 'Read operation has been performed!(Read at 2024-09-27 19:07:28.206486)', 'The dictionary has been updated! (Updated at: 2024-09-27 19:08:58.368146)', 'The dictionary has been viewed. (Viewed at 2024-09-27 19:09:09.056742)', 'Delete operation has been performed. (Deleted at 2024-09-27 19:09:39.637737)']

Enter your choice!

Press 1 for CREATE

Press 2 to READ

Press 3 to UPDATE

Press 4 to DELETE

Press 5 to view history

Press 6 to view the dictionary

Press 0 to terminate the loop.

Choose an option from the menu 0

Enter your role (enter STOP to terminate the loop!): stop

CHAPTER-10

CONCLUSION

The Basic Configuration Management Tool represents a significant step forward in streamlining the management of configuration files within IT and DevOps environments. By implementing a user-friendly interface and a modular design, this tool simplifies the process of creating, reading, updating, and deleting configurations. The separation of user and admin roles ensures that critical operations are safeguarded, reducing the risk of accidental modifications or deletions by unauthorized personnel.

With its built-in logging and change history features, the tool provides accountability and transparency. Administrators can easily track changes, enabling effective audits and troubleshooting processes. This historical insight is invaluable when diagnosing issues or rolling back changes, making the tool an essential asset for maintaining system integrity.

Moreover, the potential for future enhancements, such as automated configuration deployment across multiple servers, positions this tool as a scalable solution to meet the evolving needs of modern IT infrastructures. By allowing for greater automation and control, the Basic Configuration Management Tool not only increases efficiency but also minimizes human error, thus ensuring a more stable and secure operating environment.

In summary, this tool lays a solid foundation for effective configuration management, enhancing the overall operational capabilities of IT teams. Its straightforward design, combined with powerful functionalities, makes it a vital resource in managing configurations efficiently and effectively in any organization. As the tool continues to evolve, it will undoubtedly play a pivotal role in the automation and optimization of configuration management tasks, furth er supporting IT and DevOps initiatives.

CHAPTER-11

REFERENCES

- https://realpython.com/python-gui-tkinter/

- https://youtu.be/VMP1oQOxfM0?si=pa26e_VikeImr2bo

- https://www.coursera.org/learn/python-for-applied-data-science-ai/home/module/3 -{coursera coach}