Developer Guide

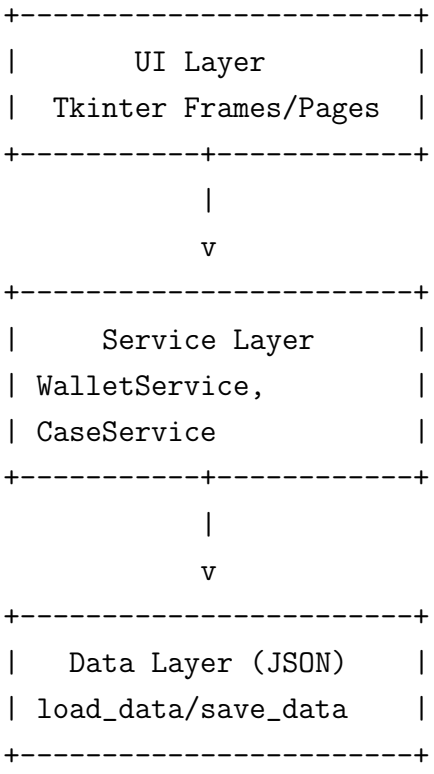
GiveOne – Adjustable Donation Application

CIS 434 – Software Engineering

Fall 2025

Ahmed Shady

1 Architecture Overview



The application follows a structured three-tier architecture:

- **UI Layer** – Tkinter frames and pages.
- **Service Layer** – application logic (wallet, cases, donations).
- **Data Layer** – JSON persistence system.

2 Technologies Used

Language

Python 3

Framework

- Tkinter
- ttk (modern UI styling)

Libraries

- `json`
- `os`
- `datetime`
- `hashlib` (SHA-256 hashing)
- `tkinter / ttk / messagebox / simpledialog`
- `copy.deepcopy`

Storage

- Local JSON file `giveone_data.json`
- No external or online dependencies

3 Key Modules and Classes

3.1 WalletService

Handles wallet operations:

- `add_funds(dollars)`
- `deduct(cents)`
- `can_donate(cents)`

3.2 CaseService

Core donation case logic:

- `list_cases()`
- `find_case(case_id)`
- `donate(case_id, cents)`
- `get_next_open_case_id()`

3.3 Auto-Pay System

Stored in JSON:

```
"autopay": {
  "enabled": true,
  "amount_cents": 100,
```

```
  "case_id": 301,
  "last_run_ts": "YYYY-MM-DD HH:MM:SS"
}
```

Function: `check_and_run_autopay(data)`

4 Algorithms

4.1 Daily Streak Algorithm

```
if user has no previous donation:
    streak = 1
else:
    if same day:
        streak unchanged
    elif time difference <= 24 hours:
        streak += 1
    else:
        streak = 1
```

Related function: `break_streak_if_inactive()` resets streak if over 24 hours since last donation.

4.2 Auto-Pay Algorithm

```
if autopay.enabled:
    if amount valid:
        if 24 hours passed:
            if wallet has enough:
                donate()
                update last_run_ts
                if case funded:
                    switch to next open case
```

Prevents double-charging and ensures time accuracy.

4.3 Case Filtering Algorithm

```
for each case:
    if category != "All" and category mismatch:
         skip
    if search not in (title + org + description).lower()
       :
         skip
    display case
```

4.4 Wallet Operations

- Stored in cents to avoid floating-point errors
- Donation fails if `can_donate()` returns false
- Balance updated synchronously

4.5 Data Persistence Algorithm

Ensures safe loading and backward compatibility.

- `load_data()`
- `save_data()`
- `ensure_data_shape()`

5 UI Flowchart

```
StartScreen
  SignupScreen
```

```
LoginScreen
    GiveOneApp
        CasesPage
        WalletPage
        HistoryPage
        FriendsPage
        SettingsPage
```

6 Data Schema (JSON Format)

```
{
  "user": {
    "first_name": "Ahmed",
    "last_name": "Shady",
    "username": "shady_ahmed",
    "email": "ahmed@example.com",
    "password_hash": "...",
    "streak_days": 3,
    "streak_last_ts": "2025-01-05 14:20:10",
    "invite_code": "GV1-ABC123"
  },
  "wallet": {
    "balance_cents": 500,
    "last_updated": "2025-01-05 14:20:10"
  },
  "cases": [...],
  "donations": [...],
  "autopay": {...},
  "friends": [...],
  "settings": {"theme": "light"},
  "payment": {"preferred_bank": "Wallet only (demo)"}
}
```

7 Adding New Features

7.1 Adding a New Case Category

1. Add a new object in `DEFAULT_CASES`
2. Include image if needed
3. Category dropdown updates automatically

7.2 Adding New Payment Providers

Modify:

- Wallet page combobox
- `_update_payment_from_ui()`

7.3 Converting to Cloud Backend

Replace JSON with REST API calls:

- `GET` for loading
- `POST` for saving

8 Error Handling

- Invalid donation amounts → alert popup
- Insufficient funds → donation blocked
- Corrupted JSON → missing fields auto-corrected
- Auto-pay prevents duplicate charges

9 Known Limitations

- JSON storage not encrypted
- No real authentication system
- Auto-pay runs only while app is open
- Tkinter not optimized for mobile layouts
- Single-user mode only

10 Future Enhancements

- SQL or Firebase backend
- OAuth login
- Real payments (Stripe, PayPal, Plaid)
- Multi-device syncing
- Mobile app version (Flutter / React Native)
- Admin dashboard for managing cases