Final Project Report – GiveOne

CIS 434 – Software Engineering

Fall 2025

Ahmed Shady

Project Repository

`https://github.com/AhmedShady84/CIS434-Project-or-SoftwareEngineeringProject`

1 Introduction

GiveOne is a simple, user-friendly donation application designed around the concept of micro-giving. Instead of large, one-time contributions, GiveOne encourages frequent, small donations—starting as little as $1—toward charitable cases with meaningful emotional impact.

The app supports a clean signup process, a digital wallet, adjustable donation amounts, progress tracking for each case, a monthly donation summary, and a complete donation history. It was developed as a desktop application using Python and Tkinter as part of the CIS 434 Software Engineering course. The project demonstrates the full software engineering lifecycle, including requirements collection, design, implementation, testing, and final delivery.

2 Requirements Summary

2.1 Functional Requirements

GiveOne supports the following core functions:

1. **User Signup**
    - First-time users register with first name, last name, email, and password.

- Passwords are hashed locally using SHA-256.
- Application initializes wallet balance, cases, and donation history.

2. Wallet Management

- Users can add funds through an input dialog.
- Wallet balance is displayed both in the header and in the Wallet tab.

3. Browse Donation Cases

- Users see predefined empathetic cases with:
  - Title, organization, category
  - Story/description
  - Goal amount vs. raised amount
  - Progress bar and percentage
  - Case status (Open/Funded)

4. Adjustable Donations

- Users may enter any donation amount or use quick buttons ($1, $5, $10).
- Donations update wallet balance, case progress, and status.
- Fully funded cases automatically lock.

5. Donation History

- Each donation is logged with:
  - Timestamp
  - Case name
  - Donation amount
  - Running wallet balance
- History is shown in a Tkinter Treeview table.

6. Monthly Donation Summary

- The header displays total donated amount for the current month.

7. Refresh and Reset Demo

- Refresh reloads data from JSON.
- Reset Demo clears all user data and returns to the signup page.

2.2 Non-Functional Requirements

- Minimalistic, clean, and accessible Tkinter UI
- Fast response times on typical hardware
- Persistent data storage in a structured JSON file
- Basic password hashing for local demo safety
- Standalone offline desktop operation
- Resilient error handling for invalid input or corrupted records

3 System Design

Although the implementation is contained in a single Python file (`giveone_app.py`), the program follows a clean, layered architecture.

3.1 Architecture Overview

GiveOne uses a three-layer structure:

1. **UI Layer**
   - Built using Tkinter and ttk widgets
   - Contains SignupFrame, LoginScreen, and main GiveOneApp interface
2. **Logic Layer**
   - `WalletService`: manages balance and funds validation
   - `CaseService`: manages donations, case progress, and funding state
   - Utility functions for formatting, password hashing, and timestamps
3. **Data Layer**
   - Persistent storage through `giveone_data.json`
   - Default dataset for first-time launches
   - Donation logs, case structures, and user profile fields stored in JSON

3.2 Data Structures

- **User**: name, email, password hash, creation date
- **Wallet**: integer balance in cents, last updated timestamp
- **Case**: ID, title, organization, goal, raised amount, category, status
- **Donation**: timestamp, case ID, amount, running balance

3.3 Key Components

- **SignupFrame** – collects user information and initializes system state
- **GiveOneApp** – main interface with header, sidebar navigation, and pages
- **Case Cards** – dynamic widgets showing case data and donation options
- **Wallet Tab** – fund management and balance display
- **History Tab** – Treeview display of all donation logs

4 Implementation Details

- Implemented in Python 3.x using only the standard library
- Tkinter with ttk styling for modern UI components
- JSON used for all persistent local storage
- Adjustable donation system built with input fields and quick buttons
- Donations instantly update UI and JSON file
- Password hashing using SHA-256

Core internal functions:

- `load_data()` and `save_data()` manage persistent state
- `WalletService.add_funds()` handles validation and balance updates
- `CaseService.donate()` updates case progress and logs donations
- `month_donated_cents()` calculates monthly totals
- `refresh_all()` updates all UI components from JSON

5 Testing Summary

5.1 Testing Approach

Testing followed manual and scenario-driven steps:

- Manual exploratory testing
- Validation of each tab and workflow
- Multiple reset and relaunch cycles

5.2 Successful Test Scenarios

- User signup creates correct initial dataset
- Wallet balance updates accurately after adding funds
- Donation updates both wallet and case progress
- Fully funded cases disable donation capability
- Donation history logs all entries accurately
- Monthly total updates instantly
- Refresh operation reloads data without errors
- Reset Demo clears data and returns to signup

5.3 Issues Addressed

- Invalid donation inputs now trigger warnings
- JSON corruption is handled by ignoring malformed rows
- UI consistently reflects the latest data state

6 Limitations

- Single-user mode only
- JSON file is not secure enough for real-world applications

- No real payment processing (simulation only)
- No server, cloud sync, or multi-device support
- Simple architecture, appropriate for educational use

7 Future Enhancements

Potential areas for extension:

- Multi-user authentication and secure login
- Real donation APIs (Stripe, PayPal)
- Donation reminders or push notifications
- Automatic recurring donations
- Organization dashboard for posting charitable cases
- Web or mobile app version
- Encrypted data storage

8 How to Run the Application

1. Install Python 3.x.
2. Clone the repository: `git clone https://github.com/AhmedShady84/CIS434-Project-or-Softwa`
3. Open a terminal in the project folder.
4. Run the application: `python giveone_app.py`
5. Complete the signup form and begin:
    - Add funds
    - Donate to cases
    - View donation history
    - Reset demo when needed

9 Repository Link

`https://github.com/AhmedShady84/CIS434-Project-or-SoftwareEngineeringProject`