Civil QC Application

Comprehensive Supervisor Documentation

Generated: 11/3/2025, 3:05:34 PM
Prepared for: Project Supervisor
Prepared by: Civil QC Development Team
Version: 1.0

Executive Summary

This document provides a concise, supervisor-focused overview of the Civil QC Application, covering architecture, technical stack, features, security posture, testing strategy, and the roadmap. The goal is to give stakeholders a clear understanding of the product scope, current status, and next steps.

Project Overview

The Civil QC Application is a comprehensive quality control management system for civil engineering projects. It supports project tracking, inspection data capture, test results recording, photo evidence, and report generation tailored for excavation and concrete quality control.

Technical Stack

Frontend: React + TypeScript UI: Tailwind CSS State Management: Zustand Backend: Firebase Firestore Authentication: Firebase Auth Hosting: Vite dev server / Firebase Hosting

Features

Core features include project management, user authentication, excavation and concrete QC workflows, real-time data entry, offline-capable Firestore sync, and PDF report generation for compliance and archival.

App Screens Functionality

Authentication: Login, Sign up, Password recovery, Session handling. Project Management: Dashboard (MyProjects), Project details, Create/Edit/Delete flows. Quality Control: Excavation Screen (measurements, compaction reports, photos), White Concrete Screen (mix & test results, curing). Common UI: Reusable Buttons, Inputs, Form validation, Loading indicators, Notifications.

Architecture

Component-based React architecture. Clear separation between UI components, domain modules (project, excavation), and library utilities (firebase config, store). Zustand provides a centralized store with Firestore-backed persistence for sync and optimistic updates.

Setup and Installation

Prerequisites: Node.js 16+, npm. Steps: clone repo, npm install, create .env with VITE_FIREBASE_* keys, npm run dev. Production: npm run build then firebase deploy.

Security Considerations

Authentication enforcement via Firebase Auth, Firestore Security Rules to restrict access, environment variables for secrets, input validation, and role-based access for supervisors vs field users. Regular security audits recommended.

Testing Procedures

Unit tests: Jest for core logic. E2E: Cypress for user flows. Integration: Firebase emulator for testing Firestore rules and auth. Acceptance: Manual test cases for supervisor sign-off.

Best Practices

Follow TypeScript strict mode, consistent code formatting (ESLint/Prettier), write unit tests for new features, use feature branches, perform code reviews, and document changes in changelog.

Troubleshooting Guide

Common issues and quick fixes: auth misconfiguration, Firestore rules failures, environment variable errors, and steps to collect logs and reproduce issues.

Future Roadmap

Planned: Mobile app, offline-first improvements, advanced reporting, integrations with other construction tools, and supervisor analytics dashboard.

Contributing Guidelines

Fork -> Feature branch -> Tests -> PR -> Review. Maintain clear commit messages and include tests.

Support Information

Technical Support: support@civilqc.com Bug Reports: bugs@civilqc.com Feature Requests: features@civilqc.com