Week 3-Task 2

PROJECT TEMPERATURE CONVERTER

PREPARED BY

Sana Fatima

sana.faati@gmail.com

PREPARED TO

A&D Tech

ad.techinnov25@gmail.com

TABLE OF CONTENTS

Problem Statement

Implementing the Program

1.Use an array or list data structure to store items dynamically.

Implement functions for:

- Adding an item to the list.
- Removing an item from the list by index or value.

- Displaying all items in the list.

2.Ensure your list can handle a variable number of items and handle cases where the list is empty.

3. User Interface:

- Create a simple menu system that lets users choose to add, remove, or view items in the list.
- Provide clear instructions and feedback based on user actions (e.g., confirmation of added items, errors when trying to remove an item that does not exist.

SOLUTION OVER VIEW

The solution overview for this program is:

This C++ project implements a simple singly linked list data structure with basic operations such as adding, removing, and displaying items. The main components include:

1. **List Class**: Encapsulates the linked list functionality.
   - **Node Struct**: Represents a node in the linked list, containing the data (string) and a pointer to the next node.
   - **List Methods**:
     - **push**: Adds an item to the end of the list.
     - **erase**: Removes an item from the list by index.
     - **remove**: Removes an item from the list by value.
     - **empty**: Checks if the list is empty.
     - **print**: Displays all items in the list.
     - **getSize**: Returns the current size of the list.
2. **Main Program**: Provides a menu-driven interface for the user to interact with the list.
   - **add_item**: Prompts the user to add an item to the list.
   - **remove_item_by_index**: Prompts the user to remove an item from the list by its index.
   - **remove_item_by_value**: Prompts the user to remove an item from the list by its value.
   - **display_items**: Displays all items in the list.

CHALLENGES AND RESOLUTIONS

Boundary Conditions for Removal

- **Challenge**: Handling edge cases such as removing items from an empty list or invalid indices.
- **Resolution**: Checks are added to methods like erase and remove to ensure the list is not empty and the provided index is valid. Appropriate error messages are displayed.

User Input Handling

- **Challenge**: Ensuring that user inputs are valid and appropriately handled.
- **Resolution**: Prompts are provided to guide the user, and input is validated to ensure indices are within valid ranges.

Linked List Traversal

- **Challenge**: Efficiently traversing the linked list for operations like adding, removing, and printing.
- **Resolution**: Iterative traversal is used, and pointers are managed carefully to avoid errors.

INSTRUCTIONS

- The main function displays a menu with Five options:
    - Add an item.
    - Remove by index.
    - Remove by value(the name of item).
    - Display the items.
    - Exit the program.
- The user selects an option and enters the item that is saved at specific index. The program then add more items or remove them by index or value and to check the list we can also display it.