**Industrial Internship Report on**

**”Expense Tracker”**

Prepared by

[Atchaya S]

|  |
| --- |
| Executive Summary |
| This report provides details of the Industrial Internship provided by upskill Campus and The IoT Academy in collaboration with Industrial Partner UniConverge Technologies Pvt Ltd (UCT).  This internship was focused on a project/problem statement provided by UCT. We had to finish the project including the report in 4 weeks’ time.  My project was (“Expense Tracker”)  This internship gave me a very good opportunity to get exposure to Industrial problems and design/implement solution for that. It was an overall great experience to have this internship. |

Contents

[1 Preface 3](#_Toc205399547)

[2 Introduction 4](#_Toc205399548)

[2.1 About UniConverge Technologies Pvt Ltd 4](#_Toc205399549)

[2.2 About upskill Campus (USC) 4](#_Toc205399550)

[2.3 Objectives of this Internship program 4](#_Toc205399551)

[2.4 Reference 5](#_Toc205399552)

[2.5 Glossary 5](#_Toc205399553)

[3 Problem Statement 6](#_Toc205399554)

[4 Existing and Proposed solution 7](#_Toc205399555)

[4.1 Code Submission (github link): 8](#_Toc205399556)

[4.2 Report Submission (Github link): 8](#_Toc205399557)

[5 Proposed Design / Model 9](#_Toc205399558)

[5.1 Interfaces (if applicable) 9](#_Toc205399559)

[6 Performance Test 10](#_Toc205399560)

[6.1 Test Plan / Test Cases 10](#_Toc205399561)

[6.2 Test Procedure 11](#_Toc205399562)

[6.3 Performance Outcome 12](#_Toc205399563)

[7 My Learnings 14](#_Toc205399564)

[8 Future Work Scope 15](#_Toc205399565)

# Preface

Time management and financial discipline are crucial in both personal and professional life. One of the simplest ways to improve both is by regularly tracking and analyzing our expenses. This thought inspired me to develop a project titled **"Expense Tracker – A Daily Spending Manager"** as part of my internship program at **Upskill Campus**, in association with **UniConverge Technologies Pvt. Ltd.** and **The IoT Academy**.

This internship provided me with an opportunity to gain valuable real-world exposure in building software applications from scratch. I used **Python** as the core programming language, along with **Tkinter** for building the graphical user interface and **SQLite** for local data storage. The project helped me understand how small-scale solutions can solve meaningful problems in daily life.

Throughout this 1-month internship, I enhanced my programming skills, gained a better understanding of project planning, and improved my problem-solving ability. I also learned how to write cleaner, modular code and document it properly — both of which are essential skills for any future developer.

I would like to express my sincere gratitude to **Upskill Campus**, **UCT**, and all mentors who guided me during this learning journey. I also thank my faculty and peers who supported me throughout this internship.

This project is a small step toward becoming a better developer and solving real-world problems using technology.

# Introduction

## About UniConverge Technologies Pvt Ltd

A company established in 2013 and working in Digital Transformation domain and providing Industrial solutions with prime focus on sustainability and RoI.

For developing its products and solutions it is leveraging various Cutting Edge Technologies e.g. Internet of Things (IoT), Cyber Security, Cloud computing (AWS, Azure), Machine Learning, Communication Technologies (4G/5G/LoRaWAN), Java Full Stack, Python, Front end etc.

## About upskill Campus (USC)

upskill Campus along with The IoT Academy and in association with Uniconverge technologies has facilitated the smooth execution of the complete internship process.

USC is a career development platform that delivers personalized executive coaching in a more affordable, scalable and measurable way.

The IoT Academy

The IoT academy is EdTech Division of UCT that is running long executive certification programs in collaboration with EICT Academy, IITK, IITR and IITG in multiple domains.

## Objectives of this Internship program

The primary objective of this internship program is to bridge the gap between academic learning and industry expectations. Through this hands-on experience, the program aims to equip students with real-world skills that prepare them for future professional roles.

✅ Key Objectives: The main objectives of this internship program are as follows:

** To gain practical exposure**

* **Apply theoretical knowledge to real-world problems using Python and GUI-based applications.**

** To build a working software project**

* **Design and develop a complete application (Expense Tracker) from scratch, including planning, coding, testing, and documentation.**

** To enhance technical and programming skills**

* **Improve proficiency in Python programming, database handling with SQLite, and UI design using Tkinter.**

** To develop analytical and problem-solving abilities**

* **Learn how to identify real-life issues and implement efficient, user-friendly solutions.**

** To learn project documentation and version control**

* **Understand the importance of writing technical reports and using platforms like GitHub for code submission.**

** To build confidence and independence**

* **Gain experience in managing tasks independently and build the confidence to handle industry-level challenges.**

** To improve communication and presentation skills**

* **Document the entire development process and present the final outcome in a structured format.**

## Reference

 **Python Official Documentation**

<https://docs.python.org/3/>

 **Tkinter GUI Documentation**

<https://docs.python.org/3/library/tkinter.html>

You can include these 2 in your report under References section.

## Glossary

|  |  |
| --- | --- |
| **Term** | **Definition** |
| **Python** | A high-level, interpreted programming language used for developing applications. |
| **Tkinter** | A built-in Python library used for creating Graphical User Interfaces (GUI). |
| **SQLite** | A lightweight, file-based database engine used to store application data locally. |
| **GUI** | Graphical User Interface – a visual interface for interacting with the application. |
| **CSV** | Comma-Separated Values – a file format used for storing tabular data in plain text. |

# Problem Statement

In today’s fast-moving world, individuals often struggle to manage their personal finances effectively. Small, daily expenses frequently go untracked, leading to poor financial habits, overspending, and lack of awareness about where money is going.

Although there are many mobile apps available for expense tracking, most of them are either paid, overly complex, or require internet access. Basic users often prefer a **simple, offline, and easy-to-use desktop solution** to monitor and record their daily expenses.

To solve this problem, a Python-based desktop application titled **“Expense Tracker – Daily Spending Manager”** is proposed. This application allows users to:

* Enter daily expenses with category, amount, and note
* Automatically record the date
* View all recorded expenses in a table
* Store data securely using a local SQLite database

The goal is to provide users with a **lightweight, offline tool** that encourages better financial awareness and daily budgeting — especially useful for students, freelancers, and working individuals.

# Existing and Proposed solution

Currently, there are many expense tracking tools and mobile apps available in the market such as **Walnut**, **Money Manager**, **Spendee**, and **Google Sheets**. While these tools offer advanced features like bank sync, cloud storage, and multi-user access, they also come with certain limitations:

* ❌ Most require **internet connectivity**
* ❌ Many apps are **paid** or offer limited features in free versions
* ❌ Some apps are **too complex** for basic users
* ❌ User data may be stored on **external servers**, raising privacy concerns
* ❌ No full control over exported or raw data

Hence, for a user looking for a **simple, offline, and personal expense tracking tool**, existing solutions may not be ideal.

* **4.2 Proposed Solution**

To overcome these limitations, a lightweight desktop application titled **“Expense Tracker – Daily Spending Manager”** is proposed. The application is built using **Python (Tkinter)** for GUI and **SQLite** for local storage. It is:

* ✅ **Completely offline**
* ✅ **Free and open-source**
* ✅ **User-friendly** interface for beginners
* ✅ **Stores data locally** without cloud dependency
* ✅ **Requires no installation** of external databases
* **🔧 Key Features of the Proposed App:**
* Add expenses with category, amount, and optional note
* Automatically record date of entry
* View expense records in a scrollable table
* Save all data in a local .db file
* Secure and easy to use on any computer with Python installed

This proposed solution provides a **simple, accessible, and customizable** way to track expenses and promotes good personal finance habits, especially for students and individuals who are just beginning to manage money consciously.

## Code Submission (github link):

## [https://github.com/Atchaya2095/upskillCampus](https://github.com/Atchaya2095/upskillCampus" \t "_new)

## Report Submission (Github link):

<https://github.com/Atchaya2095/upskillCampus/blob/main/README.md>

# Proposed Design / Model

The **Expense Tracker** is designed as a simple, standalone desktop application that allows users to enter, store, and view daily expenses. The project follows a **modular structure** with clean separation between the **user interface**, **business logic**, and **database operations**.

This project uses Python’s Tkinter library for the graphical user interface (GUI) and SQLite3 for local data storage.

1. **🔧 Design Modules**
2. **User Input Module**
   * Allows users to enter expense details such as category, amount, and optional note.
3. **Validation & Logic Module**
   * Validates the input (e.g., numeric amount), handles formatting and data flow.
4. **Database Module**
   * Interacts with SQLite to store and retrieve expenses securely.
5. **Display Module**
   * Shows a table of expenses using ttk.Treeview, allowing users to review past entries.
6. **🔁 Data Flow**

css

CopyEdit

[ User Input ] ──► [ Validation ] ──► [ SQLite Database ]

▲ │

│ ▼

[ Display in Table ] ◄────────── [ View Expenses ]

1. **📌 Why This Design Works**

* ✅ **Simple** for non-technical users
* ✅ **Offline** and lightweight (no internet or installation required)
* ✅ **Fast** due to minimal dependencies
* ✅ **Secure** because data is stored locally

1. **🧠 Optional Improvements in Future:**

* Monthly filters
* Export to CSV or Excel
* Chart-based analytics using Matplotlib
* Login system for multiple users

## Interfaces (if applicable)

The **Expense Tracker** application uses a simple and interactive **Graphical User Interface (GUI)** developed using Python’s Tkinter library. The interface is designed to be beginner-friendly and responsive, allowing users to enter and view expenses with ease

* **🔹 1. Task Entry Interface**
* **Components: Entry fields for Category, Amount, and Note.**
* **Function: Allows the user to input details of each expense.**
* **Validation: Ensures the amount is numeric and category is not empty.**
* **🔹 2. Button Interface**
* **Buttons Used:**
  + **Add Expense – To save the current entry**
  + **View Expenses – To fetch and display stored expenses**
* **Function: Connects the user’s action to the logic and database operations.**
* **🔹 3. Expense Display Interface**
* **Component: ttk.Treeview (from Tkinter)**
* **Function: Displays all stored expenses in a table format.**
* **Columns: Date, Category, Amount, Note**
* **Extra: A vertical scrollbar is included for better navigation.**
* **🔹** **🔹 4. Pop-up Message Interface**
* **Component: tk.messagebox**
* **Function: Provides feedback (success, warning, or error messages) to the user after actions like saving or input validation.**
* **Interface Characteristics:**
* **Clean layout with aligned elements**
* **Easy navigation and interaction**
* **Fast data display with real-time updates**

# Performance Test

Performance testing was conducted on the **Expense Tracker** application to ensure its responsiveness, reliability, and stability under normal usage. Since the application is lightweight and built using **Python + Tkinter + SQLite**, the primary performance metrics evaluated were **speed, memory usage, and UI responsiveness**.

**🔹 Test Setup:**

* **System**: Windows 10 (64-bit), 8GB RAM, Intel i5 Processor
* **Python Version**: 3.10
* **Testing Tool**: Manual observation, Python's time and psutil modules for memory profiling

**✅ Performance Parameters Tested:**

| **Feature** | **Test Result** | **Remarks** |
| --- | --- | --- |
| **Application Launch Time** | **< 1 second** | **Very fast due to lightweight structure** |
| **Timer/Clock Lag (if any)** | **Not applicable** | **No timer used in this project** |
| **Add Expense Execution Time** | **Instant (real-time)** | **Expense saved immediately to SQLite DB** |
| **View Expense Load Time** | **< 1 second** | **Displays full list instantly** |
| **Memory Usage** | **~30-50 MB** | **Low RAM usage during entire session** |
| **UI Responsiveness** | **Smooth** | **No freezing or button delay** |
| **Database Connection Time** | **Instant** | **Local DB connection is fast and stable** |

**🔹 Conclusion:**

The Expense Tracker application demonstrated **excellent performance** during all test cases. It runs smoothly on low-end systems, consumes minimal memory, and executes all actions with near-instant response. The system is highly suitable for personal use, especially for students and individual users.

## Test Plan / Test Cases

The following test plan outlines the key functionalities of the Expense Tracker application along with the expected and actual outcomes during testing.

| **Test Case ID** | **Test Description** | **Input Data** | **Expected Result** | **Actual Result** | **Status** |
| --- | --- | --- | --- | --- | --- |
| TC01 | Launch the application | – | Application should open without errors | Application launched successfully | ✅ Pass |
| TC02 | Add valid expense entry | Category: "Food", Amount: 100, Note: "Lunch" | Entry should be saved to the database | Entry saved successfully | ✅ Pass |
| TC03 | Add expense with empty category | Category: "", Amount: 200, Note: "Snacks" | Warning message should be shown | Warning displayed | ✅ Pass |
| TC04 | Add expense with non-numeric amount | Category: "Travel", Amount: "abc", Note: "Bus" | Error message should be shown | Error displayed | ✅ Pass |
| TC05 | View saved expenses | – | Table should display all past entries | Data displayed correctly | ✅ Pass |
| TC06 | Check if date is auto-recorded | Entry without date input | System should automatically insert current date | Date saved correctly | ✅ Pass |
| TC07 | Scroll through long list of expenses | 50+ records | Scrollbar should allow smooth scrolling | Scroll working | ✅ Pass |
| TC08 | Test database creation on first run | No DB file present | App should auto-create expenses.db | DB created successfully | ✅ Pass |

## Test Procedure

The test procedure outlines the step-by-step process used to validate the functionalities and performance of the **Expense Tracker – Daily Spending Manager** application.

* **🧪 Testing Environment Setup:**
* ✅ Python 3.10+ installed
* ✅ Required libraries: tkinter, sqlite3 (both built-in)
* ✅ Code: expense\_tracker.py file executed using Python interpreter
* ✅ Database: expenses.db auto-generated on first run
* **🔁 Step-by-Step Procedure:**

1. **Launch the Application**
   * Run expense\_tracker.py using Python.
   * Confirm that the GUI window opens without any errors.
2. **Test Expense Entry (Valid)**
   * Enter values in Category, Amount, and Note fields.
   * Click Add Expense and confirm the success popup appears.
   * Check that fields are reset after submission.
3. **Test Input Validation**
   * Leave Category empty and try to submit.
   * Enter non-numeric value in Amount field.
   * Confirm that appropriate warning/error messages are shown.
4. **Test Viewing Expenses**
   * Click View Expenses button.
   * Check that the table loads all previously added data from the database.
5. **Check Database Functionality**
   * Open expenses.db in SQLite browser (optional).
   * Confirm that data entries are correctly stored and updated.
6. **Test Interface Elements**
   * Verify all buttons are clickable.
   * Ensure Treeview table and scrollbar function correctly.
   * Validate that application remains responsive during use.
7. **Test Edge Cases**
   * Try adding extremely large amount values.
   * Use special characters in note fields.
   * Reopen the application and verify data persists.

* **✅ Conclusion:**

The testing procedure confirms that the application:

* Works smoothly without crashes
* Properly handles valid and invalid inputs
* Correctly stores and retrieves data from the SQLite database
* Maintains consistent performance across multiple entries

## Performance Outcome

he performance testing of the **Expense Tracker – Daily Spending Manager** application was conducted to evaluate the speed, stability, and responsiveness of the system under normal usage conditions.

The outcome of the tests demonstrated that the application is:

1. **✅ Fast, Stable, and Efficient**

* **🔹 Key Outcomes:**

| **Metric** | **Result** |
| --- | --- |
| **Application Launch Time** | **Less than 1 second** |
| **Expense Entry Response Time** | **Immediate (no delay)** |
| **View Expenses Load Time** | **Instantaneous (even with 50+ entries)** |
| **Memory Usage** | **Low (~30–50 MB RAM usage)** |
| **CPU Usage** | **Minimal (~2–5% usage during operation)** |
| **UI Responsiveness** | **No lags, fully interactive** |
| **Database Performance** | **Fast inserts & retrievals using SQLite** |

* **📌 Observations:**
* **The application maintains real-time response without performance drop, even after multiple entries.**
* **The SQLite database performs insert and select operations efficiently, suitable for local and small-scale use.**
* **The GUI remains stable and smooth on low-end systems with basic hardware configuration.**
* **🟢 Conclusion:**

**The Expense Tracker application shows excellent performance outcomes in all tested areas. It is well-optimized for personal finance tracking on desktop systems, making it ideal for students, freelancers, and individual users.**

# My Learnings

This internship and project experience gave me a deeper understanding of how real-world applications are developed using programming concepts I’ve learned in college. By working on the **Expense Tracker** using Python, I was able to connect theory with practice and enhance my confidence as a future developer.

* **✅ Technical Learnings:**
* **Python Fundamentals Refreshed**
  + Improved my understanding of variables, functions, loops, and conditionals.
* **GUI Development with Tkinter**
  + Learned how to create labels, entry fields, buttons, frames, and Treeview tables using Tkinter.
* **Database Handling with SQLite**
  + Understood how to connect Python with SQLite to store, retrieve, and update data securely.
* **Input Validation & Error Handling**
  + Gained knowledge in handling user inputs and displaying appropriate warning or error messages.
* **Modular Programming**
  + Practiced writing clean, reusable functions and maintaining a logical code structure.
* **Real-time Testing and Debugging**
  + Learned how to test GUI-based applications and fix bugs during runtime effectively.
* **Performance Optimization**
  + Observed how code structure affects speed, UI responsiveness, and memory usage.
* **💬 Final Note:**

This internship helped me realize that I’m capable of building useful, working applications, and has motivated me to continue learning and improving as a future developer.

# Future Work Scope

While the current version of the **Expense Tracker** application performs all basic operations effectively, there is plenty of potential to enhance its functionality and usability. These improvements can help scale the project for more advanced personal finance management.

1. **🔮 Possible Enhancements in Future:**
2. **Export to CSV or Excel**
   * Allow users to export their expense history for reports and analysis.
3. **Monthly and Category-wise Filters**
   * Add dropdowns or date pickers to filter expenses by month, week, or category.
4. **Analytics and Charts**
   * Integrate with matplotlib or plotly to visualize spending trends using bar charts and pie charts.
5. **Login System (Multi-user Support)**
   * Introduce user login and authentication to support multiple users securely.
6. **Dark Mode / Theming**
   * Add options to switch between light and dark themes for better user experience.
7. **Delete / Edit Expense Entries**
   * Allow users to modify or delete existing expense records.
8. **Cloud Syncing / Online Version**
   * Upgrade the app to support cloud storage and backup using Firebase or Google Sheets.
9. **Mobile App Version**
   * Develop a mobile version using Kivy or Flutter for cross-platform accessibility.

These enhancements will turn the current application into a more powerful and user-friendly tool that can cater to a wider audience and more diverse financial needs.

**Conclusion:**

The **Expense Tracker – Daily Spending Manager** project was successfully designed and developed as part of my internship to address the need for a simple and effective personal finance tool. The application allows users to easily record, view, and manage their daily expenses using a clean and responsive GUI built with **Python** and **Tkinter**, with data stored securely in a local **SQLite database**.

Through this project, I was able to apply my theoretical knowledge in a real-world context and gained practical skills in programming, UI design, and database integration. The application performs efficiently on low-end systems, is easy to use, and serves its purpose for individuals who need a lightweight, offline expense tracking solution.

This internship experience also helped me improve my technical confidence, problem-solving ability, and understanding of software development from idea to execution. I believe that this project has laid a strong foundation for more advanced application development in the future.