SOFTWARE REQUIREMENTS SPECIFICATION

GiveOne

Ahmed Shady

1.0 Introduction

1.1 Goals and objectives

GiveOne enables donors to contribute in simple $1 increments to real cases posted by verified organizations, track funding progress end-to-end, and receive final outcome updates when cases are completed.

- Minimize friction to donate ($1, one-tap flow).
- Provide transparent progress and status (Open → Funded → Completed).
- Persist data across sessions and surface donor history and milestones.

1.2 Statement of scope

- **Inputs:** User donations ($1 units), organization case data (goal, description, status, updates).

- **Processing:** Wallet balance changes, case progress calculation, state transitions, history logging.

- **Outputs:** Progress bars, donation receipts, history, status badges, final updates.

1.3 Software context

Standalone desktop app (Python + Tkinter). Data persisted locally in JSON/SQLite. Future versions may add reminders, dashboards, and auto-donate.

1.4 Major constraints

- Single developer, limited resources.

- Must be user-friendly for non-technical users.

- Tech: Python/Tkinter UI, JSON/SQLite storage, fast response (<250ms).

2.0 Usage scenario

2.1 User profiles

- **Donor:** Main user who donates $1, tracks cases, and views history.
- **Organization:** Provides case data and completion updates (seeded initially).

2.2 Use-cases

- **UC-1 Browse Cases:** Donor views list of cases and details.
- **UC-2 Add Funds to Wallet:** Donor adds money to balance.
- **UC-3 Donate $1 to Case:** Donor donates, progress updates, case may close.
- **UC-4 View Donation History:** Donor sees all transactions.
- **UC-5 Receive Final Update:** Case is completed, donor notified.
- **UC-6 (Future): Auto-Donate/Reminders.**

2.3 Special usage considerations

- Simple $1 donations with minimal steps.
- Visual progress and celebration when funded or completed.

3.0 Data Model and Description

3.1 Data Description

3.1.1 Data objects

- **User:** `user_id (PK)`, name, email, created_at
- **Wallet:** `wallet_id (PK)`, `user_id (FK)`, balance, last_updated
- **Case:** `case_id (PK)`, title, description, goal_amount, raised_amount, status, org_name, created_at, completed_at

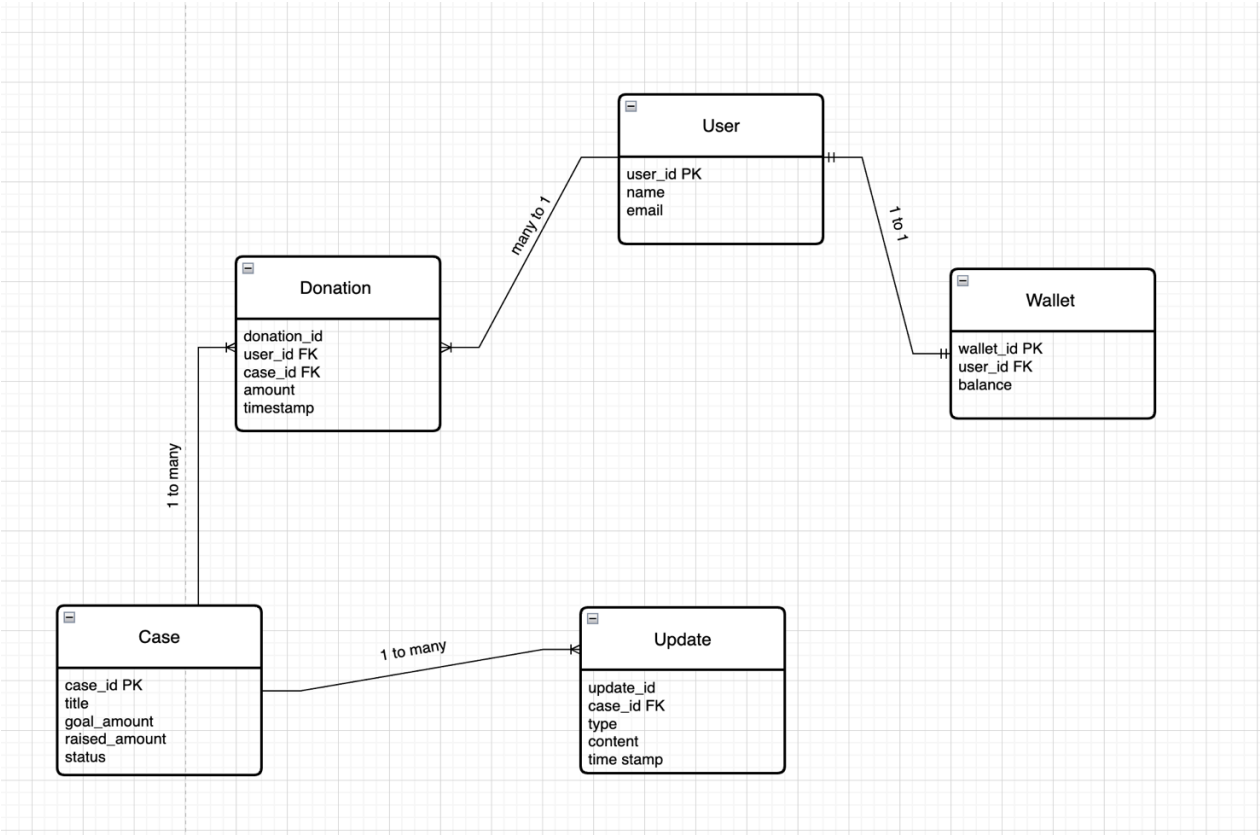
- **Donation:** `donation_id (PK)`, `user_id (FK)`, `case_id (FK)`, amount, timestamp, running_balance

- **Update:** `update_id (PK)`, `case_id (FK)`, type {Progress, Final}, content, timestamp

3.1.2 Relationships

- User 1—1 Wallet
- User 1—* Donation
- Case 1—* Donation
- Case 1—* Update

3.1.3 Complete data model



3.1.4 Data dictionary

Field	Type	Description	Constraints
user_id	Integer PK	Unique ID for user	Required
wallet_id	Integer PK	Unique wallet ID	Required
case_id	Integer PK	Unique case ID	Required
donation_id	Integer PK	Unique donation ID	Required
update_id	Integer PK	Unique update ID	Required
balance	Decimal	Wallet balance in $	≥ 0
goal_amount	Decimal	Case funding goal	> 0
raised_amount	Decimal	Amount raised so far	≥ 0
status	Enum	{Open, Funded, Completed}	Required
amount	Decimal	Donation amount	= $1 in MVP
timestamp	DateTime	Time of event	Required

4.0 Functional Model and Description

A description of each major software function and software interface is presented.

4.1 Description of Major Functions

4.1.1 Requirement 1 — Browse Cases

The system shall display all cases with title, goal, raised, progress bar, and status.

4.1.2 Requirement 2 — Wallet Management

The system shall allow the donor to add funds, validate input, and persist balance.

4.1.3 Requirement 3 — Donate $1

The system shall allow one-tap $1 donations, update wallet and case, and mark cases Funded if the goal is reached.

4.1.4 Requirement 4 — Donation History

The system shall display a chronological donation table (date, case, amount, running balance).

4.1.5 Requirement 5 — Case Updates

The system shall display progress/final updates and transition a case to Completed when final update is posted.

4.1.6 Requirement 6 — Persistence & Performance

The system shall store all data locally (JSON/SQLite), respond to UI actions in <250ms, and restore state on restart.

4.1.7 Requirement 7 — Reminders/Auto-Donate (Future)

The system should allow reminders and auto-donate options in future versions.

4.2 Software Interface Description

4.2.1 External Machine Interfaces

None in MVP (runs on a single computer).

4.2.2 External System Interfaces

None in MVP. Future versions may connect to payment gateways or notification services.

4.2.3 Human Interface

Tkinter GUI with:

- **Home/Cases:** card list, progress bar, $1 donate button, status labels.
- **Case Detail:** description, updates, progress.
- **Wallet:** balance, add funds.
- **History:** list of past donations.

5.0 Restrictions, Limitations, and Constraints

- Future features (reminders, dashboards) deferred.
- MVP assumes pre-seeded cases only.
- No sync or cloud support in MVP.
- Solo project schedule:
    - SRS due Sept 30
    - Design due Oct 14
    - Demo Oct 28
    - Test Plan Nov 4
    - Final Presentation Dec 2–4