# Software Requirements Specification (SRS)

**Project:** Sample Collection Tracking System

**Prepared for:**

**Prepared by:** kushan dissanayake

**Date:** 26 September 2025

## Revision History

| Version | Date | Author | Changes |
| --- | --- | --- | --- |
| 0.1 | 2025-09-26 | Assistant | Initial draft SRS |

## Table of Contents

1. Introduction
   1. Purpose
   2. Scope
   3. Definitions, acronyms, abbreviations
   4. References
   5. Overview
2. Overall Description
   1. Product perspective
   2. Product functions (high-level)
   3. User classes and characteristics
   4. Operating environment
   5. Design and implementation constraints
   6. Assumptions and dependencies
3. System Features and Requirements
   1. Functional requirements (by feature)
   2. Use cases and user interaction
   3. External interfaces
4. Non-Functional Requirements
   1. Performance
   2. Security
   3. Privacy & Data protection
   4. Reliability, availability, maintainability
   5. Scalability
   6. Usability / Accessibility
   7. Localization
5. Data Model and Database Design
6. API and Integration Requirements
7. Client Applications: Screens & Flows
   1. Employee Mobile App (React Native + Express backend)
   2. Admin Web Dashboard (Vite + React)
8. Real-time Location Tracking & Map Behavior
9. Authentication, Authorization & Account Management
10. Admin Features & Workflows
11. Acceptance Criteria & Test Cases
12. Deployment, Hosting & DevOps
13. Maintenance & Support
14. Risks, Open Issues & Future Enhancements
15. Appendix

# 1. Introduction

## 1.1 Purpose

This SRS describes the requirements for a Sample Collection Tracking System for the organization’s collection department. The system enables assignment and tracking of employees (collection agents) who follow fixed daily routes across branches in Sri Lanka. The system will provide real-time location tracking, route guidance, branch data capture (sample counts), and an admin dashboard to manage branches, employees and monitor daily collections.

## 1.2 Scope

The system includes two main user-facing applications:

* **Employee Mobile App** (React Native) — used by collection staff to view their daily route, see their current position on a map, navigate to branches, log collected sample counts for each branch, and mark arrival/leave.
* **Admin Web Dashboard** (React, Vite) — used by administrators to manage branches, employees, assign fixed routes, view daily sample count reports (date-wise), and view a live map of any employee’s current location and navigation path.

Backend will be implemented using **Express.js** and **Firebase** services (Firestore as primary DB; Firebase Authentication for auth; Firebase Cloud Messaging optional for notifications). Real-time updates for location and navigation will use either Firestore real-time listeners or Firebase Realtime Database (design decision included below). Map functionality will use an appropriate maps SDK (Google Maps / Mapbox) for both mobile and web.

## 1.3 Definitions, Acronyms, Abbreviations

* SRS — Software Requirements Specification
* RN — React Native
* FCM — Firebase Cloud Messaging
* DB — Database
* API — Application Programming Interface
* UI — User Interface
* GPS — Global Positioning System
* POI — Point of Interest

## 1.4 References

* Firebase (Authentication, Firestore, Realtime Database, Cloud Functions)
* Google Maps Platform or Mapbox SDK for map and navigation

## 1.5 Overview

This document details functional and non-functional requirements, data model, API endpoints, screen flows, acceptance criteria, and testing items needed to implement the system.

# 2. Overall Description

## 2.1 Product perspective

This product is a new system to replace manual or spreadsheet based tracking. It integrates with mobile devices to receive GPS coordinates, with Firebase as backend, and provides web UI for administration.

## 2.2 Product functions (high-level)

* CRUD for branches and employees (admin)
* Assign routes (an ordered list of branches) to employees
* Employee authentication and onboarding via invite link
* Daily route display for employee (route is fixed every day)
* Real-time location tracking and display on map (employee → admin)
* In-app navigation highlighting the nearest next branch and route guidance
* Branch-level sample collection logging (arrive, enter sample count, leave, finish)
* Daily and date-range reports of sample counts per branch and per employee
* Notifications and basic alerts (late arrival, missed branch)

## 2.3 User classes and characteristics

* **Employee (Collector)**
  + Uses Android/iOS mobile app
  + Low-to-medium technical skill
  + Logs in via invite link, sees only their assigned route and profile
  + Works Monday–Sunday, same fixed route each day
* **Admin**
  + Uses the web dashboard
  + Manages branches, employees, routes, sees real-time maps and historical reports
  + Higher privileges for CRUD and assignments

## 2.4 Operating environment

* Mobile App: Android and iOS (React Native)
* Admin Web: modern browsers (Chrome, Edge, Firefox, Safari)
* Backend: Node.js (Express) hosting (e.g., Cloud Run, Heroku, or VPS)
* Database: Firebase (Firestore/Realtime Database)
* Maps: Google Maps or Mapbox

## 2.5 Design and implementation constraints

* Must use Firebase for database and auth as requested.
* Mobile app must be built with React Native; backend with Express.js; admin UI with Vite + React.
* Must follow privacy rules for sharing employee location within the organization.

## 2.6 Assumptions and dependencies

* Each employee has a mobile device with GPS and internet.
* Branches have geocoordinates in DB.
* Routes are pre-defined and static (same each day) but can be updated by admin.
* Google Maps Platform (or Mapbox) API keys will be available.

# 3. System Features and Requirements

Requirements are numbered FR (functional requirement) and NFR (non-functional requirement)

## 3.1 Authentication & Onboarding

**FR-1**: Admin can create an employee record and trigger an invite email that contains an invite link.

**FR-2**: Invite link opens landing flow where employee sets temporary password and signs in (or uses password provided in the email). After first login, employee must change password.

**FR-3**: Both employee and admin have Forgot Password and Reset Password flows integrated with Firebase Auth (email based).

**FR-4**: Sessions must be secured (JWT/Firebase tokens) and expire after configurable time.

## 3.2 Employee Mobile App - Home Screen

**FR-5**: After login employee sees top area with current date and time.

**FR-6**: Home screen contains a central area with a button to view route map and a footer with Home and Profile icons.

**FR-7**: Profile screen shows employee details and allows password change.

## 3.3 Route Map & Navigation (Employee)

**FR-8**: Route Map screen displays: - Employee current location (blue dot or marker) - All branch locations for the employee’s route with red icons - Map controls (zoom, re-center)

**FR-9**: Floating Start button at bottom-right. On first press: - Start tracking session - Determine the nearest branch from current GPS location (calculate great-circle distance / haversine) - Display navigation path (polyline) from current location to that branch

**FR-10**: Employee can tap any branch marker to open Branch Details screen.

**FR-11**: While tracking, app continuously updates current location every N seconds (configurable, default 10s) and pushes updates to Firebase so admin sees real-time location.

## 3.4 Branch Details and Sample Entry

**FR-12**: Branch Details screen displays branch image, name, address, contact number, and last collected counts for that branch (today and historical summary).

**FR-13**: Footer on Branch Details has Arrived and Leave buttons. When Arrived is pressed, app marks employee status as “arrived” at branch with timestamp & location (server-validated radius check optional).

**FR-14**: In the middle of screen employee enters sample count collected (numeric input). When employee presses Finished: - The entry (branch, employee id, count, arrived\_at, left\_at, timestamps) is recorded in Firestore under today’s collection record - The app automatically finds next branch in route and displays navigation to it

**FR-15**: Leave button marks departure timestamp and location; if Finished wasn’t pressed earlier, system prompts to confirm sample count before leaving (configurable enforcement).

## 3.5 Admin Dashboard - Branches

**FR-16**: Admin can view list of branches, their details, and edit/add/remove branches.

**FR-17**: Branch listing includes daily sample count with date filter (date picker). Admin can select a date to view counts for any branch.

## 3.6 Admin Dashboard - Employees & Routes

**FR-18**: Admin can view employee list with details, assigned route, and current status (online/offline/last seen).

**FR-19**: Admin can view an employee’s current location on a map in real-time; the map shows employee path for the current session (polylines) and current target branch.

**FR-20**: Admin can create, edit and assign routes to employees. A route is an ordered array of branch IDs.

**FR-21**: Admin can add new employee and send invite email with login credentials or temporary token link.

## 3.7 Reporting

**FR-22**: Admin dashboard exposes reports: - Daily totals per branch - Daily totals per employee - Date-range export (CSV) of collection logs

## 3.8 Notifications and Alerts

**FR-23**: Optional notifications (FCM or email) for: - Admin: alerts if an employee is off-route or missed branch - Employee: reminders for next branch or missed arrival/finish

## 3.9 Audit & Logs

**FR-24**: System maintains logs of actions (arrived, left, sample recorded, login, password reset) with timestamps.

# 4. Non-Functional Requirements

## 4.1 Performance

* NFR-1: Mobile app location updates are pushed to server at default frequency of 10 seconds; system must handle up to X concurrent employees (to be sized by operations — initial target 200 employees).
* NFR-2: Admin real-time view should reflect employee location with <5 second latency subject to network conditions.

## 4.2 Security

* NFR-3: Use Firebase Authentication for secure auth flows. Passwords are never stored in plain text.
* NFR-4: Transport layer security — TLS for all API calls.
* NFR-5: Role-based access control: Admin vs Employee roles enforced server-side.

## 4.3 Privacy & Data protection

* NFR-6: Employee location data is sensitive — access restricted to Admin users only.
* NFR-7: Retention policy: keep detailed GPS traces for 90 days by default; older data aggregated or purged per policy.

## 4.4 Reliability, Availability, Maintainability

* NFR-8: Backend uptime target 99.5% (SLA to be negotiated).
* NFR-9: System shall handle intermittent mobile connectivity — app caches location and syncs when online.

## 4.5 Scalability

* NFR-10: Design to support horizontal scaling (stateless Express servers and Firebase-managed DB).

## 4.6 Usability / Accessibility

* NFR-11: Mobile UI must be simple; large touch targets for Arrived/Leave/Finished actions.

## 4.7 Localization

* NFR-12: System must support Sinhala and English UI text (future: Tamil); date/time in local timezone (Asia/Colombo).

# 5. Data Model and Database Design (Firebase Firestore schema - proposed)

* **collections**:
  + employees (doc id = employeeId)
    - name, email, phone, role, assignedRouteId, status, photoUrl, createdAt
  + branches (doc id = branchId)
    - name, address, contactNumber, imageUrl, geo {lat, lng}, notes
  + routes (doc id = routeId)
    - name, description, orderedBranchIds: [branchId], assignedEmployeeId
  + sessions (doc id = sessionId)
    - employeeId, startAt, endAt, status, routeId
  + locations (subcollection under sessions or an independent collection)
    - sessionId, timestamp, lat, lng, accuracy
  + collections (samples)
    - date (YYYY-MM-DD), employeeId, branchId, arrivedAt, leftAt, count, notes, proofs (photos optional)
  + auditLogs
    - actorId, action, target, timestamp, meta

**Notes**: - For real-time location use-case, you may store only the latest location in employees/{id}/liveLocation (for admin real-time view) and session-specific traces in sessions/{sessionId}/locations for history. - Firestore supports real-time listeners for liveLocation documents.

# 6. API and Integration Requirements (Express Backend endpoints - examples)

**Auth** - POST /api/auth/invite — (admin) create invite, send email - POST /api/auth/register — complete registration via invite - POST /api/auth/login — returns Firebase token (or uses Firebase on client side) - POST /api/auth/forgot-password — send reset

**Employees** - GET /api/employees — admin list - GET /api/employees/:id — details - POST /api/employees — create - PUT /api/employees/:id — update

**Branches** - GET /api/branches — list - GET /api/branches/:id — details - POST /api/branches — create - PUT /api/branches/:id — update

**Routes** - GET /api/routes — list - POST /api/routes — create (ordered branch IDs) - PUT /api/routes/:id — update

**Sessions & Location** - POST /api/sessions/start — start session for employee; returns sessionId - POST /api/sessions/:sessionId/location — push location (lat,lng,timestamp) - POST /api/sessions/:sessionId/arrive — mark arrival - POST /api/sessions/:sessionId/leave — mark leaving - POST /api/collections — record sample count for branch

**Reports** - GET /api/reports/daily?date=YYYY-MM-DD — daily totals - GET /api/reports/range?from=&to= — exports

**Security**: All endpoints require Firebase token in Authorization header; server verifies token via Firebase Admin SDK.

# 7. Client Applications: Screens & Flows

## 7.1 Employee Mobile App (React Native)

**Main Screens**: - Login / Forgot Password / Reset Password - Home (date & time top, central button to open Route Map, footer with Home & Profile) - Route Map (current location marker, branch markers in red, Start floating button bottom-right) - Branch Details (image, address, contact, arrived & leave footer buttons, count input, Finished button) - Profile (change password, contact info)

**Home Screen layout**: - Top: current date (e.g. “Friday, 26 Sep 2025”) and time (live updating) - Middle: large button “View Route Map” - Footer: Home | Profile icons

**Route Map behaviour**: - On pressing Start: create session on server if not exists, determine nearest branch, compute route via Maps Directions API, display polyline. - Continuous location tracking: request location permissions, background tracking optional (careful about battery).

## 7.2 Admin Web Dashboard (Vite + React)

**Sidebar menu**: Dashboard, Branches, Employees, Routes, Reports, Settings

**Dashboard**: KPIs — total branches, total employees, today’s collected samples, map with live employee locations

**Branches page**: list, search, add/edit form (including lat/lng picker on a map)

**Employees page**: list, add/edit, send invite button, view assigned route

**Employee live view**: select employee -> open map modal with real-time position marker and path polyline for current session

**Reports**: date picker, export CSV, branch-wise charts

# 8. Real-time Location Tracking & Map Behavior

* Use Firestore liveLocation doc per employee for instant updates; Admin listens to these documents.
* Use sessions/{sessionId}/locations for storing time-series location points for history and path drawing.
* Frequency: default 10s; less frequent (30s) in poor network to reduce costs.
* Map SDK: use Google Maps Directions API to compute navigation path between points. Show ETA and distance if desired.
* Geofencing: optional server-side validation to confirm arrival when GPS within configurable radius (e.g., 50 meters) of branch coordinates.

# 9. Authentication, Authorization & Account Management

* Use Firebase Auth for email/password accounts.
* Admin role flagged in employees collection or a separate admins collection.
* Invite flow: admin triggers invite API which creates user in Firebase Auth (disabled until registration) and emails link with token.
* Password reset handled by Firebase Auth built-in flows.

# 10. Admin Features & Workflows

* **Add Branch**: Admin fills form (name, address, contact, upload image, pick lat/lng). Branch saved to Firestore.
* **Add Employee**: Admin provides name/email/phone and assigns a route. System sends invite email with link.
* **Assign Route**: Admin creates route by selecting ordered list of branches, then assigns to employee.
* **Monitor**: Admin opens live map to see an employee’s current location and session path.
* **Reporting**: Admin generates daily/periodic reports and exports CSV.

# 11. Acceptance Criteria & Test Cases

Provide sample acceptance tests per feature — high-level examples below.

**AC-Auth-01**: Given admin creates an employee and sends invite, when employee uses invite link, then they must be able to register and change password.

**AC-Map-01**: When employee presses Start and is within range, the nearest branch is selected and directions displayed.

**AC-Collection-01**: When employee completes Finished on branch details, the count is stored and visible in Admin reports for that date.

**AC-RealTime-01**: Admin dashboard should reflect employee location updates within 5 seconds under normal network conditions.

# 12. Deployment, Hosting & DevOps

* **Backend**: Host Express server with Node.js on a cloud provider (Cloud Run, Heroku, DigitalOcean). Use environment variables for API keys and Firebase Admin credentials.
* **Firebase**: Configure Firestore database, Firebase Auth, and Cloud Messaging if used.
* **Admin UI**: Deploy static site (Vite build) to Firebase Hosting, Netlify, Vercel, or S3+CloudFront.
* **Mobile App**: Build and publish to Play Store / App Store or distribute internally.
* **CI/CD**: GitHub Actions pipelines for test, build and deploy.
* **Monitoring**: Use Stackdriver/Cloud Monitoring or third-party monitoring.

# 13. Maintenance & Support

* Log retention and backups: daily export of Firestore or scheduled backups.
* Support: define contact for issue reporting, triage, and release cadence for bug fixes and enhancements.

# 14. Risks, Open Issues & Future Enhancements

**Risks**: - Battery drain when continuous GPS tracking is active. - Mobile network coverage variability affecting real-time updates. - Privacy concerns around continuous location tracking — need clear policy & consent.

**Open issues**: - Which maps provider (Google Maps vs Mapbox) to use (cost implications). - Whether to use Firestore or Realtime Database for lowest-latency location streaming.

**Future enhancements**: - Offline-first behavior with robust sync - Geofencing & automated arrival detection - Mobile attachment (photo evidence) for sample pickups - Optimized route planning (if routes become dynamic)

# 15. Appendix

## 15.1 Sample screen wireframes (textual)

**Employee home**: - [Top] Fri, 26 Sep 2025 | 09:34 - [Center] Big button: VIEW ROUTE MAP - [Footer] Home Profile

**Route Map**: - Map fills screen - Red icons for branches (tappable) - Blue dot for current location - Start circular floating button bottom-right

**Branch Details**: - Branch image - Name - Address - Contact - Numeric input: Sample Count - Footer buttons: Arrived Leave and Finished

## 15.2 Example Firestore document example

branches/BRANCH001:

{  
 name: "Colombo West Clinic",  
 address: "123 Main St, Colombo",  
 contact: "+94 11 2345678",  
 geo: { lat: 6.9271, lng: 79.8612 },  
 imageUrl: "https://...",  
}

collections/2025-09-26/recordId:

{  
 date: "2025-09-26",  
 employeeId: "EMP001",  
 branchId: "BRANCH001",  
 arrivedAt: "2025-09-26T08:35:10Z",  
 leftAt: "2025-09-26T08:42:20Z",  
 count: 12  
}

## Sign-off

This SRS is intended to be a working baseline. After review by stakeholders, changes will be recorded in the revision history above.

**Stakeholders:** -