



# Software Requirements Specification

**commit.**

PDG - HEIG-VD

Leonard Cseres

Tristan Gerber

Aladin Iseni

David Schildböck

August 22, 2025

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Project Description</b>	<b>3</b>
<b>3</b>	<b>System Overview</b>	<b>3</b>
<b>4</b>	<b>Functional Requirements</b>	<b>4</b>
4.1	Roles . . . . .	4
4.2	User Management . . . . .	4
4.2.1	Authentication . . . . .	4
4.2.2	Payments & Account Receivers (MVP decisions) . . . . .	4
4.3	Core Features . . . . .	5
4.4	Goal Creation . . . . .	5
4.5	Group Challenges . . . . .	6
4.6	Settings . . . . .	6
4.7	Optional Functions (Future Enhancements) . . . . .	6
<b>5</b>	<b>Non-Functional Requirements</b>	<b>6</b>
5.1	Security . . . . .	6
5.2	Privacy . . . . .	6
5.3	Reliability & Operations notes . . . . .	7
5.4	Usability . . . . .	7
5.5	Reliability & Availability . . . . .	7
5.6	Compatibility . . . . .	7
5.7	Offline Behavior . . . . .	7
<b>6</b>	<b>Preliminary Architecture Description</b>	<b>7</b>
<b>7</b>	<b>Mockups / Landing Page</b>	<b>7</b>
<b>8</b>	<b>Technical Choices</b>	<b>7</b>
<b>9</b>	<b>Work Process</b>	<b>8</b>
9.1	Agile Methodology: SCRUM . . . . .	8
9.2	Git Flow . . . . .	8
9.3	Devops . . . . .	8
<b>10</b>	<b>Development Tools Setup</b>	<b>8</b>
<b>11</b>	<b>Deployment Environment</b>	<b>8</b>
<b>12</b>	<b>CI/CD Pipeline</b>	<b>9</b>
<b>13</b>	<b>Constraints &amp; Assumptions</b>	<b>9</b>

<b>A Appendix</b>	<b>10</b>
A.1 Glossary . . . . .	10

## 1 Introduction

- **Purpose:**
  - Primary users: Private adult individuals who want to take action in their lives and seek stronger follow-through on commitments.
  - Core problem: Providing extra motivation to get things done by introducing a financial stake and social accountability.
  - Top objectives:
    1. Increase goal adherence by letting users stake money
    2. Enable social accountability via groups and shared challenges with pooled stakes.
    3. Provide automated financial consequences when goals are missed.
  - Success metrics:
    - \* Successfully transfer money from one user to another
    - \* Cross platform mobile application (IOS and Android)
- **Scope:**
  - Platforms: iOS and Android via Expo; web landing page (marketing/information only).
  - Geography at launch: Switzerland.
  - User types: Standard users, reviewer and admin.
  - MVP goal types: Wake-up time, location-based, time-based, duration-based, and combinations of location/time/duration.
  - Money flow: Real money via Stripe (payments and transfers).
  - Group challenges: Invite-only, private groups.
  - Out of scope (MVP): App Store/Play Store deployment, tablet support, advanced analytics, notifications (not planned), appeals/dispute handling.
  - Languages: English only at launch.
  - Accessibility & compliance: Out of scope for MVP.

## 2 Project Description

- **Objective:**
  - Product vision: Commit turns intentions into actions by combining simple goal tracking with real financial stakes and social accountability.
  - Primary use cases:
    1. Daily wake-up challenge at a set time
    2. Location-based activity (arrive and stay at a gym/park)
    3. Focused session for a specified duration (no-phone-use)
  - Differentiators: Ease of use, pooled stakes for group challenges.
  - Constraints/guiding principles: Instant money transfers between parties with minimal fees.

## 3 System Overview

- High-level description of the app

- Solo flow: A user creates a goal, due date or recurrence, selects a verification method (GPS/time/duration/photo), sets a stake amount and a destination for funds. If the goal is completed and verified within the rules, no transfer occurs; if not, the staked amount is transferred to the configured destination.
- Group flow: A user creates a private, invite-only group challenge with a defined goal, due date or recurrence, selects a verification method (GPS/time/duration/photo) and stake amount (and a fallback destination if all participants fail). Invitees accept the stake. Upon completion, successful participants receive the pooled stakes from members who failed; if all fail, funds are sent to the fallback destination.
- Mobile: iOS and Android (phones only; no tablet support. Expo + React Native)
- Permissions/capabilities: Background location (GPS), camera, device usage detection (for no-phone-use goals)
- Web: Single static marketing landing page (Astro)
- Target audience
  - Adults 18+ seeking productivity, fitness improvements, and habit-building.

## 4 Functional Requirements

### 4.1 Roles

- **User:**
  - Standard user role
- **Reviewer:**
  - Has all the capabilities of a standard user.
  - Can access and review user-submitted pictures for goal verification.
  - Can approve or reject verification evidence as part of the manual review process.
- **Admin:**
  - Role exists in the system but currently has no special functions defined for MVP.
  - Future admin capabilities will be specified as the project evolves.

### 4.2 User Management

- Users can register
- Users can manage their profile

#### 4.2.1 Authentication

- Sign-in methods: Google, Apple (no email/password at MVP)
- Pre-stake verification: No email/phone verification required before staking
- KYC/identity checks: Not required at MVP for payouts via Stripe

#### 4.2.2 Payments & Account Receivers (MVP decisions)

- Charging model: Stakes are defined at creation but only deducted if the goal is not achieved (on failure). If the goal is achieved, no funds are captured.

- Stake range: CHF 1 (min) to CHF 1000 (max) per goal.
- Currency: CHF only at launch.
- Recipients:
  - Solo challenge: at creation the user selects a recipient among: (a) a named person who is an existing app user, (b: if possible within MVP timeline) a charity from a small predefined list, or (c) the developers (platform donation account).
  - Group challenge: on resolution, winners split the stakes evenly among all winners. If no participants succeed, the pooled funds go to the destination configured by the creator for the group goal.
- Platform fees: No operational commission will be taken on stake transfers for the MVP; transfers to the developers' account are treated as donations. Stripe processing fees are available [here](#).
- Payout timing: instant payouts to winners are preferred; this requires connected payout accounts for recipients or platform-managed routing via Stripe. Stripe Connect patterns will be used appropriately in the implementation.

### 4.3 Core Features

- Goal lifecycle: create, edit, delete, view history
- Group lifecycle: create private group, invite by link, join/leave, view results
- Verification capture: GPS check, time check, in-app photo capture within window
- Money: create stake authorization, capture on failure, distribute to winners or fallback destination
- Activity/history: per-user list of past goals/challenges with outcomes
- Settings: change display name and profile photo

### 4.4 Goal Creation

- Required fields: name, goal type, start date, and due date or recurrence.
- Recurrence: select days of the week with an end date.
- Verification window: allowed;  $N$  minutes around the scheduled time (configurable per goal).
- Location goals: geofence with default and maximum radius (meters) and a must-stay duration.
- Duration/focus goals: strictly continuous session; minimum and maximum duration values.
- Photo verification: photo must be captured within the verification window; front or back camera allowed; selfie not required.
- Failure definition: missing verification or verification outside the allowed window results in automatic failure.
- Grace/retries: none for MVP.

## 4.5 Group Challenges

- Size limit: up to 100 participants per group challenge.
- Stake uniformity: same stake amount for all participants.
- Join flow: invite via link with expiration; joiners must register/sign in and have a valid payment method on file.
- Invite expiration: default 24 hours.
- Schedule: group goals follow the creator's schedule. The creator may set a time interval window to allow flexibility for participants to perform within their availability.
- Distribution: on resolution, winners split the pooled stakes evenly. If no participants succeed, funds go to the destination selected by the creator for this group goal.
- Failure to verify: not providing required verification within the window is an automatic failure.
- Cancellation: if the creator cancels before the start, no stakes are captured (since capture happens only on failure at resolution).

## 4.6 Settings

- Update display name

## 4.7 Optional Functions (Future Enhancements)

- **Community Challenges:** Public or open group challenges where any user can join and compete, with shared stakes and leaderboards.
- **AI Image Check:** Automated verification of user-submitted photos using AI to reduce manual reviewer workload and improve scalability.
- **Additional Payment Methods:** Support for more payment options beyond TWINT, such as PayPal or credit cards.
- **Charity Donations:** Users can choose a charity to donate to when selecting the destination for a goal.

# 5 Non-Functional Requirements

## 5.1 Security

- Data encryption in transit and at rest
- Secure authentication (OAuth2, JWT, etc.)
- Photo storage: stored as objects in Supabase Storage (S3-compatible). Retention policy: Out of scope for MVP.
- Location data: not stored server-side; processed on-device for verification where possible.

## 5.2 Privacy

- Photos are stored for verification purposes only; access is restricted to the account owner and authorized reviewers.

- Location traces are not persisted server-side; only ephemeral checks are performed for verification.
- Compliance posture: out of scope for MVP.

### 5.3 Reliability & Operations notes

- Manual verification SLA: initial target is to perform manual photo. The team will adjust this SLA based on capacity.

### 5.4 Usability

- Intuitive navigation
- Consistent UI across platforms

### 5.5 Reliability & Availability

- Graceful error handling

### 5.6 Compatibility

- Responsive design for different iPhone/android screen sizes

### 5.7 Offline Behavior

- Online-only MVP: goal creation and verification require connectivity

## 6 Preliminary Architecture Description

- Presentation layer: React native
- Application layer: React native + Supabase
- Data layer: Supabase + PostgreSQL
- Infrastructure: Supabase hosting

## 7 Mockups / Landing Page

- Figma designs
- Paper/whiteboard sketches

## 8 Technical Choices

- Programming languages & frameworks: Expo + React Native (TypeScript)
- Database: Postgres (Supabase)
- Backend/services: Supabase (Auth, DB, storage, edge functions)
- Payments: Stripe Connect Standard with TWINT enabled for Switzerland
- Third-party libraries & APIs: Stripe SDK, Expo Location/Camera
- Hosting: Supabase (backend, DB, auth); Cloudflare Workers (Astro landing page)



## 9 Work Process

### 9.1 Agile Methodology: SCRUM

- **Team roles:** Roles (Product Owner, Scrum Master, Developers) are rotating among team members throughout the project.
- **Sprint length:** Each sprint lasts 3 days, reflecting the short 3-week project timeline.
- **Ceremonies:** The team holds a daily standup and a sprint review at the end of each sprint. Other SCRUM ceremonies (planning, retrospective) are adapted or combined as needed.
- **Backlog management:** All tasks and user stories are tracked as GitHub Issues, organized and prioritized in a GitHub Project Kanban board.
- **Definition of Done:** A task is considered done when the code is merged, all tests pass, and the feature works as intended.
- **Sprint goal:** Each sprint has a defined goal or deliverable to be implemented.
- **Estimation:** Tasks are estimated in terms of expected time to complete.
- **Review and acceptance:** The team collectively reviews and accepts completed work at the end of each sprint.
- **Adaptations:** Roles may be combined and ceremonies adapted based on project advancement and team needs, in line with the university context.
- **Process:** lightweight Kanban for MVP

### 9.2 Git Flow

- The `main` branch contains production code.
- Every feature/fix/task is discussed via an issue and a branch is created with the issue.
- All changes are integrated via pull requests with code review and CI checks

### 9.3 Devops

- Continuous delivery to production when changes pass CI and review

## 10 Development Tools Setup

- Issue tracker: GitHub Issues
- Code review: GitHub pull requests
- Documentation: repository README and SRS in `docs/`
- Code style: Prettier + ESLint with TypeScript rules
- Testing approach: jest
- Secrets/config: Supabase and Stripe keys via env files with secure storage

## 11 Deployment Environment

- Backend and data: Supabase project (Auth, Postgres, Storage, Edge Functions).
- Web landing page: Cloudflare Workers (static hosting with global CDN).

- Mobile app: Expo EAS builds with two channels: development and production. App Store/Play Store distribution is out of scope for MVP; internal distribution only (TestFlight/Android internal testing).
- Environments: development and production only, with separate Supabase projects and isolated resources.
- Secrets/configuration: managed via Supabase project settings, and EAS secrets. Stripe runs in Test mode for development and Live mode for production.

## 12 CI/CD Pipeline

- CI on PRs: lint and tests
- CD on `main`: build Expo APK and deploy static website

## 13 Constraints & Assumptions

- Budget & time constraints: 70K CHF budget; 3-week MVP timeline
- Regulatory compliance: out of scope for MVP

## A Appendix

### A.1 Glossary

<b>Stake</b>	The amount of money a user commits that may be captured if the goal is not achieved.
<b>Capture</b>	Charging the authorized stake when a goal is marked as failed.
<b>Goal window</b>	The scheduled time period during which the user must complete and verify the goal.
<b>Verification window</b>	The allowed buffer around the scheduled time for submitting verification.
<b>Geofence</b>	A virtual radius around a location used to verify presence.
<b>Dwell time</b>	The minimum time a user must remain inside a geofence.
<b>Destination</b>	The recipient configured to receive funds when a goal fails (person, charity, or platform donation).
<b>Winner pool</b>	The set of participants in a group challenge who achieved the goal and split the captured stakes.
<b>Group challenge</b>	A private, invite-only challenge with a uniform stake and shared rules created by a user.