Developer Guide: Expense Tracking System

Overview

This program implements a simple command-line expense tracking system in C++. The system allows users to register, login, add, view, delete, and calculate total expenses. It uses file handling for persistent data storage. This guide covers the structure of the program, the purpose of each function, and the logic behind the expense tracking operations.

Table of Contents

---

1. Program Flow

The program follows a simple flow:

1. The user is welcomed with the **Starting Menu**, where they can either:
    - Register as a new user
    - Login to an existing account
    - Access the helpline or exit the program.
2. After login, the user is taken to a menu where they can:
    - Add an expense
    - View all expenses
    - Delete an expense
    - View total expenses for the logged-in user
3. The user can choose to log out from the system at any time.

2. Function Descriptions

StartingMenu()

- This is the main entry point where the user chooses to either register, log in, access the helpline, or exit the program.
- It uses a `do-while` loop to keep showing the menu until the user selects the option to exit.
- Invalid input is handled by calling `HandleInvalidInput()`.

RegisterUser()

- Registers a new user by asking for their name and password.
- It checks if a file with the user's name already exists. If so, the registration is aborted.
- If the file does not exist, it creates a new file named `<username>.txt` to store the user's credentials.

LoginMenu()

- Allows users to login using their credentials.
- It verifies the user's name and password by reading from the file created during registration.
- If login is successful, the user is presented with options to add, view, delete expenses, or calculate total expenses.

AddExpense()

- This function allows users to add an expense by specifying a description and amount.
- The expense is saved in a file named `<username>_expenses.txt` in an appending manner.

ViewExpenses()

- Displays all the expenses of the logged-in user by reading the `<username>_expenses.txt` file.
- It outputs the description and amount of each expense in a formatted table.
- If no expenses exist, it notifies the user.

DeleteExpense()

- Allows the user to delete a specific expense by description.
- The function creates a temporary file (`temp.txt`), writes all the expenses except the one to be deleted, and then renames the temp file to overwrite the original file.
- If no matching expense is found, it informs the user.

TotalExpenses()

- Calculates the total of all expenses by reading through the `<username>_expenses.txt` file.
- It sums up the amounts and displays the total in PKR.

HandleInvalidInput()

- This function is called when the user enters an invalid choice (e.g., a number outside the valid range).
- It clears the input buffer and prompts the user to re-enter a valid choice.

3. Error Handling

The program uses a robust error handling mechanism to ensure the user is always informed of potential issues:

- If the user tries to register with an already existing name, it displays an error message.
- If login credentials are incorrect or a file doesn't exist, it provides the user with a clear error message.
- File operations (reading and writing) are surrounded by checks to ensure that the file is successfully opened.
- Invalid inputs are handled by clearing the input buffer and prompting the user for correct input using `HandleInvalidInput()`.

4. Data Storage

The program stores user credentials and expenses in separate text files:

- **User Credentials**: Each user's credentials are stored in a file named `<username>.txt`, which contains the user's name and password.
- **Expenses**: Each user's expenses are stored in a separate file named `<username>_expenses.txt`. Each line in this file contains an expense in the format `description:amount`.

5. User Interaction

The interaction with the program is entirely text-based. The user is prompted to enter their choice from a menu or provide specific input for tasks like adding an expense or deleting one. Clear and concise messages are displayed throughout the program to guide the user.

6. Code Structure and Libraries

- **Libraries Used**:
    - `<iostream>`: For input/output operations.
    - `<fstream>`: For file handling (reading and writing files).
    - `<string>`: For string manipulation.
    - `<limits>`: For handling invalid numeric inputs.
    - `<iomanip>`: For formatting output (e.g., setting precision for floating point numbers).

- **Function Breakdown**:
    - The program is divided into different functions, each handling a specific task. This modular approach makes the code easy to understand and maintain.
    - Error handling is separated into a distinct function (`HandleInvalidInput()`), promoting code reusability and clarity.

Conclusion

This Expense Tracking System is a simple yet functional application for managing personal expenses. It incorporates basic file handling, error handling, and formatted output to ensure a smooth user experience. The code structure is clear and modular, making it easy to extend with additional features, such as data encryption for password storage or a more advanced user interface.