Introduction

Expense Tracker is an GUI application that allows its users to manage and attain financial freedom. It is a functional tool to help adding, viewing and managing the personal finances adequately.

In this project, expense tracker provides users useful functionalities like adding expenses, viewing expenses and overviewing expenses. Users can add their expenses depending on their categories like food, transport, shopping, housing and others; and stores the date when they made their expenses. Additionally, users can also view their expenses data in a monthly tabular form and overview their data in a pie chart.

This project uses the Python Tkinter GUI for our user-friendly interface and sqlite3 as database system to store users and expenses information. Using our Expense Tracker, the following activities are our functional features of the application:

- Login
- Sign Up
- Forgot Password
- Add Expenses
- View Expenses
- Delete Expenses
- Overview
- Export Expenses
- Logout

In conclusion, our application lets the users to make quick and easy addition and visualization of the expenses they make.

Aim

The Expense Tracker uses simple logic and covers the basic tracking of the personal finances. The foremost aim of our project is to make users aware of the expenses they make and to make the everyday tracking experiences fun.

Objectives

The predominant objectives of our application addresses solely on simple and interactive user interface where user can utilize productive finance management and visualization. These objectives are:

1. Building an intuitive and user-friendly GUI using Tkinter that simplifies the process of entering, viewing, and managing personal expenses.
2. Giving users control over their short-term and long-term spending.
3. Enhancing the readability of the user expenses by categorizing them into different categories.
4. Letting users to export their expense table in a .csv file.
5. Storing the overall user information and their expenses in a structured database.
6. Helping to visualize the monthly expenditures in a pie-chart.

Problem Statement

Organizing personal expenses could be very annoying thing to do, using the classic methods of expense tracking like using physical notepad. A person can encounter input errors which can lead to incorrect financial records. There are many expense tracking applications which are complex to use which leads to inconsistent tracking that hinders financial analysis and decision making.

Our project enhances the simplicity of the tracking process by making it easy-to-use with our interactive user experience and minimal but efficient functionalities like adding, viewing and overviewing expenses.

Features

The project accumulates the following key features:

- Data Storage: Using this application, data can be added are stored in a database, and can be retrieved efficiently.
- Export Expenses: The expenses that are stored, can be exported in a .csv file and opened in Microsoft Excel for viewing and storing the expense information in the user's local directory.
- Data Integrity: The application is protected from errors like invalid datatype and false information.
- Data Sorting: The data entered by the user can be sorted in a monthly basis by the date the expenses is made.

Functional Requirements

The Expense Tracker application is comprised of the following key features:

- Login: The Login checks if the user already exists in the database and lets them to login and access the application. If the user does not exist, then it asks them to create an account.
- Sign-up: The Sign-up allows the user to create their account if they are not registered before. If the username already exists in the database, then that username cannot be used by other users.
- Logout: The Logout allows the user to logout from their account and redirects them into login page.
- Add Expense: Add Expense allows the user to add the expenses according to the expense item's name, price, category and date. After listing the items, the add button stores the information in the database.
- View Expense: View Expense permits the user to view their listed expenses from database in a tabular form. The user can also change the order of the expense table which is already sorted in a monthly format.
- Overview Expense: Overview Expenses makes the visualization of the data understandable by allowing the users to view their expense information in a pie-chart, which is sorted by the categories the expenses are stored.

Non-Functional Requirements

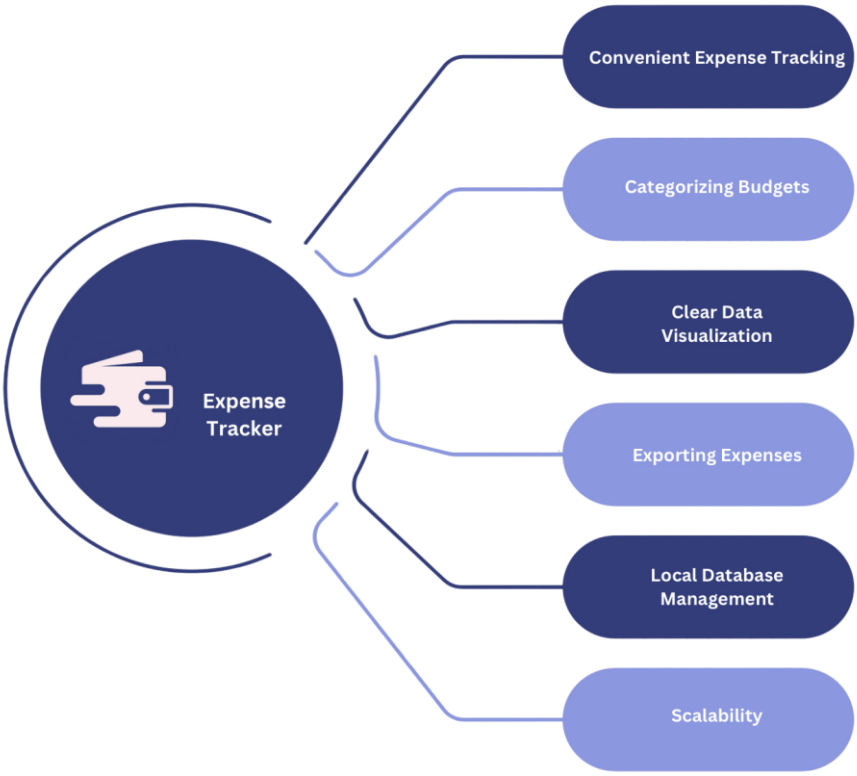
The Non-Functional Requirements of the Expense Tracker includes:

- Minimal but effective GUI: The application is comprised of simple and subtle GUI which makes it easier for the users to interact with the application.
- Export .csv: While this function is not a major requirement, it allows users to export their all-time expenses and store it to be used later even when they don't have access to the application.
- Speedy Navigation: The application is very quick to navigate as it uses Python's Tkinter framework, and the lines of code are less compared to other application with the same core idea.
- Reliable: Users can rely on Expense Tracker as the functions written are very Pythonic.

Scope

Figure 1:

Scope of Expense Tracker



The major scope of Expense Tracker includes Convenient Expense Tracking which is possible only with our user-interface kept simple and understandable. It also includes Categorizing Budgets by categories like: Food, Transport, Shopping, Housing and Others and provides Clear Data Visualization through pie chart which makes the users to understand the expenses through category where they are spending. Furthermore, it also allows users to export their data in a .CSV file which can also be viewed in Microsoft Excel.

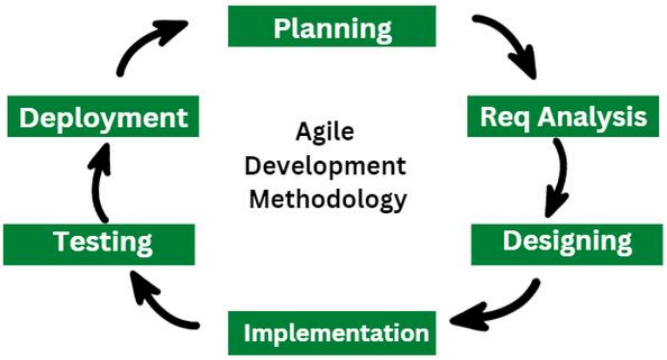
Finally, our project stores data locally where the data can be stored individually in user's local directory. As the data is stored locally, the chance of data scalability is very high.

Development Methodologies

Methodology

Figure 2:

Agile Methodology



The Agile Methodology was adapted by our team to complete the development of this application. At the beginning, we gathered the requirements to build the application. We analyzed the requirements and only followed the ones that were practically important and felt possible. After, gathering all the requirements to build the application, we created a modern, subtle and eye-soothing design which was best suited for the application. We also prototyped our system design from the beginning to vision our ideas. We implemented the design and converted the prototype into a working application using Python, TKinter and SQlite3. Gradually, we tested the working mechanism of the code through some testing. We did small sprints every once-in-a-while, and in each sprint, we incremented new functionalities whilst refining the existing ones. Finally, we deployed the actual working application on GitHub.

Tools and Technologies

Figure 3:

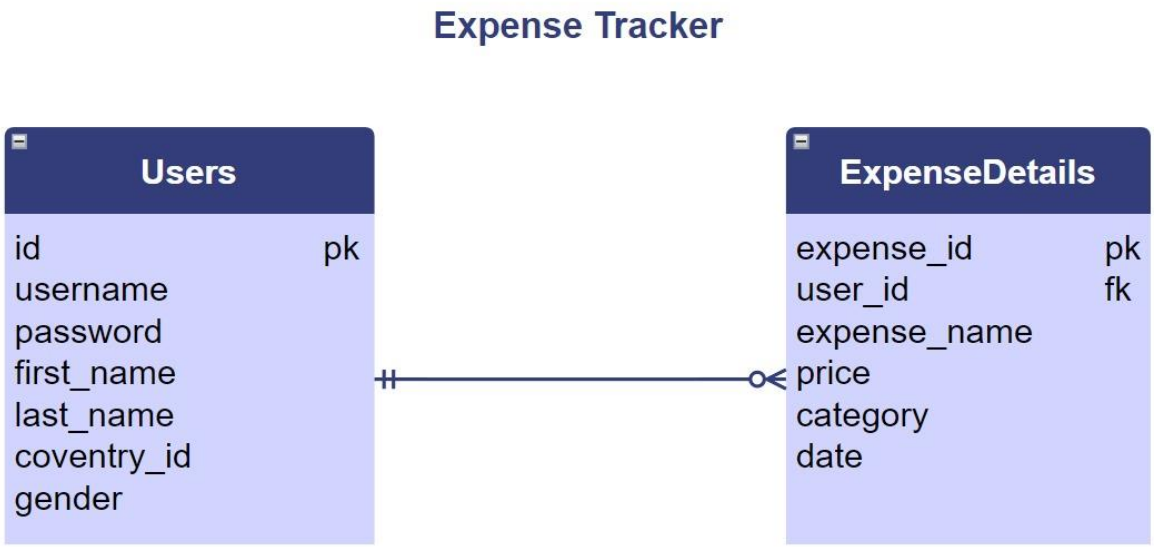
Tools and Technologies

The tools that were used to complete this application are: Figma, for designing and prototyping the application; Draw.io, to create Entity-Relationship Diagrams to visualize our database; VS Code, as an IDE(Integrated Development Environment) to write and debug the code; Discord, to run sprints reviews and scrum meetings; GitHub, to push the code in remote repository; Microsoft word, to write the documentation for our application; Sqlite3, to store the user information and expenses information and DB browser, to view the data information.

Conceptual Diagram

Figure 4:

Entity-Relationship Diagram



The diagram above is an Entity-Relationship Diagram of our Expense Tracker's database showing the relation between the users and the expense details table.

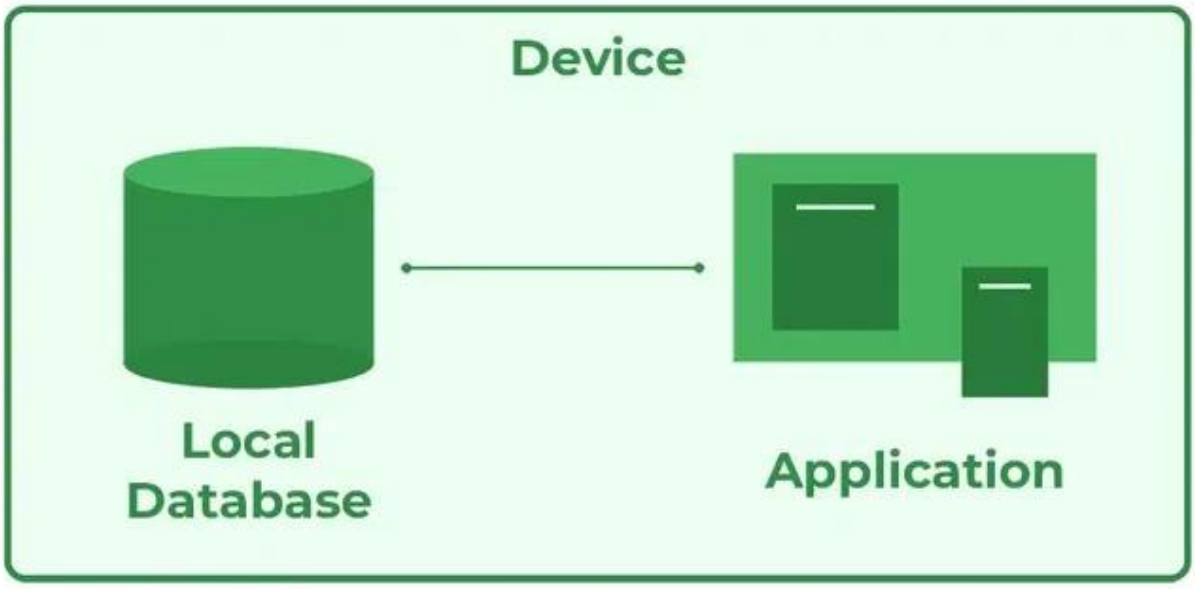
System Architecture

In the Development of the Expense Tracker application, we have utilized one-tier system architecture. The presentation layer or user interface is operated by the user on their personal computer, and the data layer or the data structure is stored or kept at the database at the local directory of the user's computer. Having these two components in the same location represents a one-tier architecture, as opposed to a two-tier architecture. Other kinds of multi-tier architectures add additional layers in distributed software design.

Since, the database is made directly available to the user and the server of the application also resides locally, the application enables users to directly interact with the database and execute operations. Thus, user, server and database all reside locally which makes our project a one-tier system architecture.

Figure 5:

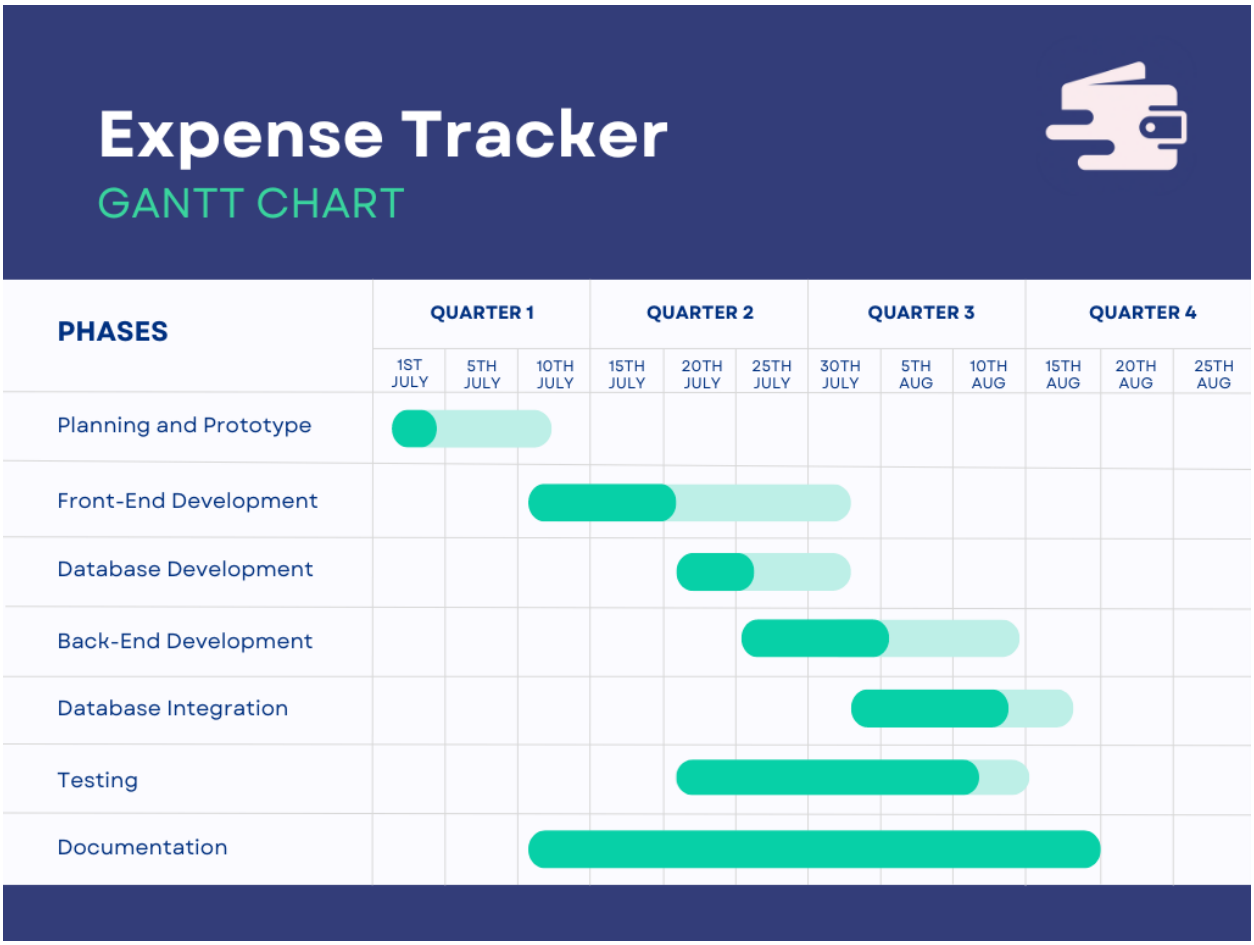
One-Tier System Architecture



Project Plans

Figure 6:

Project Gantt Chart



Prototype