# Database Access Information

## Stormwater Infrastructure Assessment Project

### Database Type

The project currently uses SQLite, a file-based database system. For future scalability, we recommend migrating to a cloud-based solution such as MongoDB Atlas or PostgreSQL.

### Database Schema

#### Users Table

* id (INTEGER, PRIMARY KEY): Unique user identifier
* username (TEXT): User’s login name
* password\_hash (TEXT): Bcrypt hashed password
* is\_admin (BOOLEAN): Administrator status
* created\_at (TIMESTAMP): Account creation timestamp

#### Projects Table

* id (INTEGER, PRIMARY KEY): Unique project identifier
* name (TEXT): Project name
* description (TEXT): Project description
* created\_by (INTEGER, FOREIGN KEY): User ID of creator
* created\_at (TIMESTAMP): Project creation timestamp

#### Project\_Members Table

* id (INTEGER, PRIMARY KEY): Unique member entry identifier
* project\_id (INTEGER, FOREIGN KEY): Project identifier
* user\_id (INTEGER, FOREIGN KEY): User identifier
* role (TEXT): Role within project (admin, member, viewer)
* joined\_at (TIMESTAMP): Timestamp of joining project

#### Assessments Table

* id (INTEGER, PRIMARY KEY): Unique assessment identifier
* user\_id (INTEGER, FOREIGN KEY): Creator of assessment
* project\_id (INTEGER, FOREIGN KEY): Associated project
* data (JSON): Assessment data in JSON format
* timestamp (TIMESTAMP): Assessment creation timestamp

### Assessment Data Structure

The assessment data is stored as a JSON object with the following structure:

{  
 "condition": {  
 "stormwaterHydraulicAssetCondition": {  
 "pipeDiameterWidth": "600mm",  
 "pipeScore": 7,  
 "routineInspectionFrequency": "Quarterly",  
 "damageLevels": {  
 "cracks": {  
 "condition": "low",  
 "score": 8  
 },  
 "sediment\_build\_up": {  
 "condition": "moderate",  
 "score": 5  
 },  
 "corrosion": {  
 "condition": "low",  
 "score": 7  
 }  
 }  
 },  
 "OSAC": {  
 "score": 7,  
 "rating": "Good"  
 }  
 },  
 "functionality": {  
 "hydraulicPerformance": {  
 "flowAttenuation": 69,  
 "volumeReduction": 73,  
 "csoVolume": 250,  
 "totalFlowVolume": 1000,  
 "overflowPercentage": 25.0,  
 "dryWeatherFlow": 82,  
 "overflowFrequency": 65,  
 "drainageDurationPerformance": 70  
 },  
 "hydrologicalPerformance": {  
 "runoffFrequency": 65,  
 "baseFlowPerformance": 70,  
 "inflowPerformance": 80.0,  
 "catchmentScalePerformance": 85  
 },  
 "overallFunctionality": {  
 "score": 8,  
 "rating": "Good"  
 }  
 },  
 "time\_effectiveness": {  
 "longTermFunctionality": 60,  
 "designLifespan": 50,  
 "currentAge": 15,  
 "agePercentage": 30.0,  
 "lagTimePerformance": 80,  
 "maintenanceResponseTarget": 14,  
 "actualResponseTime": 18,  
 "monitoringFrequency": "Weekly",  
 "overallTimeScore": 7,  
 "overallTimeRating": "Good"  
 },  
 "cost\_effectiveness": {  
 "preliminaryCosts": 75000,  
 "constructionCosts": 1250000,  
 "annualOperationalCosts": 52000,  
 "lifecyclePeriod": 30,  
 "discountRate": 4.5,  
 "totalLifecycleCost": 1885000.0,  
 "annualFloodDamagePrevention": 95000,  
 "annualEnvironmentalBenefits": 45000,  
 "totalBenefits": 4200000.0,  
 "benefitCostRatio": 2.2,  
 "roi": 122.8,  
 "overallCostScore": 8,  
 "overallCostRating": "Good"  
 },  
 "environmental\_social": {  
 "pollutantConcentrationAttenuation": 70,  
 "eventBasedPollutantRemoval": 55,  
 "pollutionRetentionPerformance": 66,  
 "tssReduction": 75,  
 "nutrientReduction": 60,  
 "bacteriaReduction": 68,  
 "customerSatisfaction": 50,  
 "communityEngagement": "Moderate",  
 "accidentFrequency": 2,  
 "accidentSeverity": "Minor",  
 "environmentalScore": 7.0,  
 "socialScore": 6.0,  
 "overallScore": 7,  
 "overallRating": "Good"  
 },  
 "infrastructure\_points": [  
 {  
 "name": "Main Outfall",  
 "type": "outfalls",  
 "latitude": 40.7128,  
 "longitude": -74.006,  
 "age": 12,  
 "last\_maintenance\_days": 45  
 },  
 {  
 "name": "Retention Basin North",  
 "type": "bioRetentionBasins",  
 "latitude": 40.7135,  
 "longitude": -74.0052,  
 "age": 5,  
 "last\_maintenance\_days": 120  
 }  
 ],  
 "timestamp": "2025-04-07T14:30:00"  
}

### Access Instructions for Developers

#### Local Development

1. The SQLite database file is located at: stormwater\_assessment.db
2. Use SQL queries or ORM (Object-Relational Mapping) to interact with the database
3. For Python access, use the utilities in utils/db.py

#### Recommended Migration Steps

1. Create a cloud database (PostgreSQL or MongoDB)
2. Execute migration scripts (provided in utils/db\_migration.py)
3. Update connection strings in the application
4. Verify data integrity after migration

### Authentication Information

Default database access credentials:

* **Regular User Access**
  + Username: user
  + Password: password123
* **Admin Access**
  + Username: admin
  + Password: admin123

### Database Backup Information

* Current backup schedule: Daily at 2:00 AM
* Backup location: backups/ directory
* Backup naming convention: stormwater\_assessment\_YYYY-MM-DD.db

### API Integration Information

For AI model integration, data can be accessed through the following functions in utils/db.py:

* get\_assessments(): Retrieve assessment data
* get\_infrastructure\_points(): Get geospatial data
* get\_historical\_metrics(): Get time-series metrics

Sample code for AI model integration:

from utils.db import get\_assessments  
  
# Get all assessments for a specific project  
assessments = get\_assessments(project\_id=12)  
  
# Extract features for model training  
features = []  
labels = []  
  
for assessment in assessments:  
 # Extract relevant features from assessment data  
 data = assessment['data']  
   
 # Example: Predicting condition score based on infrastructure parameters  
 features.append([  
 data['time\_effectiveness']['currentAge'],  
 data['time\_effectiveness']['designLifespan'],  
 data['functionality']['hydraulicPerformance']['flowAttenuation'],  
 data['functionality']['hydraulicPerformance']['volumeReduction']  
 ])  
   
 # Target label is the condition score  
 labels.append(data['condition']['OSAC']['score'])  
  
# Train ML model with these features and labels  
# model.fit(features, labels)

This document should be updated when migrating to a cloud-based database solution.