Mon Valley Pollution Tracking System

Master Plan & System Architecture

Version: 1.0
Date: July 23, 2025
Status: Final
Client: Valley Clean Air Now (VCAN)
Implementer: Liberate X
Project: Mon Valley Pollution Tracking System

Table of Contents

---

1. Executive Summary

1.1 Project Overview

The Mon Valley Pollution Tracking System is a comprehensive, real-time environmental monitoring and health correlation platform designed to empower the residents of the Mon Valley with transparent pollution data and early warning capabilities. This system will integrate multiple data sources, provide community health reporting tools, and deliver actionable insights through advanced analytics and AI.

1.2 Strategic Approach

This master plan outlines a Firebase-centric, open-source architecture that balances cost-effectiveness with scalability. The system is designed to:

- **Start Small, Scale Big**: Begin with the immediate scope of work while building a foundation that supports the full long-term vision
- **Minimize Costs**: Leverage Google Firebase's generous free tier and open-source technologies
- **Ensure Quality**: Implement enterprise-grade security, performance, and maintainability standards
- **Enable Handoffs**: Provide comprehensive documentation for seamless developer transitions

1.3 Key Success Factors

1. **Modular Architecture**: Microservices-based design using Cloud Functions
2. **Real-time Capabilities**: Instant data processing and alerting
3. **Community-Centric Design**: User-friendly interfaces for diverse community members
4. **Data Privacy**: HIPAA-aligned principles for health data protection
5. **Scalable Infrastructure**: Serverless components that grow with demand

2. Project Vision & Scope

2.1 Long-Term Vision

To create a scalable, intelligent platform that leverages advanced data analytics and artificial intelligence to:

- Precisely identify pollution sources and their health impacts
- Provide real-time early warning systems for environmental hazards
- Enable data-driven advocacy for environmental justice
- Support community empowerment through transparent access to environmental data
- Facilitate research and policy development through comprehensive data collection

2.2 Initial Scope of Work

The project is structured in four phases as defined in the scope of work:

Phase I: System Design and Integration Strategy

- Modular architecture plan integrating PurpleAir, Sniffer4D, Summa Canister, and NASA satellite data
- Data Governance Framework covering ownership, privacy, storage, and access protocols
- Technical implementation roadmap with software stack and deployment schema

Phase II: Community Health Reporting and AI Development

- Resident-facing symptom tracking module using OSAC framework
- BreatheAI virtual assistant integration with conversational architecture
- Accessibility, cultural responsiveness, and linguistic inclusivity evaluation

Phase III: Policy Alignment and Regulatory Mapping

- Regulatory matrix for PM2.5, VOCs, ozone, and other pollutants

- Real-time and forecast-based health alert protocols
- Technical briefing document for stakeholders

Phase IV: Reporting, Evaluation, and Knowledge Transfer

- Comprehensive final report with outcomes and strategic opportunities
- Live presentation to VCAN leadership and community partners
- Technical consultation and troubleshooting support

2.3 Scalability Considerations

The architecture is designed to support expansion from the Mon Valley pilot to: - Regional and national deployment - Additional data sources and sensor types - Advanced AI/ML capabilities for predictive modeling - Integration with healthcare systems and research institutions

---

3. System Architecture

3.1 Architectural Principles

The system follows these core architectural principles:

1. **Serverless-First**: Utilize Cloud Functions for automatic scaling and cost optimization
2. **Event-Driven**: Implement reactive architecture using Pub/Sub and Firestore triggers
3. **Microservices**: Decompose functionality into independent, deployable services
4. **API-Centric**: Design all components to communicate through well-defined APIs
5. **Security by Design**: Implement authentication, authorization, and encryption at every layer

3.2 High-Level Architecture Diagram

```
┌──────────────────────────────────────────────────────────────────────────────┐
│                          EXTERNAL DATA SOURCES                               │
├─────────────────┬─────────────────┬─────────────────┬──────────────────────┤
│   PurpleAir     │   Sniffer4D     │ NASA Satellite  │   Summa Canisters    │
│   API Polling   │   MQTT/UDP      │   API Polling   │   Manual Upload      │
└─────────────────┴─────────────────┴─────────────────┴──────────────────────┘
          │                 │                 │                 │
          ▼                 ▼                 ▼                 ▼
┌──────────────────────────────────────────────────────────────────────────────┐
│                          DATA INGESTION LAYER                                │
├─────────────────┬─────────────────┬─────────────────┬──────────────────────┤
│ Cloud Function  │  IoT Core +     │ Cloud Function  │  Cloud Function      │
│ + Scheduler     │  Pub/Sub        │ + Scheduler     │  + Storage Trigger   │
└─────────────────┴─────────────────┴─────────────────┴──────────────────────┘
          │                 │                 │                 │
          └─────────────────┼─────────────────┼─────────────────┘
                            ▼                 ▼
┌──────────────────────────────────────────────────────────────────────────────┐
│                         DATA PROCESSING LAYER                                │
├──────────────────────────────────────────────────────────────────────────────┤
│  • Data Validation & Cleaning (Cloud Functions)                              │
│  • Bias Correction (PurpleAir EPA correction)                                │
│  • Data Transformation & Normalization                                       │
│  • Quality Assurance & Anomaly Detection                                     │
└──────────────────────────────────────────────────────────────────────────────┘
                               │
                               ▼
┌──────────────────────────────────────────────────────────────────────────────┐
│                           DATA STORAGE LAYER                                 │
├──────────────────────────────────────────────────────────────────────────────┤
│                          Cloud Firestore                                     │
│  ┌─────────────┬─────────────┬─────────────┬─────────────┬─────────────┐   │
│  │   sensors   │sensorReadings│    users    │symptomReports│   alerts    │  │
│  └─────────────┴─────────────┴─────────────┴─────────────┴─────────────┘   │
└──────────────────────────────────────────────────────────────────────────────┘
                               │
                               ▼
┌──────────────────────────────────────────────────────────────────────────────┐
│                        ANALYTICS & AI/ML LAYER                               │
├──────────────────────────────────────────────────────────────────────────────┤
│  • Statistical Correlation Analysis (scikit-learn)                           │
│  • Predictive Modeling (TensorFlow.js)                                       │
│  • BreatheAI Conversational Bot (Natural Language Processing)                │
│  • Real-time Alert Generation                                                │
│  • Data Aggregation & Insights Generation                                    │
└──────────────────────────────────────────────────────────────────────────────┘
                               │
                               ▼
┌──────────────────────────────────────────────────────────────────────────────┐
│                          PRESENTATION LAYER                                  │
├──────────────────────────────────────────────────────────────────────────────┤
│                       React.js Web Application                               │
│  ┌─────────────┬─────────────┬─────────────┬─────────────┬─────────────┐   │
│  │  Dashboard  │    Maps     │  Symptom    │   Alerts    │   Reports   │   │
│  │             │ (Leaflet.js)│  Tracking   │             │             │   │
│  └─────────────┴─────────────┴─────────────┴─────────────┴─────────────┘   │
│                          Firebase Hosting                                    │
└──────────────────────────────────────────────────────────────────────────────┘
                               │
                               ▼
┌──────────────────────────────────────────────────────────────────────────────┐
```

```
│                      NOTIFICATION SERVICES                            │
├───────────────────────────────────────────────────────────────────────┤
│  • Firebase Cloud Messaging (Push Notifications)                       │
│  • SendGrid/Mailgun (Email Alerts)                                     │
│  • Twilio (SMS Alerts)                                                 │
└───────────────────────────────────────────────────────────────────────┘
```

3.3 Data Flow Architecture

```
 External Data → Ingestion Functions → Processing Functions → Firestore →
Analytics Functions → Insights Storage → Web App Display → User Actions →
Symptom Reports → Correlation Analysis → Alert Generation → Notifications
```

3.4 Component Interaction Diagram

```
┌─────────────────┐     ┌─────────────────┐     ┌─────────────────┐
│   Web Client    │◄───►│ Firebase Auth   │◄───►│ Firestore DB    │
│   (React.js)    │     │                 │     │                 │
└─────────────────┘     └─────────────────┘     └─────────────────┘
         │                                               │
         ▼                                               ▼
┌─────────────────┐     ┌─────────────────┐     ┌─────────────────┐
│ Firebase        │◄───►│ Cloud Functions │◄───►│ External APIs   │
│ Hosting         │     │ (Backend Logic) │     │ (PurpleAir,etc) │
└─────────────────┘     └─────────────────┘     └─────────────────┘
                                 │
                                 ▼
                        ┌─────────────────┐
                        │ Notification    │
                        │ Services        │
                        └─────────────────┘
```

4. Technology Stack

4.1 Core Firebase Services

Service	Purpose	Cost Optimization
Firebase Authentication	User management, secure login	Free tier: 50,000 MAU
Cloud Firestore	Primary database for all data	Free tier: 1GB storage, 50K reads/day
Cloud Functions	Serverless backend logic	Free tier: 2M invocations/month
Firebase Hosting	Web app deployment	Free tier: 10GB storage, 10GB transfer
Cloud Storage	File storage for uploads	Free tier: 5GB storage
Firebase Cloud Messaging	Push notifications	Completely free

4.2 Open Source Technologies

Component	Technology	License	Purpose
Frontend Framework	React.js	MIT	User interface development
Mapping Library	Leaflet.js	BSD-2-Clause	Interactive maps and geospatial visualization
Data Visualization	Chart.js	MIT	Charts and graphs for data display
Machine Learning	TensorFlow.js	Apache 2.0	Client-side and server-side ML models
Statistical Analysis	scikit-learn (Python)	BSD	Correlation analysis and data science
Natural Language Processing	Natural (Node.js)	MIT	BreatheAI conversational capabilities
HTTP Client	Axios	MIT	API communication
State Management	Redux Toolkit	MIT	Application state management
UI Components	Material-UI	MIT	Consistent, accessible UI components

4.3 Third-Party Services (Free Tiers)

Service	Free Tier Limits	Purpose
SendGrid	100 emails/day	Email notifications
Twilio	Trial credits	SMS alerts
Mapbox	50,000 map loads/month	Advanced mapping features (optional)

4.4 Development Tools

Tool	Purpose	Cost
Firebase CLI	Deployment and management	Free
Node.js & npm	Development environment	Free
Git & GitHub	Version control	Free for public repos
VS Code	Code editor	Free
Postman	API testing	Free tier available

---

5. Data Models & Database Design

5.1 Firestore Collections Structure

5.1.1 `sensors` Collection

Stores metadata about each monitoring device or data source.

```
sensors/{sensorId} = {
  source: "PurpleAir" | "Sniffer4D" | "NASA" | "Summa",
  name: "Main Street Sensor",
  location: new firebase.firestore.GeoPoint(latitude, longitude),
  status: "active" | "inactive" | "maintenance",
  installDate: firebase.firestore.Timestamp,
  lastReadingAt: firebase.firestore.Timestamp,
  metadata: {
    model: "PA-II",
    serialNumber: "ABC123",
    calibrationDate: firebase.firestore.Timestamp,
    owner: "Community Group"
  }
}
```

5.1.2 `sensorReadings` Collection

Time-series data from all sensors. This will be the largest collection.

```
sensorReadings/{readingId} = {
  sensorId: firebase.firestore.DocumentReference,
  timestamp: firebase.firestore.Timestamp,
  location: firebase.firestore.GeoPoint, // For mobile sensors like Sniffer4D
  data: {
    pm2_5: 15.2,        // μg/m³
    pm10: 25.1,         // μg/m³
    voc: 0.8,           // ppm
    o3: 45.3,           // ppb
    no2: 12.7,          // ppb
    temperature: 22.5,   // °C
    humidity: 65.2,      // %
    pressure: 1013.25    // hPa
  },
  qualityFlags: {
    isCorrected: true,   // EPA bias correction applied
    confidence: 0.95,    // Data quality confidence
    anomaly: false       // Anomaly detection flag
  },
  source: "PurpleAir",
  rawData: {}            // Original uncorrected data
}
```

5.1.3 `users` Collection

User profiles linked to Firebase Authentication.

```
users/{userId} = {  // userId matches Firebase Auth UID
  email: "user@example.com",
  displayName: "John Doe",
  createdAt: firebase.firestore.Timestamp,
  lastLoginAt: firebase.firestore.Timestamp,
  roles: ["communityMember", "researcher", "advocate"],
  profile: {
    age: 35,
    zipCode: "15120",
    healthConditions: ["asthma", "copd"], // Optional, encrypted
    occupation: "Teacher"
  },
  notificationPreferences: {
    emailEnabled: true,
    smsEnabled: false,
    pushEnabled: true,
    alertThresholds: {
      pm2_5: 35.0,       // Custom alert threshold
      aqi: 100
    },
    smsNumber: "+1234567890"
  },
  privacy: {
    shareLocation: true,
    shareSymptoms: false,
    researchParticipation: true
  }
}
```

5.1.4 `symptomReports` Collection

Sensitive health data with strict access controls.

```
symptomReports/{reportId} = {
  userId: firebase.firestore.DocumentReference,
  timestamp: firebase.firestore.Timestamp,
  location: firebase.firestore.GeoPoint,
  symptoms: ["coughing", "headache", "shortness_of_breath"],
  severity: 3,           // 1-5 scale
  duration: "2-4 hours", // How long symptoms lasted
  perceivedCause: "Industrial smell from steel mill",
  environmentalFactors: {
    indoorOutdoor: "outdoor",
    activity: "walking",
    weather: "clear"
  },
  osac: {              // OSAC Framework data
    onset: "sudden",
    severity: "moderate",
    aggravatingFactors: ["physical_activity"],
    alleviatingFactors: ["moved_indoors"]
  },
  anonymized: false,    // Whether this can be used in aggregated data
  followUp: {
    medicalAttention: false,
    medication: "inhaler"
  }
}
```

5.1.5 `alerts` Collection

System-generated alerts and notifications.

```
alerts/{alertId} = {
  type: "pollution_spike" | "health_advisory" | "system_maintenance",
  severity: "low" | "moderate" | "high" | "emergency",
  title: "High PM2.5 Levels Detected",
  message: "PM2.5 levels have exceeded 35 μg/m³ in your area.",
  createdAt: firebase.firestore.Timestamp,
  expiresAt: firebase.firestore.Timestamp,
  location: firebase.firestore.GeoPoint,
  radius: 5000,         // Alert radius in meters
  triggerData: {
    sensorId: firebase.firestore.DocumentReference,
    value: 45.2,
    threshold: 35.0,
    pollutant: "pm2_5"
  },
  recipients: {
    total: 150,
    sent: 148,
    failed: 2
  },
  status: "active" | "resolved" | "expired"
}
```

5.1.6 `insights` Collection

AI-generated insights and correlations.

```
insights/{insightId} = {
  type: "correlation" | "prediction" | "trend",
  title: "PM2.5 Correlation with Respiratory Symptoms",
  description: "Analysis shows 23% increase in respiratory symptoms when PM2.5
> 35 μg/m³",
  createdAt: firebase.firestore.Timestamp,
  dataRange: {
    startDate: firebase.firestore.Timestamp,
    endDate: firebase.firestore.Timestamp
  },
  statistics: {
    correlation: 0.67,
    confidence: 0.89,
    sampleSize: 1250,
    pValue: 0.001
  },
  visualization: {
    chartType: "scatter",
    dataUrl: "gs://bucket/chart-data.json"
  },
  methodology: "Pearson correlation analysis",
  limitations: "Self-reported symptoms may have reporting bias"
}
```

5.2 Data Relationships

```
 sensors (1) ───────▶ (many) sensorReadings
users (1) ──────────▶ (many) symptomReports
users (many) ◀─────▶ (many) alerts (via location/preferences)
sensorReadings (many) ───▶ (1) insights (aggregated analysis)
symptomReports (many) ────▶ (1) insights (aggregated analysis)
```

5.3 Indexing Strategy

Critical indexes for performance:

```
// Firestore Indexes (configured via Firebase Console or CLI)
sensorReadings:
  - timestamp (descending)
  - sensorId, timestamp (descending)
  - location (geo), timestamp (descending)

symptomReports:
  - userId, timestamp (descending)
  - location (geo), timestamp (descending)
  - timestamp (descending), anonymized (ascending)

alerts:
  - location (geo), status (ascending)
  - createdAt (descending), severity (ascending)
```

---

6. Implementation Roadmap

6.1 Phase 1: Foundation & Core Infrastructure (Months 1-4)

6.1.1 Sprint 1: Project Setup (Weeks 1-2)

Objectives: Establish development environment and basic project structure

Tasks: - [ ] Initialize Firebase project with all required services - [ ] Set up GitHub repository with proper branching strategy - [ ] Configure development, staging, and production environments - [ ] Create initial React.js application using Create React App - [ ] Set up Firebase CLI and deployment scripts - [ ] Configure Firestore security rules (basic version) - [ ] Set up continuous integration/deployment pipeline

Deliverables: - Working Firebase project - Basic web application deployed to Firebase Hosting - Development environment documentation

6.1.2 Sprint 2: Authentication & User Management (Weeks 3-4)

Objectives: Implement secure user authentication and basic user profiles

Tasks: - [ ] Implement Firebase Authentication (Email/Password, Google Sign-In) - [ ] Create user registration and login UI components - [ ] Design and implement user profile management - [ ] Set up role-based access control (RBAC) foundation - [ ] Create user onboarding flow - [ ] Implement password reset functionality

Deliverables: - Complete authentication system - User profile management interface - Security documentation

6.1.3 Sprint 3: Data Ingestion - PurpleAir (Weeks 5-6)

Objectives: Establish first data source integration

Tasks: - [ ] Research PurpleAir API documentation and rate limits - [ ] Create Cloud Function for PurpleAir data polling - [ ] Implement EPA bias correction algorithm - [ ] Set up Cloud Scheduler for regular data collection - [ ] Design and implement sensor registration system - [ ] Create data validation and error handling - [ ] Set up monitoring and alerting for ingestion failures

Deliverables: - Working PurpleAir data ingestion pipeline - Sensor management interface - Data quality monitoring dashboard

6.1.4 Sprint 4: Basic Visualization (Weeks 7-8)

Objectives: Create initial data visualization and mapping

Tasks: - [ ] Integrate Leaflet.js mapping library - [ ] Create sensor location display on map - [ ] Implement real-time data display - [ ] Design responsive UI for mobile and desktop - [ ] Create basic dashboard with key metrics - [ ] Implement data filtering and time range selection

Deliverables: - Interactive map with sensor data - Basic dashboard interface - Mobile-responsive design

6.2 Phase 2: Community Health & Advanced Features (Months 5-9)

6.2.1 Sprint 5: Symptom Reporting Module (Weeks 9-12)

Objectives: Enable community health data collection

Tasks: - [ ] Design OSAC framework integration - [ ] Create symptom reporting UI with accessibility features - [ ] Implement secure health data storage - [ ] Design privacy controls and consent management - [ ] Create symptom history and trends visualization - [ ] Implement data anonymization for research use

Deliverables: - Complete symptom reporting system - Privacy and consent management - Health data visualization tools

6.2.2 Sprint 6: BreatheAI Virtual Assistant (Weeks 13-16)

Objectives: Develop conversational AI for user guidance

Tasks: - [ ] Design conversational flow and decision trees - [ ] Implement natural language processing using Natural.js - [ ] Create chatbot UI component - [ ] Integrate with symptom reporting system - [ ] Develop personalized recommendations engine - [ ] Implement multi-language support foundation

Deliverables: - Working BreatheAI chatbot - Conversational interface - Recommendation system

6.2.3 Sprint 7: Additional Data Sources (Weeks 17-20)

Objectives: Expand data collection capabilities

Tasks: - [ ] Implement Sniffer4D drone data integration - [ ] Set up NASA satellite data ingestion - [ ] Create Summa canister data upload system - [ ] Develop data harmonization and standardization - [ ] Implement multi-source data correlation - [ ] Create data source management dashboard

Deliverables: - Multi-source data integration - Data harmonization system - Comprehensive data management tools

6.3 Phase 3: Analytics & Intelligence (Months 10-15)

6.3.1 Sprint 8: Alert System (Weeks 21-24)

Objectives: Implement real-time alerting and notifications

Tasks: - [ ] Design alert threshold configuration system - [ ] Implement real-time data monitoring - [ ] Create multi-channel notification system (email, SMS, push) - [ ] Develop escalation workflows - [ ] Implement geofencing for location-based alerts - [ ] Create alert history and analytics

Deliverables: - Complete alerting system - Multi-channel notifications - Alert management interface

6.3.2 Sprint 9: Correlation Analysis (Weeks 25-28)

Objectives: Develop AI-powered insights and correlations

Tasks: - [ ] Implement statistical correlation analysis using scikit-learn - [ ] Create automated insight generation - [ ] Develop trend analysis and forecasting - [ ] Implement machine learning models for pattern recognition - [ ] Create public health insights dashboard - [ ] Develop research data export capabilities

Deliverables: - AI-powered correlation analysis - Insights generation system - Research tools and dashboards

6.3.3 Sprint 10: Advanced Visualization (Weeks 29-32)

Objectives: Create sophisticated data visualization tools

Tasks: - [ ] Implement 3D pollution modeling visualization - [ ] Create time-series analysis tools - [ ] Develop heat maps and pollution dispersion models - [ ] Implement advanced filtering and data exploration - [ ] Create exportable reports and visualizations - [ ] Develop stakeholder-specific dashboards

Deliverables: - Advanced visualization suite - 3D modeling capabilities - Stakeholder reporting tools

6.4 Phase 4: Optimization & Scaling (Months 16-18)

6.4.1 Sprint 11: Performance Optimization (Weeks 33-36)

Objectives: Optimize system performance and scalability

Tasks: - [ ] Implement database query optimization - [ ] Set up caching strategies - [ ] Optimize Cloud Function performance - [ ] Implement data archiving and lifecycle management - [ ] Conduct load testing and performance tuning - [ ] Optimize mobile application performance

Deliverables: - Performance-optimized system - Scalability documentation - Load testing results

6.4.2 Sprint 12: Final Integration & Testing (Weeks 37-40)

Objectives: Complete system integration and comprehensive testing

Tasks: - [ ] Conduct end-to-end system testing - [ ] Implement comprehensive error handling - [ ] Create system monitoring and observability - [ ] Conduct security audit and penetration testing - [ ] Finalize documentation and user guides - [ ] Prepare for production deployment

Deliverables: - Production-ready system - Complete documentation - Security audit report

---

7. Security & Compliance

7.1 Authentication & Authorization

7.1.1 Firebase Authentication Implementation

```javascript
 // Authentication configuration
const firebaseConfig = {
  // Firebase config
  authDomain: "mon-valley-pollution.firebaseapp.com",
  // Other config...
};

// Multi-provider authentication
const authProviders = {
  email: new firebase.auth.EmailAuthProvider(),
  google: new firebase.auth.GoogleAuthProvider(),
  phone: new firebase.auth.PhoneAuthProvider()
};

// Role-based access control
const userRoles = {
  COMMUNITY_MEMBER: 'communityMember',
  RESEARCHER: 'researcher',
  ADVOCATE: 'advocate',
  ADMIN: 'admin'
};
```

7.1.2 Firestore Security Rules

```
 rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    // Users can only access their own data
    match /users/{userId} {
      allow read, write: if request.auth != null && request.auth.uid == userId;
    }

    // Symptom reports - highly restricted
    match /symptomReports/{reportId} {
      allow create: if request.auth != null &&
                    request.auth.uid == resource.data.userId;
      allow read, update: if request.auth != null &&
                          request.auth.uid == resource.data.userId;
      allow delete: if false; // Never allow deletion
    }

    // Sensor data - read-only for authenticated users
    match /sensors/{sensorId} {
      allow read: if request.auth != null;
      allow write: if request.auth != null &&
                    hasRole(request.auth.uid, 'admin');
    }

    match /sensorReadings/{readingId} {
      allow read: if request.auth != null;
      allow write: if false; // Only Cloud Functions can write
    }

    // Public insights - read-only
    match /insights/{insightId} {
      allow read: if true; // Public data
      allow write: if false; // Only Cloud Functions can write
    }

    // Helper function for role checking
    function hasRole(userId, role) {
      return
get(/databases/$`(database)/documents/users/`$(userId)).data.roles.hasAny([role])
    }
  }
}
```

7.2 Data Privacy & HIPAA Alignment

7.2.1 Privacy by Design Principles

1. **Data Minimization**: Collect only necessary health information
2. **Purpose Limitation**: Use data only for stated purposes
3. **Consent Management**: Clear, granular consent for data use
4. **Right to Deletion**: Allow users to delete their data

5. **Data Portability**: Enable data export in standard formats

7.2.2 Health Data Protection

```
 // Encryption for sensitive health data
const encryptHealthData = (data, userKey) => {
  // Implement client-side encryption before storage
  return encrypt(JSON.stringify(data), userKey);
};

// Anonymization for research use
const anonymizeSymptomData = (symptomReport) => {
  return {
    timestamp: symptomReport.timestamp,
    location: fuzzyLocation(symptomReport.location), // Reduce precision
    symptoms: symptomReport.symptoms,
    severity: symptomReport.severity,
    // Remove all personally identifiable information
  };
};
```

7.2.3 Consent Management System

```
 // Consent tracking
const consentTypes = {
  DATA_COLLECTION: 'dataCollection',
  RESEARCH_PARTICIPATION: 'researchParticipation',
  MARKETING_COMMUNICATIONS: 'marketingCommunications',
  DATA_SHARING: 'dataSharing'
};

// User consent record
const userConsent = {
  userId: 'user123',
  consents: {
    [consentTypes.DATA_COLLECTION]: {
      granted: true,
      timestamp: new Date(),
      version: '1.0'
    },
    [consentTypes.RESEARCH_PARTICIPATION]: {
      granted: false,
      timestamp: new Date(),
      version: '1.0'
    }
  }
};
```

7.3 Security Monitoring & Incident Response

7.3.1 Security Monitoring

```
 // Cloud Function for security monitoring
exports.securityMonitor = functions.firestore
  .document('users/{userId}')
  .onUpdate((change, context) => {
    const before = change.before.data();
    const after = change.after.data();

    // Monitor for suspicious activity
    if (detectSuspiciousActivity(before, after)) {
      // Alert security team
      sendSecurityAlert({
        userId: context.params.userId,
        activity: 'suspicious_profile_change',
        timestamp: new Date()
      });
    }
  });
```

7.3.2 Incident Response Plan

1. **Detection**: Automated monitoring and alerting
2. **Assessment**: Rapid evaluation of security incidents
3. **Containment**: Immediate steps to limit damage
4. **Eradication**: Remove threats and vulnerabilities
5. **Recovery**: Restore normal operations
6. **Lessons Learned**: Post-incident analysis and improvements

---

8. Cost Optimization Strategy

8.1 Firebase Free Tier Utilization

Service	Free Tier Limit	Estimated Usage	Cost After Free Tier
Firestore	1GB storage, 50K reads/day	500MB, 30K reads/day	$0
Cloud Functions	2M invocations/month	1M invocations/month	$0
Authentication	50,000 MAU	5,000 MAU	$0
Hosting	10GB storage, 10GB transfer	2GB storage, 5GB transfer	$0
Cloud Storage	5GB storage	2GB storage	$0

Estimated Monthly Cost (Phase 1): $0

8.2 Scaling Cost Projections

Phase 2 (Community Health Features)

- Firestore: ~$20/month (increased reads/writes)
- Cloud Functions: ~$10/month (additional processing)
- **Total**: ~$30/month

Phase 3 (Full Analytics)

- Firestore: ~$50/month
- Cloud Functions: ~$25/month
- External APIs: ~$15/month (SendGrid, Twilio)
- **Total**: ~$90/month

Production Scale (1000+ active users)

- Firestore: ~$150/month
- Cloud Functions: ~$75/month
- External Services: ~$50/month
- **Total**: ~$275/month

8.3 Cost Optimization Techniques

1. **Efficient Queries**: Use compound indexes and limit query results
2. **Caching**: Implement client-side and server-side caching
3. **Data Lifecycle**: Archive old data to cheaper storage
4. **Function Optimization**: Minimize cold starts and execution time
5. **Batch Operations**: Group multiple operations to reduce function calls

---

9. Development Guidelines

9.1 Code Standards & Best Practices

9.1.1 JavaScript/React Standards

```
// Use functional components with hooks
const SensorMap = ({ sensors, onSensorSelect }) => {
  const [selectedSensor, setSelectedSensor] = useState(null);

  useEffect(() => {
    // Load sensor data
    loadSensorData();
  }, []);

  return (
    <div className="sensor-map">
      {/* Component JSX */}
    </div>
  );
};

// PropTypes for type checking
SensorMap.propTypes = {
  sensors: PropTypes.arrayOf(PropTypes.object).isRequired,
  onSensorSelect: PropTypes.func.isRequired
};
```

9.1.2 Cloud Functions Standards

```
// Use async/await for better error handling
exports.processSensorData = functions.firestore
  .document('sensorReadings/{readingId}')
  .onCreate(async (snap, context) => {
    try {
      const reading = snap.data();

      // Process the reading
      const processedData = await processReading(reading);

      // Check for alerts
      await checkAlertThresholds(processedData);

      return { success: true };
    } catch (error) {
      console.error('Error processing sensor data:', error);
      throw new functions.https.HttpsError('internal', 'Processing failed');
    }
  });
```

9.2 Testing Strategy

9.2.1 Unit Testing

```
// Jest testing for React components
import { render, screen, fireEvent } from '@testing-library/react';
import SensorMap from './SensorMap';

describe('SensorMap', () => {
  test('renders sensor markers', () => {
    const mockSensors = [
      { id: '1', name: 'Sensor 1', location: { lat: 40.4, lng: -79.9 } }
    ];

    render(<SensorMap sensors={mockSensors} onSensorSelect={jest.fn()} />);

    expect(screen.getByText('Sensor 1')).toBeInTheDocument();
  });
});
```

9.2.2 Integration Testing

```
// Firebase emulator testing
const { initializeTestApp, clearFirestoreData } = require('@firebase/rules-
unit-testing');

describe('Firestore Security Rules', () => {
  beforeEach(async () => {
    await clearFirestoreData({ projectId: 'test-project' });
  });

  test('users can only read their own symptom reports', async () => {
    const db = initializeTestApp({ projectId: 'test-project', auth: { uid:
'user1' } }).firestore();

    // Test security rules
    await expect(
      db.collection('symptomReports').doc('user2-report').get()
    ).rejects.toThrow();
  });
});
```

9.3 Deployment & CI/CD

9.3.1 GitHub Actions Workflow

```yaml
name: Deploy to Firebase

on:
  push:
    branches: [ main ]

jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2

    - name: Setup Node.js
      uses: actions/setup-node@v2
      with:
        node-version: '16'

    - name: Install dependencies
      run: npm ci

    - name: Run tests
      run: npm test

    - name: Build project
      run: npm run build

    - name: Deploy to Firebase
      uses: FirebaseExtended/action-hosting-deploy@v0
      with:
        repoToken: '${{ secrets.GITHUB_TOKEN }}'
        firebaseServiceAccount: '${{ secrets.FIREBASE_SERVICE_ACCOUNT }}'
        projectId: mon-valley-pollution
```

9.4 Documentation Standards

9.4.1 Code Documentation

```javascript
/**
 * Processes raw sensor data and applies corrections
 * @param {Object} rawData - Raw sensor reading data
 * @param {string} sensorType - Type of sensor (PurpleAir, Sniffer4D, etc.)
 * @returns {Promise<Object>} Processed and corrected sensor data
 * @throws {Error} When sensor type is not supported
 */
async function processSensorData(rawData, sensorType) {
  // Implementation
}
```

9.4.2 API Documentation

```
/**
 * @api {post} /api/symptom-report Submit Symptom Report
 * @apiName SubmitSymptomReport
 * @apiGroup Health
 *
 * @apiParam {String} userId User ID
 * @apiParam {Array} symptoms List of symptoms
 * @apiParam {Number} severity Severity level (1-5)
 *
 * @apiSuccess {String} reportId Generated report ID
 * @apiSuccess {String} message Success message
 *
 * @apiError {String} error Error message
 */
```

---

10. Appendices

10.1 Appendix A: External API Documentation

10.1.1 PurpleAir API Integration

```
// PurpleAir API configuration
const PURPLEAIR_CONFIG = {
  baseURL: 'https://api.purpleair.com/v1',
  apiKey: process.env.PURPLEAIR_API_KEY,
  rateLimit: 100, // requests per minute
  endpoints: {
    sensors: '/sensors',
    sensorData: '/sensors/{sensor_index}'
  }
};

// EPA bias correction formula
const applyEPACorrection = (pm25Raw) => {
  // EPA correction for PurpleAir sensors
  return 0.52 * pm25Raw - 0.085 * (pm25Raw * pm25Raw) / 1000 + 5.71;
};
```

10.1.2 NASA Satellite Data Integration

```
// NASA API configuration
const NASA_CONFIG = {
  baseURL: 'https://api.nasa.gov',
  apiKey: process.env.NASA_API_KEY,
  datasets: {
    MODIS: 'modis/terra',
    OMI: 'omi/aura',
    TEMPO: 'tempo/geostationary'
  }
};
```

10.2 Appendix B: Regulatory Standards Matrix

Pollutant	WHO Standard	EPA Standard	Local Standard	Alert Threshold
PM2.5	15 μg/m^3 (annual)	12 μg/m^3 (annual)	TBD	35 μg/m^3 (24hr)
PM10	45 μg/m^3 (annual)	150 μg/m^3 (24hr)	TBD	150 μg/m^3 (24hr)
Ozone	100 μg/m^3 (8hr)	70 ppb (8hr)	TBD	70 ppb (8hr)
NO2	40 μg/m^3 (annual)	100 ppb (1hr)	TBD	100 ppb (1hr)
VOCs	Varies by compound	Varies by compound	TBD	TBD

10.3 Appendix C: OSAC Framework Implementation

```
// OSAC (Onset, Severity, Aggravating factors, Course) Framework
const OSACFramework = {
  onset: {
    options: ['sudden', 'gradual', 'intermittent'],
    required: true
  },
  severity: {
    scale: [1, 2, 3, 4, 5],
    labels: ['Mild', 'Moderate', 'Severe', 'Very Severe', 'Extreme'],
    required: true
  },
  aggravatingFactors: {
    options: [
      'physical_activity',
      'outdoor_exposure',
      'industrial_smell',
      'weather_conditions',
      'time_of_day'
    ],
    multiple: true
  },
  course: {
    options: ['improving', 'stable', 'worsening'],
    required: true
  }
};
```

10.4 Appendix D: Performance Benchmarks

Metric	Target	Measurement Method
Page Load Time	< 3 seconds	Lighthouse audit
API Response Time	< 500ms	Cloud Function monitoring
Database Query Time	< 100ms	Firestore performance monitoring
Mobile Performance	> 90 Lighthouse score	Mobile audit
Accessibility	WCAG 2.1 AA compliance	Automated testing

10.5 Appendix E: Disaster Recovery Plan

10.5.1 Backup Strategy

```
// Automated Firestore backup
exports.scheduledFirestoreBackup = functions.pubsub
  .schedule('every 24 hours')
  .onRun(async (context) => {
    const client = new firestore.v1.FirestoreAdminClient();

    const databaseName = client.databasePath(
      process.env.GCLOUD_PROJECT,
      '(default)'
    );

    const bucket = `gs://${process.env.BACKUP_BUCKET}`;

    return client.exportDocuments({
      name: databaseName,
      outputUriPrefix: bucket,
      collectionIds: []
    });
  });
```

10.5.2 Recovery Procedures

1. **Data Loss Recovery**: Restore from automated Firestore backups
2. **Service Outage**: Failover to backup Firebase project
3. **Security Breach**: Immediate access revocation and audit
4. **Performance Degradation**: Auto-scaling and load balancing

---

Conclusion

This master plan provides a comprehensive blueprint for Liberate X to develop the Mon Valley Pollution Tracking System for Valley Clean Air Now (VCAN) using Google Firebase and open-source technologies. The architecture is designed to be cost-effective, scalable, and maintainable, ensuring that the system can grow from the initial scope of work to support the full long-term vision.

The phased implementation approach allows for incremental value delivery while building a robust foundation for future enhancements. The emphasis on security, privacy, and community-centric design ensures that the system will serve the needs of the Mon Valley community effectively and responsibly.

This document serves as the definitive guide for Liberate X's development team and any future developers working on the project, providing clear technical specifications, implementation guidelines, and best practices to ensure project success and continuity for VCAN.

Document Version: 1.0
Last Updated: July 23, 2025
Next Review: August 23, 2025