EXPENSE TRACKER PROJECT REPORT

PROJECT TITLE

Expense Tracker Application Using Python and SQLite

INTRODUCTION

This project is a desktop-based Expense Tracker application developed in Python. It allows users to manage their daily expenses efficiently by adding, viewing, editing, and deleting records stored in an SQLite database.

OBJECTIVES

- To create a simple and user-friendly interface for tracking expenses.
- To provide functionalities to add, update, and delete expense records.
- To enable easy retrieval and visualization of expense details.
- To implement data persistence using SQLite database.
- To enhance the usability using Tkinter GUI and DateEntry widget for date selection.

TOOLS AND TECHNOLOGIES USED

- Programming Language: Python
- GUI Library: Tkinter, tkcalendar (DateEntry)
- Database: SQLite
- IDE: Visual Studio Code

FEATURES

- Add an expense with details like date, payee, description, amount, and mode of payment.

- View all expenses in a tabular format with the ability to scroll through records.
- Edit existing expense records.
- Delete single or all expense records with confirmation prompts.
- Convert expense details to readable sentences for easy understanding.
- Clear input fields to enter new expenses swiftly.

DESIGN AND IMPLEMENTATION

Database

- SQLite database named "Expense Tracker.db".
- Single table "ExpenseTracker" with fields:
    - ID (Primary Key)
    - Date (Date of expense)
    - Payee (Whom the expense was paid to)
    - Description (Purpose of the expense)
    - Amount (Expense amount in float)
    - Mode of Payment (Cash, Card, UPI, etc.)

User Interface

- Built using Tkinter with distinct frames:
    - Data Entry Frame for input fields.
    - Buttons Frame for functional buttons.
    - Table Frame displaying expense records.
- Custom fonts and colors applied for aesthetics.
- Date selection done using the DateEntry widget from tkcalendar.

Functionality

- All data operations (Add, Edit, Delete) are synchronized with the SQLite database.
- Error handling includes input validation and confirmation of dialogs.
- The Treeview widget from tkinter.ttk displays the expense list dynamically.

CHALLENGES FACED

- Ensuring proper synchronization between UI elements and the database.
- Handling date formats appropriately for display and storage.
- Implementing editable rows in the Treeview with correct updates to the database.

FUTURE ENHANCEMENTS

- Add expense categorization and filtering options.
- Provide data export to CSV or Excel.
- Implement charts to visualize monthly or yearly expense trends.
- Add user authentication for personal expense tracking.

CONCLUSION

This Expense Tracker project demonstrates the integration of Python, SQLite, and Tkinter to build a functional and practical application. It helps users maintain a record of their expenditures easily and serves as a good example of beginner-to-intermediate level GUI database programming.