# Mine Safety Reporter PWA

## 1. The Problem and the Solution

### 1.1 The Problem: A Fragmented and Reactive Safety System

The existing safety reporting system is a complex and inefficient mix of digital and manual processes that hinder proactive safety management. The core problems stem from:

* **Lack of Standardization:** Reporting formats vary widely between the 11 different mines. Some reports are submitted as PDFs, others as Word or Excel files, and some are even handwritten. This diversity forces the central safety department to manually re-enter or format data into a single, cohesive format, leading to delays and potential data entry errors.
* **Decentralized Data & Poor Retention:** Safety records are saved on individual mine PCs, creating a fragmented database. This poses a significant risk of data loss due to hard drive crashes, necessitating a reliance on physical copies. There is no centralized server for reliable data retention.
* **Ineffective Incident Follow-up:** While fatal incidents are followed up diligently, the status of other accidents is largely unknown after the initial enquiry report is submitted. The central safety department lacks visibility into whether suggested measures have been implemented or if the incident has been officially closed.
* **Absence of Real-time Insights & Proactive Safety:**
  + There is no central, real-time dashboard to monitor the safety status of all mines. This prevents immediate appraisal of performance and proactive management.
  + The collection of critical data points prescribed by DGMS—such as time of incident, hot spot areas, age and category of personnel, and nature of injury—is inconsistent. Without this structured data, it's impossible to identify trends that could prevent future accidents.
* **Poor Cross-Mine Awareness:** Lessons learned and corrective measures from an incident in one mine are not easily shared or made visible to other mines. This limits a company-wide culture of continuous safety improvement.
* **Barrier to Proactive Reporting:** Due to poor on-site PC and internet infrastructure, employees find it difficult to report proactive safety indicators like **near misses** and **high-potential incidents**. This results in a reactive, rather than a preemptive, safety culture.

### 1.2 The Solution: A Centralized and Intelligent Digital Safety Platform

The Mine Safety Reporter PWA is the definitive solution to these challenges. It transforms the safety reporting landscape by creating a centralized, real-time platform accessible to everyone.

* **Standardized Digital Reporting:** The app forces a consistent reporting format across all mines, ensuring every report captures the specific data points required by DGMS.
* **Centralized, Secure Data Storage:** All data is stored on a centralized server (Firebase Firestore), eliminating the risk of individual PC failures and ensuring reliable data retention.
* **Real-time Visibility & Accountability:** The central safety department gains real-time visibility into the status of all incidents, including whether follow-up actions have been taken and if reports are officially closed. An immutable audit log ensures every change is tracked with a timestamp and user name.
* **Proactive Trend Analysis:** By consistently capturing and centralizing data on DGMS-prescribed heads, the app enables real-time trend analysis. This allows management to identify and address "hotspots" and high-risk behaviors before they lead to serious incidents.
* **Instant Awareness & Knowledge Sharing:** An incident in one mine and its suggested corrective actions are immediately visible on a central dashboard. This fosters a company-wide learning culture, helping to prevent the same incident from occurring elsewhere.
* **Mobile-First Reporting:** By allowing on-site personnel to use their mobile phones to report incidents instantly, even with limited connectivity, the app significantly boosts the reporting of near misses and high-potential incidents, shifting the focus from a reactive to a proactive safety mindset.

## 2. Executive Summary

### 2.1 Overview

The Mine Safety Reporter is a comprehensive PWA built on a modern stack to provide a robust solution for mine safety management. The core technologies include:

* **React:** For a single-page application experience.
* **Vite:** A fast and efficient build tool.
* **Tailwind CSS:** A utility-first CSS framework for a consistent and professional design.
* **Firebase:** Provides a secure backend for authentication and a real-time NoSQL database (Firestore).

The application is designed to function as a Progressive Web App, offering features like offline access and a native-app feel, making it ideal for use in remote or low-connectivity environments.

### 2.2 Key Features

* **User-Friendly Reporting:** A multi-step form allows for quick submission of detailed incident reports.
* **Customizable Data:** Administrators can manage lists of mines, sections, and incident types via an AdminPanel.
* **Real-time Dashboards:** Visual dashboards provide real-time analytics and Key Performance Indicators (KPIs) to monitor safety trends.
* **Detailed Incident Log:** A central log displays all incidents, which can be filtered, sorted, and expanded to show full details, comments, and an audit trail.
* **Automated Metrics:** The app automatically calculates important safety metrics like Days Since Last Lost Mandays Injury and Lost Time Injury Frequency Rate (LTIFR).
* **User Preferences:** Users can switch between dark and light modes and choose their preferred mobile navigation style.

## 3. Technical Architecture

### 3.1 Technology Stack

* **Frontend Libraries:**
  + react, react-dom: For the user interface.
  + lucide-react: A library of lightweight, customizable icons.
  + recharts: For rendering responsive and interactive charts.
  + date-fns: For efficient date and time manipulation.
  + jspdf, html2canvas: Used together for generating PDF documents from HTML content.
* **Build & Deployment:**
  + **Vite:** The project's build tool.
  + **vite-plugin-pwa:** Manages the PWA manifest and service worker to enable offline capabilities and a native-app experience.
* **Styling:**
  + **Tailwind CSS:** Provides utility classes for a modern, responsive design.
  + **PostCSS & Autoprefixer:** Used to process CSS for browser compatibility.
* **Backend:**
  + **Firebase:** Provides authentication (firebase/auth) and a NoSQL database (firebase/firestore).

### 3.2 Folder Structure

* src/: The root of the application's source code.
  + components/: Contains all reusable UI components, such as IncidentCard.jsx, Sidebar.jsx, and FloatingFilterBar.jsx.
  + context/: Houses the global state management system, including AppContext, AuthContext, and ConfigContext.
  + pages/: Contains the top-level components for each application view, like HomePage.jsx and ExecutiveDashboardPage.jsx.
  + utils/: Stores helper modules like formatters.js and mockData.js that handle non-component logic.

### 3.3 State Management

The application uses a powerful, layered context architecture to manage state globally:

* **AuthContext.jsx:** The outermost context layer, responsible for all Firebase authentication logic. It tracks the currentUser and handles functions like login, logout, and password updates.
* **ConfigContext.jsx:** Fetches and manages global configuration data such as mines, sections, and incidentTypes from Firestore's config\_ collections. This ensures all parts of the app are using the same, centrally-controlled reference data.
* **AppContext.jsx:** The core data layer, which listens for real-time changes to the incidents and users collections in Firestore. It provides this data to all components, along with business logic functions for adding/updating incidents, submitting daily statuses, and toggling UI preferences. The app can also switch to a demoMode which uses locally generated mock data for demonstration purposes, preventing any changes from affecting the live database.

## 4. Application Features & Pages

### 4.1 HomePage (src/pages/HomePage.jsx)

This is the application's landing page, providing a quick overview of daily activities.

* **Daily Submission Status:** A tabbed interface categorizes mines based on their submission status for a selected date: "No Submission," "No Accident" reported, or "Accident" reported. Clicking on an "Accident" mine opens a modal showing the specific incidents for that day.
* **"No Accident" Reporting:** A simple form allows a user to quickly log a "No Accident" status for a selected mine, which updates the submission status dashboard.
* **Home Page Notice:** The page can display an administrative notice or announcement, which is configured via the AdminPanel.

### 4.2 Report Incident Page (src/pages/ReportIncidentPage.jsx)

A multi-step form for submitting a new incident report.

* **Guided Workflow:** The form guides the user through entering incident details, a list of involved persons (victims), and an enquiry report (optional).
* **Dynamic Fields:** The form fields are populated with active configuration data from the ConfigContext.
* **Involved Persons:** The form allows adding multiple "victims" with details like name, category (Regular/Contractual), and age, which are mandatory for certain incident types.
* **Photo Uploads:** Users can attach photos to the report (though the application currently only stores the filename for security reasons).
* **Preview & Submission:** A final step provides a preview of the complete report before it is submitted to the database. After submission, a unique Incident ID is provided, and a PDF version of the report can be downloaded.

### 4.3 Incident Log Page (src/pages/IncidentLogPage.jsx)

The central hub for reviewing and managing all incident data.

* **Filtering & Sorting:** A control bar at the top allows users to filter the list by status, mine, type, and date range. The list can also be sorted by various criteria like date or days lost.
* **IncidentCard Component:** Each incident is represented by a customizable card with a color-coded border based on the mine. The card displays a summary and can be expanded to show full details.
* **Interactive Updates:** The card features an UpdateModal where users can add comments with tags, update the daysLost count for injury incidents, and change the incident's status.
* **Audit History:** An expandable "History" section within the card displays a time-stamped log of all changes made to the incident report.

### 4.4 Executive Dashboard (src/pages/ExecutiveDashboardPage.jsx)

A powerful analytics page for management.

* **KPIs:** Displays key metrics like Total Incidents, LTIFR, and Near Miss Ratio, with trend comparisons against the previous period.
* **Visualizations:** The page features several interactive charts powered by recharts:
  + **Mine Performance:** A stacked bar chart showing the breakdown of incidents per mine.
  + **Individual Mine Analysis:** A pie chart for a deep dive into incident types for a selected mine.
  + **Monthly Trends:** A stacked bar chart visualizing how incident types have trended over time.
  + **Incident Hotspots:** A horizontal bar chart identifying sections with the most incidents.
* **Floating Filter Bar:** A sticky, responsive filter bar at the top allows users to dynamically update all charts and KPIs in real time.

### 4.5 Comparison Page (src/pages/ComparisonPage.jsx)

A specialized dashboard for performing side-by-side performance analysis.

* **Period Comparison:** Allows users to compare any two time periods (e.g., "This Quarter vs Last Quarter") using predefined presets or custom date ranges.
* **Comparative KPIs:** A series of cards display a metric's value for both periods and calculate the percentage change, highlighting improvement or decline.
* **Mine & Type Variance:** Provides detailed tables and charts that show how individual mines and incident types have performed between the two periods.

### 4.6 Admin Panel (src/pages/AdminPanel.jsx)

The administrative control hub for managing core application data.

* **Config Managers:** Separate, reusable components allow for adding, editing, and deactivating mines, sections, and incident types directly in Firestore.
* **Operational Data:** A form to manually input monthly hours worked for each mine, which is used in metrics calculations.
* **Home Page Notice:** A simple interface for creating a custom, active notice to display on the HomePage.
* **Demo Mode Controls:** Provides buttons to activate or deactivate demo mode, populating the application with mock data without writing to the live database.

## 5. Development & Maintenance

### 5.1 Getting Started

1. **Install Dependencies:** Run npm install to install all required packages from package.json.
2. **Run Development Server:** Use npm run dev to start the local development server.
3. **Build for Production:** Use npm run build to create a production-ready build of the application.

### 5.2 Firebase Integration & Security

* **Authentication & Access Control:**
  + **Custom Login:** The login page uses a custom flow where a user ID is mapped to an email address for authentication with Firebase Auth, a standard security practice.
  + **Password Enforcement:** New users or those with a default password are required to set a new password on their first login. This is a critical step in preventing unauthorized access.
  + **Role-Based Access:** The AdminPanel is protected by checking a user's isAdmin flag, which is a form of client-side role-based access control.
  + **Firestore Rules:** The application's security model is built on Firebase's backend security rules. It is **essential** that these rules are configured to ensure that only authenticated users can read and write data, and that only administrators can modify configuration data.
* **Data Integrity & Auditing:**
  + **Immutability:** The history field in every incident record creates an immutable audit trail. Each time an incident is updated, a new entry is added with a timestamp and the user's name, providing a transparent record of all actions.
  + **Centralized Storage:** All data is stored in Firebase Firestore, a secure cloud database, which mitigates the risk of data loss from local PC failures.
* **PWA Cache Strategy:** The app leverages the PWA caching API to provide reliable offline access. A robust caching strategy is configured to ensure that core application assets are available even without an internet connection, enhancing usability and data reliability in areas with poor network coverage.

### 5.3 Customization

* **Theme:** The tailwind.config.js file defines a full light and dark mode color palette. You can easily modify these colors to match your organization's branding.
* **Data:** Administrative users can manage the core lists of mines, sections, and incident types directly from the AdminPanel page, allowing the application to be easily customized for any new mine or department.

C:\Users\Sony\Desktop\safety incident report\src>tree /F

C:.

│ App.jsx

│ firebaseConfig.js

│ index.css

│ main.jsx

│ tree

│

├───components

│ AdminNoticeManager.jsx

│ AssignSections.jsx

│ BackToTopButton.jsx

│ BottomNav.jsx

│ CustomSelect.jsx

│ FloatingFilterBar.jsx

│ FloatingNav.jsx

│ IncidentCard.jsx

│ IncidentReportPDF.jsx

│ LogoIcon.jsx

│ PageControlBar.jsx

│ Sidebar.jsx

│

├───context

│ AppContext.jsx

│ AuthContext.jsx

│ ConfigContext.jsx

│

├───pages

│ AdminPanel.jsx

│ ComparisonPage.jsx

│ ExecutiveDashboardPage.jsx

│ HomePage.jsx

│ IncidentLogPage.jsx

│ IncidentReportPDF.jsx

│ LoginPage.jsx

│ ReportIncidentPage.jsx

│ SecondDashboardPage.jsx

│ SettingsPage.jsx

│ vite.svg

│

└───utils

formatters.js

mockData.js

mockData.jsx