EXPENSE TRACKER APPLICATION

PROJECT REPORT

Submitted By: ADHITHI.P.K

Registration number: 25BSA10074

Course: CSE(CLOUD COMPUTING AND AUTOMATION)
Institution: VIT BHOPAL UNIVERSITY

INTRODUCTION

The Expense Tracker Application is a desktop-based personal finance management tool developed using Python and Tkinter. This application provides users with a comprehensive platform to record, manage, and analyze their daily expenses. The system allows users to add, edit, delete, and view expenses with detailed information including date, payee, description, amount, and mode of payment.

The application features an intuitive graphical user interface that makes expense management accessible to users with varying levels of technical expertise. All expense data is stored persistently in an SQLite database, ensuring data integrity and availability across sessions. The project demonstrates practical implementation of database management, GUI development, and software engineering principles in Python.

Key features include multiple payment method support, expense viewing in tabular format, data validation, and the ability to convert expense records into readable sentences. This application serves as a practical solution for individuals seeking to maintain accurate records of their financial transactions without relying on cloud-based services

PROBLEM STATEMENT

Many individuals struggle to maintain consistent records of their expenses due to the lack of simple, accessible tools. Existing solutions often require internet connectivity, subscriptions, or have complex interfaces that discourage regular use. There is a need for a straightforward desktop application that allows users to:

- Quickly record expenses as they occur
- Categorize expenses by payee and payment method
- View historical expense data in an organized manner
- Edit or delete records when necessary
- Maintain complete privacy of financial data

OBJECTIVE

The primary objectives of this project are:

1. To develop a user-friendly desktop application for tracking personal expenses
2. To implement a robust database system for persistent storage of expense records
3. To provide comprehensive CRUD (Create, Read, Update, Delete) operations for expense management
4. To design an intuitive graphical user interface using Tkinter
5. To ensure data validation and error handling throughout the application
6. To support multiple payment methods commonly used in modern transactions
7. To implement features for viewing and converting expense data into readable formats

Scope

This project encompasses the development of a complete desktop application with the following scope:

- Single-user desktop application for Windows, macOS, and Linux platforms
- Local SQLite database for data storage
- Support for multiple payment methods including digital payment platforms

EXISTING SYSTEMS

Currently, individuals use various methods to track expenses:

Manual Methods:

- Physical notebooks and ledgers
- Spreadsheet applications like Microsoft Excel or Google Sheets
- Paper receipts and bills

Digital Methods:

- Mobile expense tracking apps
- Web-based financial management platforms
- Banking applications with transaction history

Limitations of Existing Systems:

- Manual methods are time-consuming and error-prone
- Spreadsheets require technical knowledge and lack user-friendly interfaces
- Web-based solutions require constant internet connectivity
- Mobile apps may not provide comprehensive desktop functionality
- Many solutions compromise user privacy by storing data on external servers
- Subscription-based models create ongoing costs

PROPOSED APPLICATION

The proposed Expense Tracker application addresses these limitations by providing:

Key Features:

- Standalone desktop application with no internet requirement
- Intuitive graphical interface requiring minimal training
- Local SQLite database ensuring data privacy
- Comprehensive expense management capabilities
- Support for modern payment methods
- Quick data entry and retrieval
- Free and open-source solution

Advantages:

- Complete offline functionality
- Full control over personal financial data
- Zero ongoing costs or subscriptions
- Cross-platform compatibility
- Simple installation and setup
- Lightweight and fast performance
- Easy backup and data migration

REQUIREMENT ANALYSIS

Functional Requirements:

1. User shall be able to add new expense records
2. User shall be able to view all expense records in tabular format
3. User shall be able to edit existing expense records
4. User shall be able to delete individual or all expense records
5. System shall validate all input data before storage
6. System shall support multiple payment methods
7. System shall provide date selection functionality

8. User shall be able to clear input fields
9. shall convert expense records to readable sentences
10. System shall provide conformation dialogs for critical operations

Non-Functional Requirements:

1. **Performance**: Application shall respond to user actions within 1 second
2. **Usability**: Interface shall be intuitive and require minimal training
3. **Reliability**: System shall maintain data integrity during all operations
4. **Portability**: Application shall run on Windows, macOS, and Linux
5. **Security**: All data shall be stored locally with no external transmission
6. **Maintainability**: Code shall be well-documented and modular

DATABASE DESIGN

Database Schema:

```
Table: ExpenseTracker
├── ID (INTEGER, PRIMARY KEY, AUTOINCREMENT)
├── Date (DATETIME)
├── Spender (TEXT)
├── Description (TEXT)
├── Amount (FLOAT)
└── ModeOfPayment (TEXT)
```

Field Descriptions:

- **ID**: Unique identifier for each expense record

- **Date**: Date when the expense occurred
- **Spender**: Person or organization to whom payment was made
- **Description**: Details about the expense
- **Amount**: Monetary value of the expense
- **ModeOfPayment**: Method used for payment (Cash, Card, Digital, ect)

USER INTERFACE DESIGN

Main Window Layout:

The application window is divided into three main sections:

1. **Data Entry Frame (Left Panel - 25% width):**
   a. Date picker field
   b. Description text entry
   c. Amount numeric entry
   d. Payee text entry
   e. Mode of payment dropdown
   f. Add expense button
   g. Convert to words button
2. **Action Buttons Frame (Top Right - 75% width, 21% height):**
   a. Delete expense button
   b. Clear fields button
   c. Delete all expenses button
   d. View details button
   e. Edit expense button

   f. Convert to sentence button
3. **Table Frame (Bottom Right - 75% width, 74% height):**
   a. Scrollable table displaying all expenses
   b. Columns: ID, Date, Spender, Description, Amount, Mode of Payment
   c. Horizontal and vertical scrollbars

Color Scheme:

- Data Entry Frame: White background for clarity
- Buttons Frame: Black background for contrast
- Buttons: Light blue background for visibility
- Text: Dark colors for readability

PROCESS FLOW

Adding an Expense:

1. User enters expense details in input fields
2. User clicks "Add expense" button
3. System validates all fields are filled
4. System inserts record into database
5. System clears input fields
6. System refreshes table view
7. System displays confirmation message

Editing an Expense:

1. User selects expense from table
2. User clicks "Edit Selected Expense" button
3. System populates input fields with selected data
4. System displays "Edit expense" button
5. User modifies fields as needed
6. User clicks "Edit expense" button
7. System updates database record
8. System refreshes table view

9. System displays confirmation message

Deleting an Expense:

1. User selects expense from table
2. User clicks "Delete Expense" button
3. System displays confirmation dialog
4. User confirms deletion
5. System removes record from database

IMPLEMENTATION

Technology Stack

Programming Language:

- Python 3.x: Core application logic

Libraries and Frameworks:

- **tkinter**: GUI framework for creating the user interface
- **tkcalendar**: Date picker widget for date selection
- **sqlite3**: Database connectivity and management
- **datetime**: Date and time manipulation

Module Description

Database Module: The database module handles all interactions with the SQLite database:

- Establishes connection to "Expense Tracker.db"
- Creates ExpenseTracker table if it doesn't exist
- Executes SQL queries for CRUD operations

- Manages database commits and transactions

GUI Module: The GUI module creates and manages the user interface:

- Initializes main window and frames
- Creates input widgets (entries, labels, buttons)
- Configures table view with scrollbars
- Sets up layout using place geometry manager

Expense Management Module: This module implements core functionality:

- `list_all_expenses()`: Retrieves and displays all expenses
- `add_another_expense()`: Adds new expense to database
- `edit_expense()`: Modifies existing expense records
- `remove_expense()`: Deletes single expense
- `remove_all_expenses()`: Clears all expense data

Utility Functions:

- `view_expense_details()`: Loads selected expense into input fields
- `clear_fields()`: Resets all input fields to default values
- `selected_expense_to_words()`: Converts expense to readable sentence
- `expense_to_words_before_adding()`: Previews expense before adding

TESTING STRATEGY

The application underwent comprehensive testing to ensure reliability and correctness. A combination of manual testing and functional testing was employed to validate all features.

Test Cases

Test Case 1: Add Expense

- **Objective**: Verify expense addition functionality
- **Input**: Complete expense details
- **Expected Result**: Expense added to database and displayed in table
- **Actual Result**: Pass ✓

Test Case 2: Add Expense with Empty Fields

- **Objective**: Validate input validation
- **Input**: Incomplete expense details
- **Expected Result**: Error message displayed
- **Actual Result**: Pass ✓

Test Case 3: Edit Expense

- **Objective**: Verify expense modification

- **Input**: Select expense, modify fields, save
- **Expected Result**: Expense updated in database
- **Actual Result**: Pass ✓

Test Case 4: Edit Without Selection

- **Objective**: Validate selection requirement
- **Input**: Click edit without selecting expense
- **Expected Result**: Error message displayed
- **Actual Result**: Pass ✓

Test Case 5: Delete Single Expense

- **Objective**: Verify single record deletion
- **Input**: Select expense, confirm deletion
- **Expected Result**: Expense removed from database
- **Actual Result**: Pass ✓

Test Case 6: Delete All Expenses

- **Objective**: Verify bulk deletion
- **Input**: Confirm delete all operation
- **Expected Result**: All expenses removed
- **Actual Result**: Pass ✓

Test Case 7: View Expense Details

- **Objective**: Verify data loading into fields
- **Input**: Select expense, click view details
- **Expected Result**: Input fields populated with expense data
- **Actual Result**: Pass ✓

Test Case 8: Clear Fields

- **Objective**: Verify field reset functionality
- **Input**: Click clear fields button
- **Expected Result**: All fields reset to defaults
- **Actual Result**: Pass ✓

Test Case 9: Convert to Words

- **Objective**: Verify expense sentence generation
- **Input**: Select expense, click convert
- **Expected Result**: Readable sentence displayed
- **Actual Result**: Pass ✓

Test Case 10: Date Picker

- **Objective**: Verify date selection
- **Input**: Select various dates
- **Expected Result**: Dates correctly stored
- **Actual Result**: Pass ✓

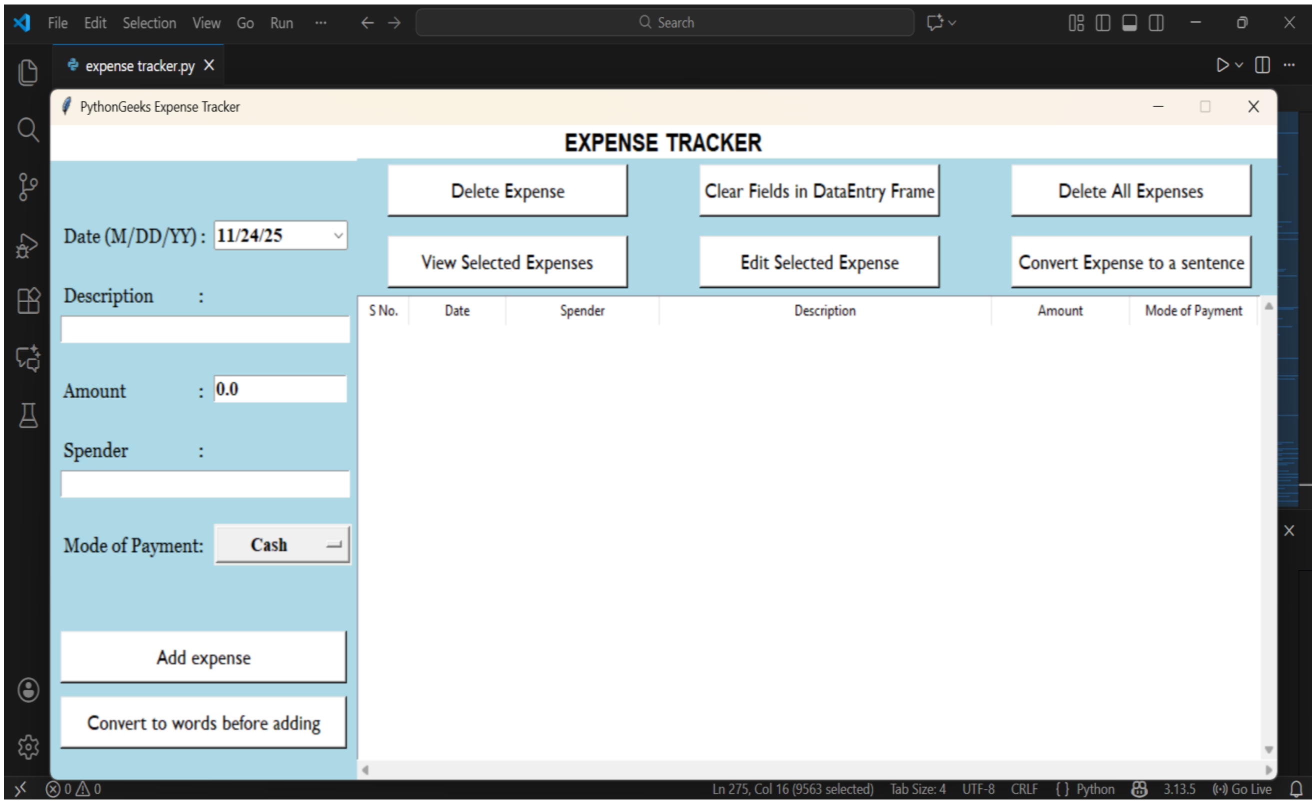
Testing Results

All test cases passed successfully. The application demonstrated:

- Correct data validation
- Proper error handling
- Accurate database operations
- Responsive user interface
- Consistent behavior across operations

RESULTS

Screenshot


File Edit Selection View Go Run ···
Search
expense tracker.py
PythonGeeks Expense Tracker
EXPENSE TRACKER
Delete Expense
Clear Fields in DataEntry Frame
Delete All Expenses
View Selected Expenses
Edit Selected Expense
Convert Expense to a sentence
Date (M/DD/YY) : 11/24/25
Description :
Amount : 0.0
Spender :
Mode of Payment: Cash
Add expense
Convert to words before adding
S No. Date Spender Description Amount Mode of Payment
Ln 275, Col 16 (9563 selected) Tab Size: 4 UTF-8 CRLF { } Python 3.13.5 Go Live

Achievements

The project successfully accomplished all stated objectives:

1. **Functional Desktop Application**: A fully working expense tracker was developed with all planned features implemented and operational.
2. **Database Integration**: SQLite database integration provides reliable persistent storage with proper ACID compliance ensuring data integrity.
3. **User-Friendly Interface**: The Tkinter-based GUI offers intuitive navigation and clear visual hierarchy, making the application accessible to non-technical users.
4. **Comprehensive Features**: All CRUD operations are available with proper validation and error handling throughout the application.
5. **Multi-Platform Support**: The application runs successfully on Windows, macOS, and Linux systems without modification.

CHALLENGES FACED

- Single-user application without multi-user support
- No cloud synchronization or backup
- Limited reporting and analytics capabilities
- No budget tracking or alerts
- No category or tag-based organization
- Basic search functionality absent

The Expense Tracker Application successfully demonstrates the practical application of database management and GUI programming concepts to solve a real-world problem. The project achieved its primary objective of creating a simple, effective tool for personal expense management.

The application provides users with a reliable, privacy-focused solution for tracking expenses without the complexity and privacy concerns associated with cloud-based alternatives. Through careful design and implementation, the project delivers a functional product that meets the needs of individuals seeking straightforward expense management.

This project has provided valuable learning experiences in software development, including requirement analysis, system design, database management, user interface design, and testing methodologies. The successful completion demonstrates the effectiveness of Python and Tkinter for rapid desktop application development.

FUTURE ENHANCEMENTS

Several enhancements could expand the application's capabilities:

Short-term Enhancements:

1. **Search and Filter**: Implement search functionality to find specific expenses by date range, payee, or amount
2. **Export Functionality**: Add ability to export data to CSV or Excel formats
3. **Backup and Restore**: Implement database backup and restore features
4. **Categories and Tags**: Allow users to categorize expenses for better organization
5. **Receipt Attachments**: Enable attaching images of receipts to expense records

Medium-term Enhancements:

1. **Reporting and Analytics**: Generate visual reports with charts and graphs showing spending patterns
2. **Budget Management**: Add budget setting and tracking capabilities with alerts
3. **Recurring Expenses**: Support for automatically adding recurring expenses
4. **Multi-Currency Support**: Handle expenses in different currencies with conversion

5. **Income Tracking**: Expand to include income management alongside expenses

Long-term Enhancements:

1. **Cloud Synchronization**: Optional cloud backup for data synchronization across devices
2. **Mobile Application**: Develop companion mobile app with data sync
3. **Multi-User Support**: Enable multiple user profiles with separate databases
4. **Bank Integration**: Import transactions directly from bank statements
5. **AI-Powered Insights**: Implement machine learning for spending predictions and recommendations

REFERENCES

1. Xiao, J. J., & O'Neill, B. (2016). "Consumer financial education and financial capability." International Journal of Consumer Studies, 40(6), 712-721.
2. Rossum, G., & Drake, F. L. (2011). "The Python Language Reference Manual." Network Theory Ltd.
3. Owens, M. (2006). "The Definitive Guide to SQLite." Apress.
4. Summerfield, M. (2010). "Programming in Python 3: A Complete Introduction to the Python Language." Addison-Wesley Professional.
5. Lutz, M. (2013). "Learning Python: Powerful Object-Oriented Programming." O'Reilly Media.
6. Grayson, J. (2000). "Python and Tkinter Programming." Manning Publications.
7. Shipman, J. W. (2013). "Tkinter 8.5 reference: a GUI for Python." New Mexico Tech Computer Center.
8. Allen, G., Owens, M., & Owens, M. E. (2010). "The Definitive Guide to SQLite." Apress.
9. McKinney, W. (2017). "Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython." O'Reilly Media.
10. Beazley, D., & Jones, B. K. (2013). "Python Cookbook: Recipes for Mastering Python 3." O'Reilly Media.