Final Project Report – GiveOne

Course: CIS 434 – Software Engineering
Project: GiveOne – Adjustable Donation Application
Student: Ahmed Shady
Semester: Fall 2025
Repository: https://github.com/AhmedShady84/CIS434-Project-or-SoftwareEngineeringProject

---

1. Introduction

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

The application includes a clean signup process, a digital wallet, adjustable donations, progress-tracking for each case, a monthly donation summary, and a full donation history. The app was developed as a standalone desktop application for the CIS 434 Software Engineering course, demonstrating the full software engineering lifecycle: requirements collection, design, implementation, testing, and final delivery.

2. Requirements Summary

2.1 Functional Requirements

The GiveOne application supports the following core functions:

1. **User Signup**
    - First-time users register with name, email, and password (hashed locally).
    - Application initializes wallet balance, cases, and history.
2. **Wallet Management**
    - Add funds through an input dialog.
    - Balance displayed in both the header and Wallet tab.
3. **Browse Donation Cases**
    - Users see 5 predefined empathetic cases with:
        - Title, organization, category
        - Story/description
        - Goal amount vs. raised amount
        - Progress bar and funding percentage
        - Status (Open/Funded)

4. **Adjustable Donations**
    - User enters any amount (default $1.00), or uses quick buttons ($1, $5, $10).
    - Donation updates wallet, case progress, and funding status.
    - Fully funded cases are locked and show a confirmation message.
5. **Donation History**
    - Every donation is logged with:
        - Timestamp
        - Case name
        - Amount donated
        - Running wallet balance
    - Displayed in a sortable Treeview table.
6. **Monthly Donation Summary**
    - Header shows total amount donated in the current month.
7. **Refresh & Reset Demo**
    - **Refresh** reloads JSON data into the UI.
    - **Reset Demo** clears user/wallet/history/cases and returns to signup.

2.2 Non-Functional Requirements

- Minimalistic, clear, and accessible Tkinter UI
- Fast UI responses (near-instant on typical machines)
- Persistence through a structured JSON file
- Basic password hashing for local demo safety
- Offline standalone desktop app
- Resilient error handling for invalid inputs and corrupted records

3. System Design

Although implemented in a single file (`giveone_app.py`), the app follows a structured architecture.

3.1 Architecture Overview

The internal design follows a three-layer style:

1. **UI Layer**
    - Tkinter frames, buttons, labels, notebook tabs
    - SignupFrame, GiveOneApp, Cases/Wallet/History tabs
2. **Logic Layer**
    - `WalletService`
    - `CaseService`
    - Helper utilities (formatting, hashing, timestamping)
3. **Data Layer**
    - JSON read/write (`giveone_data.json`)

    - Default dataset for first-time launch
    - Donation logs and case updates

3.2 Data Structures

- **User:** name, email, password hash, created_at
- **Wallet:** integer balance in cents, last_updated timestamp
- **Case:** id, title, org, goal_cents, raised_cents, description, category, status
- **Donation:** timestamp, case_id, amount_cents, running_balance_cents

3.3 Key Components

- **SignupFrame:** collects user information
- **GiveOneApp:** main interface with header + tabs
- **Case Cards:** dynamic, scrollable components showing each case
- **Wallet Tab:** add funds + display balance
- **History Tab:** Treeview log of all donations

4. Implementation Details

- Written in **Python 3.x** using **only the standard library**
- UI built with Tkinter's ttk widgets
- Uses JSON file for persistent local storage
- Adjustable donations via input field and quick buttons
- Donations update both UI and data file instantly
- Password hashing via SHA-256 for demo safety
- Full app behavior contained in one file for easy grading and portability

Core functions:

- `load_data()` / `save_data()` maintain JSON state
- `WalletService.add_funds()` validates and updates balance
- `CaseService.donate()` updates raised amount, status, and logs donation
- `month_donated_cents()` computes monthly totals
- `refresh_all()` repaints all tabs from disk

5. Testing Summary

5.1 Testing Approach

Testing was done through:

- Manual scenario testing using the Test Plan
- Step-by-step flow validation
- Multiple reset + relaunch cycles

5.2 Successful Test Scenarios

- User signup initializes default data correctly
- Adding funds updates wallet + displays correct balance
- Donating reduces wallet and increases case progress
- Fully funded cases disable donation button
- History tab logs every donation accurately
- Monthly total updates correctly after each donation
- Refresh operation reloads data without errors
- Reset Demo wipes all data and returns to signup correctly

5.3 Issues Addressed

- Invalid donation inputs trigger helpful warnings
- JSON corruption handled by skipping malformed rows
- UI consistently reflects latest saved data

6. Limitations

- Single-user mode only
- Local JSON file is not secure for real-world use
- No real payment processing (simulation only)
- No server, cloud sync, or multi-device support
- Simple architecture, suitable for coursework but not large-scale deployment

7. Future Enhancements

Potential next steps:

- Real authentication system with multiple users
- Integration with real donation APIs (Stripe, PayPal)
- Push notifications or donation reminders
- Auto-donation schedules
- Organization-side dashboard to post/update cases
- Mobile app or web version
- Data encryption for secure storage

---

8. How to Run the Application

1. Install Python 3.x.
2. Download or clone the repository:
3. `git clone https://github.com/AhmedShady84/CIS434-Project-or-SoftwareEngineeringProject`
4. Open a terminal in the project's folder.
5. Run the app:
6. `python giveone_app.py`
7. Complete the signup form and begin using the app:
    - Add funds
    - Donate
    - View history
    - Reset demo as needed

9. Repository Link

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