# **RepairTrack - Project Documentation**

Version: 1.0 (As of August 28, 2025)

## **1. Project Overview**

RepairTrack is a lightweight, web-based SaaS (Software as a Service) application designed for small repair shops. It allows shopkeepers to manage customer repair jobs, track their status, and provide customers with a way to view the progress of their repairs.

* Frontend Technology: React.js
* Backend & Database: Supabase (PostgreSQL, Authentication, and Row-Level Security)
* Styling: Tailwind CSS
* Core Libraries: react-router-dom, react-hot-toast, qrcode

## **2. Core Features Implemented**

This section details the key functionalities built into the application so far.

## **2.1. User Authentication & Profile Management**

* Sign Up & Login: Shopkeepers can create an account and log in using an email and password. Authentication is handled securely by Supabase Auth.
* Session Management: The application automatically manages user sessions. Once a shopkeeper logs in, they remain logged in until they explicitly log out.
* Profile Creation: After the first login, users are prompted to create a Profile, where they must enter their Full Name, Phone Number, and Shop Name. This information is stored in a separate profiles table in the database.
* Profile Editing: The shopkeeper can edit their profile information at any time via a "Profile" button in the dashboard header, which opens a slide-over panel.

## **2.2. Job Management Dashboard**

This is the main workspace for the shopkeeper after logging in.

* Personalized Header: The dashboard header dynamically displays the shop\_name from the user's profile, making the interface feel personalized.
* Create New Jobs: A form allows the shopkeeper to add a new repair job with the following details:
  + Customer Name
  + Customer WhatsApp Number
  + Item Name (e.g., iPhone 13 Pro)
  + Problem Description (e.g., Screen cracked)
  + Price (Quoted repair cost)
* Job ID Generation: When a new job is created, a unique, human-readable Job Code is automatically generated (e.g., RT-2025-000001).
* Jobs Table: All created jobs are displayed in a clean, organized table that shows all key details.
* Status Color Chips: The job status is displayed with a colored badge for quick visual identification (e.g., "In Progress" is blue, "Completed" is green).
* Quick Filters: Users can filter the jobs list to quickly see:
  + My Open Jobs (All jobs that are not "Completed")
  + Awaiting Parts
  + Completed
* Status Updates: The status of any job can be changed directly from the table using a dropdown menu.

## **2.3. Customer-Facing Features**

* Printable Receipt: For each job, the shopkeeper can generate and print a customer receipt.
* QR Code Tracking: The receipt includes a QR code. When scanned by the customer, this code links to a future web page where they can track the status of their specific repair job (e.g., your-app-url.com/track/RT-2025-000001).

## **2.4. User Experience (UX) & Interface (UI)**

* Modern UI: The interface is styled with Tailwind CSS, providing a clean, responsive, and professional look.
* Toast Notifications: The application uses react-hot-toast to provide instant, non-intrusive feedback for actions like saving a profile, creating a job, or encountering an error.
* Protected Routes: The application architecture ensures that only authenticated users can access the dashboard. Unauthenticated users are automatically redirected to the login page.

## **3. Technical Architecture & Database Schema**

This section outlines the backend structure that powers the application's features.

## **3.1. Supabase Backend**

* Authentication: Utilizes Supabase's built-in authentication to manage users, passwords, and sessions.
* Database: A PostgreSQL database hosts two primary tables.
* Row-Level Security (RLS): This is the core of the application's security. RLS policies are enabled on all tables to ensure that a logged-in shopkeeper can only see and manage their own data (their profile and their jobs). This provides critical data isolation in a multi-tenant environment.

## **3.2. Database Tables**

1. profiles Table: Stores user-specific information that is not related to authentication.
   * id (uuid, Primary Key, Foreign Key to auth.users.id)
   * full\_name (text)
   * phone (text)
   * shop\_name (text)
2. jobs Table: Stores all the information for each repair job.
   * id (bigint, Primary Key)
   * user\_id (uuid, Foreign Key to auth.users.id, links the job to a shopkeeper)
   * job\_code (text, the unique human-readable ID)
   * customer\_name (text)
   * customer\_whatsapp (text)
   * item\_name (text)
   * problem (text)
   * price (numeric)
   * status (text, e.g., 'Received', 'Completed')

## **4. Potential Future Features & Improvements**

This list captures ideas for the next versions of the application.

* Customer Status Page: Build the public-facing page that the QR code links to.
* WhatsApp Notifications: Integrate with a service like Twilio to automatically send WhatsApp updates to customers when their job status changes.
* Inventory Management: Add a simple inventory system to track spare parts.
* Analytics: Create a basic analytics view to show metrics like jobs per month, average repair time, and revenue.
* Multi-user Shops: Allow a shop owner to invite staff members who can also manage jobs under the same shop profile.