

SafeCity Infrastructure Reporter

System Documentation

1. Executive Summary

SafeCity is a Progressive Web Application (PWA) designed to empower citizens to report and track infrastructure issues in their communities. Built specifically for Johannesburg, the platform bridges the gap between citizens and local authorities through an intuitive reporting system, real-time tracking, and community engagement features.

Key Capabilities

- Real-time infrastructure issue reporting with GPS and photo capture
- Interactive mapping with filtering and search functionality
- User authentication and personalized dashboards
- Community voting and engagement features
- Cross-platform compatibility (web, mobile, desktop)
- Offline functionality through PWA technology

2. System Architecture

2.1 High-Level Architecture

Layer	Component	Technology / Notes
Client Layer	Web App	HTML / CSS / JavaScript
	Mobile PWA	Native Progressive Web App
	Desktop PWA	Electron-based Progressive Web App
Service Layer	Supabase API	Backend-as-a-Service (BaaS)
	EmailJS API	Email delivery via client-side API
	WhatsApp API	Messaging integration
Data Layer	Supabase PostgreSQL	Cloud-hosted relational database

	Users Table	Stores user credentials and profiles
	Reports Table	Stores incident or feedback reports
	User Statistics View	Aggregated analytics and metrics

Let me know if you'd like a visual diagram version, or if you want to add deployment notes, security layers, or data flow descriptions. I can also help format this for presentations or client proposals.

2.2 Technology Stack

Layer	Technology	Purpose
Frontend	HTML5, CSS3, JavaScript (ES6+)	User interface and interactions
Styling	CSS Grid, Flexbox, Custom Properties	Responsive design and theming
Database	Supabase (PostgreSQL)	Data storage and real-time updates
Authentication	Supabase Auth	User management and sessions
Notifications	EmailUS	Automated email alerts
PWA	Service Worker API	Offline functionality
Geolocation	Browser Geolocation API	GPS location detection
Media	MediaDevices API	Camera access for photos

2.3 Application Structure

Path	File/Folder	Description
SafeCityWeb/	index.html	Landing page
SafeCityWeb/pages/	dashboard.html	Main application interface
	offline.html	PWA offline fallback page
SafeCityWeb/css/	styles.css	Complete styling system
SafeCityWeb/js/	script.js	Main application logic
	auth.js	Authentication system

SafeCityWeb/assets/	logo-6.jpeg	Application logo
SafeCityWeb/	manifest.json	PWA configuration
	sw.js	Service worker for offline capabilities

3. Core Features & Functionality

3.1 User Authentication System

Authentication Flow:

User Registration → Email Verification → Profile Creation → Dashboard Access

User Login → Session Token → Persistent Login → Dashboard Access

Implementation:

- Supabase Auth handles secure authentication
- Session tokens stored securely
- Password encryption and validation
- Email verification for new accounts

3.2 Issue Reporting Workflow

1. User Location Detection (GPS)

↓

2. Issue Type Selection (6 categories)

↓

3. Photo Capture (Optional)

↓

4. Description Entry (Optional)

↓

5. Submit to Database

↓

6. Email Notification Sent







↓

7. Unique Report ID Generated



8. WhatsApp Share Option

Issue Categories:




-  Potholes
-  Water Leaks
-  Traffic Lights
-  Street Lights
-  Drainage Issues
-  Other Infrastructure

3.3 Interactive Map System

Features:

- Real-time issue visualization
- Color-coded status markers (New/Acknowledged/Resolved)
- Smart filtering by issue type
- Interactive hover effects showing upvotes and details
- Click for detailed popup information

Map States:

-  **New** - Recently reported issues
-  **Acknowledged** - Issues under review
-  **Resolved** - Completed repairs

3.4 Community Dashboard

Real-time Statistics:

- Total reports submitted
- Issues resolved count
- Average resolution time
- Active user count

- Trending issues by location and type

User Engagement:

- Upvote system for issue prioritization
 - Community impact metrics
 - Personal contribution tracking
-

4. Database Schema

4.1 Users Table (Managed by Supabase Auth)

```
CREATE TABLE auth.users (  
  id UUID PRIMARY KEY,  
  email VARCHAR UNIQUE NOT NULL,  
  encrypted_password VARCHAR NOT NULL,  
  created_at TIMESTAMP DEFAULT NOW(),  
  last_sign_in_at TIMESTAMP,  
  user_metadata JSONB  
);
```

4.2 Reports Table

```
CREATE TABLE reports (  
  id BIGSERIAL PRIMARY KEY,  
  user_id UUID REFERENCES auth.users(id),  
  user_email VARCHAR NOT NULL,  
  type VARCHAR(50) NOT NULL,  
  location JSONB NOT NULL,  
  description TEXT,  
  photo_url TEXT,  
  upvotes INTEGER DEFAULT 0,  
  status VARCHAR(20) DEFAULT 'new',  
  created_at TIMESTAMP DEFAULT NOW(),  
  updated_at TIMESTAMP DEFAULT NOW()  
);
```

4.3 User Statistics View

```
CREATE VIEW user_stats AS
SELECT
    user_id,
    COUNT(*) as total_reports,
    SUM(upvotes) as total_upvotes,
    MIN(created_at) as member_since,
    MAX(updated_at) as last_activity
FROM reports
GROUP BY user_id;
```

5. API Integration

5.1 Supabase API Operations

Create Report:

```
async function submitReport(reportData) {
    const { data, error } = await supabase
        .from('reports')
        .insert([
            {
                user_id: user.id,
                user_email: user.email,
                type: reportData.type,
                location: reportData.location,
                description: reportData.description,
                photo_url: reportData.photo
            }
        ]);
    return { data, error };
}
```

Fetch Reports:

```
async function getReports() {
    const { data, error } = await supabase
        .from('reports')
        .select('*')
        .order('created_at', { ascending: false });
    return { data, error };
}
```

Update Upvotes:

```
async function upvoteReport(reportId) {
  const { data, error } = await supabase
    .rpc('increment_upvotes', { report_id: reportId });
  return { data, error };
}
```

5.2 EmailUS Integration

```
async function sendNotificationEmail(reportData) {
  const templateParams = {
    user_name: user.name,
    report_type: reportData.type,
    report_location: reportData.location.address,
    report_id: reportData.id
  };

  return await emailjs.send(
    'SERVICE_ID',
    'TEMPLATE_ID',
    templateParams
  );
}
```

6. Security & Privacy

6.1 Security Measures

- **Authentication:** Supabase Auth with encrypted passwords
- **Session Management:** Secure token-based authentication
- **Data Encryption:** HTTPS for all communications
- **Input Validation:** Client and server-side validation
- **SQL Injection Protection:** Parameterized queries

6.2 Privacy Considerations

- User email addresses stored securely
- Optional location sharing
- No personal data shared without consent
- GDPR-compliant data handling
- User-controlled data export and deletion

7. Progressive Web App (PWA)

7.1 PWA Features

Installability:

- Add to home screen on mobile devices
- Desktop installation support
- Native app-like experience

Offline Functionality:

- Service Worker caching
- Offline page display
- Background sync preparation

Manifest Configuration:

```
{  
  "name": "SafeCity Infrastructure Reporter",  
  "short_name": "SafeCity",  
  "start_url": "/",  
  "display": "standalone",  
  "theme_color": "#667eea",  
  "background_color": "#ffffff"  
}
```

8. User Interface Design

8.1 Design Principles

- **Mobile-First:** Optimized for mobile devices
- **Accessibility:** WCAG 2.1 AA compliant
- **Responsive:** Breakpoints at 480px, 768px, 1024px
- **Intuitive:** Clear navigation and feedback
- **Modern:** Gradient backgrounds and smooth animations

8.2 Theme System

Dual Theme Support:

- Light Mode: Clean, bright interface
- Dark Mode: Eye-friendly low-light design
- System preference detection
- User preference persistence

Theme Variables:

```
/* Light Mode */
--bg-primary: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
--text-primary: #333333;
--card-bg: #ffffff;

/* Dark Mode */
--bg-primary: linear-gradient(135deg, #2c3e50 0%, #34495e 100%);
--text-primary: #e0e0e0;
--card-bg: #2d2d2d;
```

9. Performance Optimization

9.1 Loading Performance

- Lazy loading for images
- Minified CSS and JavaScript
- Compressed assets
- Browser caching strategy

9.2 Runtime Performance

- Debounced search functionality
 - Efficient DOM manipulation
 - Optimized re-rendering
 - Memory leak prevention
-

10. Deployment & Installation

10.1 Local Development

Clone repository

```
git clone https://github.com/Uwami-Mgxekwa/SafeCityWeb.git
```

```
cd SafeCityWeb
```

Start local server

```
python -m http.server 8000
```

Open in browser

```
http://localhost:8000
```

10.2 Production Deployment

Requirements:

- Modern web server (Apache, Nginx)
- HTTPS certificate (required for PWA)
- Supabase account and project
- EmailJS account (optional)

Deployment Steps:

1. Configure Supabase credentials
2. Set up EmailJS service
3. Update API keys in configuration
4. Deploy to web server
5. Test PWA installation

11. Future Enhancements

11.1 Planned Features

- **Multi-language Support:** Zulu, Afrikaans, Sotho translations
- **Native Mobile Apps:** iOS and Android applications

- **AI Classification:** Automatic issue categorization from photos
- **Push Notifications:** Real-time status updates
- **Analytics Dashboard:** Advanced reporting for officials
- **Government Integration:** Connect with city management systems

11.2 Scalability Considerations

- Database optimization for high traffic
 - CDN integration for asset delivery
 - Load balancing for API requests
 - Caching strategies for frequently accessed data
-

12. Support & Maintenance

12.1 Support Channels

- **GitHub Issues:** Bug reports and feature requests
- **Documentation:** Comprehensive README and guides
- **Email Support:** Developer contact available
- **Community Forum:** GitHub Discussions

12.2 Maintenance Schedule

- **Regular Updates:** Security patches and bug fixes
 - **Feature Releases:** Quarterly major updates
 - **Database Backups:** Daily automated backups
 - **Performance Monitoring:** Continuous system health checks
-

13. Conclusion

SafeCity represents a comprehensive solution for civic engagement and infrastructure management. By combining modern web technologies with user-centered design, the platform empowers citizens to actively participate in improving their communities while providing authorities with valuable data for prioritizing repairs and maintenance.

The system's modular architecture, robust security measures, and scalable design ensure it can grow with community needs while maintaining performance and reliability.

Document Version: 1.0

Last Updated: October 2025

Author: Uwami Mgxekwa

License: MIT License

For more information, visit: github.com/Uwami-Mgxekwa/SafeCityWeb