🌱 Interactive Soil Health Monitoring System

A production-ready, full-stack application for soil health monitoring and analysis with JWT authentication, real-time mapping, and personalized fertilizer recommendations.

✨ Features

Core Features

- 🔐 **Secure Authentication**: JWT-based authentication with bcrypt password hashing
- 🗺️ **Interactive Maps**: Leaflet-based soil health visualization with NPK data
- 📊 **Real-time Analytics**: District-wise soil health dashboard with Chart.js
- 🌾 **Crop Recommendations**: Personalized fertilizer recommendations based on soil tests
- 📱 **Responsive Design**: Modern, mobile-first UI with stunning animations
- 🔗 **API Integration**: Ready for data.gov.in API integration

Technical Features

- Micronaut 4.x framework for high performance
- PostgreSQL with PostGIS for geospatial data
- JWT token-based authentication
- RESTful API architecture
- Production-ready configuration
- Docker support

🚀 Quick Start

Prerequisites

- Java 17 or higher
- PostgreSQL 14+ with PostGIS extension
- Gradle 8.x
- Node.js (optional, for frontend development)

Installation

1. **Clone the repository**

```
git clone https://github.com/zenzxnse/soil-health-monitor.git
cd soil-health-monitor
```

2. **Set up PostgreSQL**

```
# Create database
createdb ishm
```

```
# Enable PostGIS
psql ishm -c "CREATE EXTENSION IF NOT EXISTS postgis;"

# Run schema
psql ishm < src/main/resources/schema-enhanced.sql
```

3. **Configure environment variables**

```
# Create .env file
cat > .env << EOF
DB_URL=jdbc:postgresql://localhost:5432/ishm
DB_USER=your_db_user
DB_PASSWORD=your_db_password
JWT_SECRET=your-super-secret-jwt-key-change-this-in-production-min-256-bits
DATA_GOV_API_KEY=your_data_gov_api_key_here
SOIL_HEALTH_RESOURCE_ID=your_resource_id
EOF
```

4. **Build and run**

```
# Build the application
./gradlew clean build

# Run the application
./gradlew run

# Or run the JAR
java -jar build/libs/soil-health-monitor-1.0.0.jar
```

5. **Access the application**

```
Open browser: http://localhost:8080
```

🔧 Configuration

Environment Variables

Variable	Description	Default
`DB_URL`	PostgreSQL connection URL	`jdbc:postgresql://localhost:5432/ishm`
`DB_USER`	Database username	`ishm`

Variable	Description	Default
`DB_PASSWORD`	Database password	`ishm`
`JWT_SECRET`	JWT signing secret (min 256 bits)	`changeMeInProduction...`
`JWT_EXPIRATION`	Token expiration in seconds	`86400` (24 hours)
`DATA_GOV_API_KEY`	Data.gov.in API key	-
`SOIL_HEALTH_RESOURCE_ID`	Soil health resource ID	-
`CORS_ALLOWED_ORIGINS`	Allowed CORS origins	`*`
`LOG_LEVEL_APP`	Application log level	`INFO`

Application Profiles

- **development**: Debug logging, relaxed CORS
- **production**: Optimized for production, strict security
- **docker**: Configured for Docker deployment
- **test**: Test configuration with in-memory database

Activate profile:

```
MICRONAUT_ENVIRONMENTS=production java -jar app.jar
```

API Documentation

Authentication Endpoints

Register New Farmer

```
POST /api/auth/register
Content-Type: application/json

{
  "username": "farmer123",
  "password": "secure_password",
  "postalCode": "110001",
  "fullName": "John Farmer",
  "phone": "9876543210"
}

Response 200 OK:
{
```

```
  "success": true,
  "message": "Registration successful",
  "farmer": {
    "id": 1,
    "username": "farmer123",
    "postalCode": "110001",
    "district": "Delhi",
    "state": "Delhi"
  }
}
```

Login

```
POST /api/auth/login
Content-Type: application/json

{
  "username": "farmer123",
  "password": "secure_password"
}

Response 200 OK:
{
  "success": true,
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "farmer": {
    "id": 1,
    "username": "farmer123",
    "district": "Delhi",
    "state": "Delhi"
  }
}
```

Map Endpoints

Get All Districts (GeoJSON)

```
GET /api/map/districts?state=Delhi

Response 200 OK:
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": { ... },
      "properties": {
```

```
        "district_name": "Delhi",
        "state_name": "Delhi",
        "nitrogen_avg": 110,
        "nitrogen_status": "Low",
        "phosphorus_avg": 18,
        "phosphorus_status": "Medium",
        "potassium_avg": 300,
        "potassium_status": "High"
      }
    }
  ]
}
```

Get State Statistics

```
GET /api/map/stats/Delhi

Response 200 OK:
{
  "state": "Delhi",
  "district_count": 11,
  "avg_nitrogen": 110.5,
  "avg_phosphorus": 18.2,
  "avg_potassium": 300.1,
  "total_samples": 15234,
  "npk_distribution": {
    "nitrogen": { "low": 8, "medium": 2, "high": 1 },
    "phosphorus": { "low": 3, "medium": 6, "high": 2 },
    "potassium": { "low": 1, "medium": 5, "high": 5 }
  }
}
```

Recommendation Endpoints

Calculate Fertilizer Recommendations

```
POST /api/recommendations/calculate
Content-Type: application/json

{
  "state": "Delhi",
  "district": "Delhi",
  "crop": "wheat",
  "season": "rabi",
  "nitrogen": 250,
  "phosphorus": 15,
  "potassium": 180,
```

```
  "ph": 7.2
}

Response 200 OK:
{
  "nitrogenStatus": "Low",
  "phosphorusStatus": "Medium",
  "potassiumStatus": "Medium",
  "ureaDose": 150.5,
  "dapDose": 100.2,
  "mopDose": 80.0,
  "schedule": {
    "basal": "Apply 50% N, full P and K at sowing",
    "firstTopdress": "30-35 days after sowing",
    "secondTopdress": "60-65 days after sowing"
  },
  "tips": [
    "Apply fertilizers when soil has adequate moisture",
    "Avoid application during heavy rain"
  ]
}
```

Dashboard Endpoints

Get Dashboard Summary

```
GET /api/dashboard/summary?state=Delhi&year=2025
Authorization: Bearer <token>

Response 200 OK:
{
  "metrics": {
    "districtsCovered": 127,
    "totalSamples": 4150000,
    "avgSoilHealth": 6.8,
    "farmersBenefited": 2300000
  },
  "npkTrends": { ... },
  "stateDistribution": [ ... ],
  "districtSummary": [ ... ]
}
```

🐳 Docker Deployment

Using Docker Compose

```yaml
version: '3.8'

services:
  db:
    image: postgis/postgis:14-3.3
    environment:
      POSTGRES_DB: ishm
      POSTGRES_USER: ishm
      POSTGRES_PASSWORD: ishm
    volumes:
      - postgres_data:/var/lib/postgresql/data
      - ./schema-enhanced.sql:/docker-entrypoint-initdb.d/init.sql
    ports:
      - "5432:5432"

  app:
    build: .
    ports:
      - "8080:8080"
    environment:
      DB_URL: jdbc:postgresql://db:5432/ishm
      DB_USER: ishm
      DB_PASSWORD: ishm
      JWT_SECRET: ${JWT_SECRET}
      MICRONAUT_ENVIRONMENTS: docker
    depends_on:
      - db

volumes:
  postgres_data:
```

Run:

```
docker-compose up -d
```

Building Docker Image

```
./gradlew dockerBuild
docker run -p 8080:8080 soil-health-monitor:1.0.0
```

Security Best Practices

1. **Change Default JWT Secret**: Always use a strong, random secret in production
2. **Use HTTPS**: Enable SSL/TLS in production environments
3. **Secure Database**: Use strong passwords and restrict database access

4. **Environment Variables**: Never commit secrets to version control
5. **Rate Limiting**: Implement rate limiting for API endpoints
6. **Input Validation**: All inputs are validated server-side
7. **CORS**: Configure allowed origins properly

📊 Database Schema

Key Tables

- **farmers**: User authentication and profile data
- **districts**: Geographic boundaries with PostGIS geometry
- **soil_health_data**: NPK levels and soil parameters
- **crop_recommendations**: Crop-specific nutrient requirements
- **states**: State-level geographic data

Indexes

- Geographic indexes (GiST) for spatial queries
- B-tree indexes on foreign keys and frequently queried columns
- Composite indexes for common query patterns

🧪 Testing

```
# Run all tests
./gradlew test

# Run specific test class
./gradlew test --tests AuthControllerTest

# Run with coverage
./gradlew test jacocoTestReport
```

📈 Performance Optimization

- **Database**: Connection pooling with HikariCP (max 50 connections)
- **Caching**: TTL-based caching for district and statistics data
- **Lazy Loading**: Efficient data fetching with JOIN queries
- **CDN**: Serve static assets via CDN in production
- **Compression**: Enable GZIP compression for API responses

🌍 Data.gov.in Integration

To integrate with India's Open Data platform:

1. Register at https://data.gov.in
2. Get API key and resource ID for soil health data
3. Set environment variables:

```
DATA_GOV_API_KEY=your_api_key
SOIL_HEALTH_RESOURCE_ID=resource_id
DATA_GOV_API_ENABLED=true
```

4. The system will automatically enrich local data with government datasets

🛠️ Troubleshooting

Common Issues

Issue: Database connection failed

```
# Check PostgreSQL is running
systemctl status postgresql

# Verify connection
psql -U ishm -d ishm -h localhost
```

Issue: JWT token invalid

- Ensure JWT_SECRET is set and consistent across restarts
- Check token expiration time
- Verify Authorization header format: `Bearer <token>`

Issue: Map not loading

- Check browser console for errors
- Verify API endpoints are accessible
- Ensure database has district geometry data

📝 License

This project is licensed under the MIT License - see LICENSE file for details.

🤝 Contributing

1. Fork the repository
2. Create a feature branch (`git checkout -b feature/amazing-feature`)
3. Commit your changes (`git commit -m 'Add amazing feature'`)
4. Push to branch (`git push origin feature/amazing-feature`)
5. Open a Pull Request

📧 Support

For support and queries:

- Email: support@soilhealth.gov.in

- Issues: GitHub Issues
- Documentation: https://docs.soilhealth.gov.in

🙏 Acknowledgments

- Ministry of Agriculture & Farmers Welfare, Government of India
- OpenStreetMap for map tiles
- data.gov.in for open datasets
- PostgreSQL and PostGIS communities

Built with 💚 for Indian farmers# 🌱 Interactive Soil Health Monitoring System

A production-ready, full-stack application for soil health monitoring and analysis with JWT authentication, real-time mapping, and personalized fertilizer recommendations.

✨ Features

Core Features

- 🔐 **Secure Authentication**: JWT-based authentication with bcrypt password hashing
- 🗺️ **Interactive Maps**: Leaflet-based soil health visualization with NPK data
- 📊 **Real-time Analytics**: District-wise soil health dashboard with Chart.js
- 🌾 **Crop Recommendations**: Personalized fertilizer recommendations based on soil tests
- 📱 **Responsive Design**: Modern, mobile-first UI with stunning animations
- 🔗 **API Integration**: Ready for data.gov.in API integration

Technical Features

- Micronaut 4.x framework for high performance
- PostgreSQL with PostGIS for geospatial data
- JWT token-based authentication
- RESTful API architecture
- Production-ready configuration
- Docker support

🚀 Quick Start

Prerequisites

- Java 17 or higher
- PostgreSQL 14+ with PostGIS extension
- Gradle 8.x
- Node.js (optional, for frontend development)

Installation

1. **Clone the repository**

```
git clone https://github.com/yourusername/soil-health-monitor.git
cd soil-health-monitor
```

2. **Set up PostgreSQL**

```
# Create database
createdb ishm

# Enable PostGIS
psql ishm -c "CREATE EXTENSION IF NOT EXISTS postgis;"

# Run schema
psql ishm < src/main/resources/schema-enhanced.sql
```

3. **Configure environment variables**

```
# Create .env file
cat > .env << EOF
DB_URL=jdbc:postgresql://localhost:5432/ishm
DB_USER=your_db_user
DB_PASSWORD=your_db_password
JWT_SECRET=your-super-secret-jwt-key-change-this-in-production-min-256-bits
DATA_GOV_API_KEY=your_data_gov_api_key_here
SOIL_HEALTH_RESOURCE_ID=your_resource_id
EOF
```

4. **Build and run**

```
# Build the application
./gradlew clean build

# Run the application
./gradlew run

# Or run the JAR
java -jar build/libs/soil-health-monitor-1.0.0.jar
```

5. **Access the application**

```
Open browser: http://localhost:8080
```

🔧 Configuration

Environment Variables

Variable	Description	Default
`DB_URL`	PostgreSQL connection URL	`jdbc:postgresql://localhost:5432/ishm`
`DB_USER`	Database username	`ishm`
`DB_PASSWORD`	Database password	`ishm`
`JWT_SECRET`	JWT signing secret (min 256 bits)	`changeMeInProduction...`
`JWT_EXPIRATION`	Token expiration in seconds	`86400` (24 hours)
`DATA_GOV_API_KEY`	Data.gov.in API key	-
`SOIL_HEALTH_RESOURCE_ID`	Soil health resource ID	-
`CORS_ALLOWED_ORIGINS`	Allowed CORS origins	`*`
`LOG_LEVEL_APP`	Application log level	`INFO`

Application Profiles

- **development**: Debug logging, relaxed CORS
- **production**: Optimized for production, strict security
- **docker**: Configured for Docker deployment
- **test**: Test configuration with in-memory database

Activate profile:

```
MICRONAUT_ENVIRONMENTS=production java -jar app.jar
```

API Documentation

Authentication Endpoints

Register New Farmer

```
POST /api/auth/register
Content-Type: application/json

{
  "username": "farmer123",
  "password": "secure_password",
```

```
  "postalCode": "110001",
  "fullName": "John Farmer",
  "phone": "9876543210"
}

Response 200 OK:
{
  "success": true,
  "message": "Registration successful",
  "farmer": {
    "id": 1,
    "username": "farmer123",
    "postalCode": "110001",
    "district": "Delhi",
    "state": "Delhi"
  }
}
```

Login

```
POST /api/auth/login
Content-Type: application/json

{
  "username": "farmer123",
  "password": "secure_password"
}

Response 200 OK:
{
  "success": true,
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "farmer": {
    "id": 1,
    "username": "farmer123",
    "district": "Delhi",
    "state": "Delhi"
  }
}
```

Map Endpoints

Get All Districts (GeoJSON)

```
GET /api/map/districts?state=Delhi

Response 200 OK:
```

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": { ... },
      "properties": {
        "district_name": "Delhi",
        "state_name": "Delhi",
        "nitrogen_avg": 110,
        "nitrogen_status": "Low",
        "phosphorus_avg": 18,
        "phosphorus_status": "Medium",
        "potassium_avg": 300,
        "potassium_status": "High"
      }
    }
  ]
}
```

Get State Statistics

```
GET /api/map/stats/Delhi

Response 200 OK:
{
  "state": "Delhi",
  "district_count": 11,
  "avg_nitrogen": 110.5,
  "avg_phosphorus": 18.2,
  "avg_potassium": 300.1,
  "total_samples": 15234,
  "npk_distribution": {
    "nitrogen": { "low": 8, "medium": 2, "high": 1 },
    "phosphorus": { "low": 3, "medium": 6, "high": 2 },
    "potassium": { "low": 1, "medium": 5, "high": 5 }
  }
}
```

Recommendation Endpoints

Calculate Fertilizer Recommendations

```
POST /api/recommendations/calculate
Content-Type: application/json

{
```

```
  "state": "Delhi",
  "district": "Delhi",
  "crop": "wheat",
  "season": "rabi",
  "nitrogen": 250,
  "phosphorus": 15,
  "potassium": 180,
  "ph": 7.2
}

Response 200 OK:
{
  "nitrogenStatus": "Low",
  "phosphorusStatus": "Medium",
  "potassiumStatus": "Medium",
  "ureaDose": 150.5,
  "dapDose": 100.2,
  "mopDose": 80.0,
  "schedule": {
    "basal": "Apply 50% N, full P and K at sowing",
    "firstTopdress": "30-35 days after sowing",
    "secondTopdress": "60-65 days after sowing"
  },
  "tips": [
    "Apply fertilizers when soil has adequate moisture",
    "Avoid application during heavy rain"
  ]
}
```

Dashboard Endpoints

Get Dashboard Summary

```
GET /api/dashboard/summary?state=Delhi&year=2025
Authorization: Bearer <token>

Response 200 OK:
{
  "metrics": {
    "districtsCovered": 127,
    "totalSamples": 4150000,
    "avgSoilHealth": 6.8,
    "farmersBenefited": 2300000
  },
  "npkTrends": { ... },
  "stateDistribution": [ ... ],
  "districtSummary": [ ... ]
}
```

Docker Deployment

Using Docker Compose

```
version: '3.8'

services:
  db:
    image: postgis/postgis:14-3.3
    environment:
      POSTGRES_DB: ishm
      POSTGRES_USER: ishm
      POSTGRES_PASSWORD: ishm
    volumes:
      - postgres_data:/var/lib/postgresql/data
      - ./schema-enhanced.sql:/docker-entrypoint-initdb.d/init.sql
    ports:
      - "5432:5432"

  app:
    build: .
    ports:
      - "8080:8080"
    environment:
      DB_URL: jdbc:postgresql://db:5432/ishm
      DB_USER: ishm
      DB_PASSWORD: ishm
      JWT_SECRET: ${JWT_SECRET}
      MICRONAUT_ENVIRONMENTS: docker
    depends_on:
      - db

volumes:
  postgres_data:
```

Run:

```
docker-compose up -d
```

Building Docker Image

```
./gradlew dockerBuild
docker run -p 8080:8080 soil-health-monitor:1.0.0
```

Security Best Practices

1. **Change Default JWT Secret**: Always use a strong, random secret in production
2. **Use HTTPS**: Enable SSL/TLS in production environments
3. **Secure Database**: Use strong passwords and restrict database access
4. **Environment Variables**: Never commit secrets to version control
5. **Rate Limiting**: Implement rate limiting for API endpoints
6. **Input Validation**: All inputs are validated server-side
7. **CORS**: Configure allowed origins properly

📊 Database Schema

Key Tables

- **farmers**: User authentication and profile data
- **districts**: Geographic boundaries with PostGIS geometry
- **soil_health_data**: NPK levels and soil parameters
- **crop_recommendations**: Crop-specific nutrient requirements
- **states**: State-level geographic data

Indexes

- Geographic indexes (GiST) for spatial queries
- B-tree indexes on foreign keys and frequently queried columns
- Composite indexes for common query patterns

🧪 Testing

```
# Run all tests
./gradlew test

# Run specific test class
./gradlew test --tests AuthControllerTest

# Run with coverage
./gradlew test jacocoTestReport
```

📈 Performance Optimization

- **Database**: Connection pooling with HikariCP (max 50 connections)
- **Caching**: TTL-based caching for district and statistics data
- **Lazy Loading**: Efficient data fetching with JOIN queries
- **CDN**: Serve static assets via CDN in production
- **Compression**: Enable GZIP compression for API responses

🌍 Data.gov.in Integration

To integrate with India's Open Data platform:

1. Register at https://data.gov.in
2. Get API key and resource ID for soil health data
3. Set environment variables:

```
DATA_GOV_API_KEY=your_api_key
SOIL_HEALTH_RESOURCE_ID=resource_id
DATA_GOV_API_ENABLED=true
```

4. The system will automatically enrich local data with government datasets

🛠️ Troubleshooting

Common Issues

Issue: Database connection failed

```
# Check PostgreSQL is running
systemctl status postgresql

# Verify connection
psql -U ishm -d ishm -h localhost
```

Issue: JWT token invalid

- Ensure JWT_SECRET is set and consistent across restarts
- Check token expiration time
- Verify Authorization header format: `Bearer <token>`

Issue: Map not loading

- Check browser console for errors
- Verify API endpoints are accessible
- Ensure database has district geometry data

📝 License

This project is licensed under the MIT License - see LICENSE file for details.

🤝 Contributing

1. Fork the repository
2. Create a feature branch (`git checkout -b feature/amazing-feature`)
3. Commit your changes (`git commit -m 'Add amazing feature'`)
4. Push to branch (`git push origin feature/amazing-feature`)
5. Open a Pull Request

📪 Support

For support and queries:

- Email: support@soilhealth.gov.in
- Issues: GitHub Issues
- Documentation: https://docs.soilhealth.gov.in

🙏 Acknowledgments

- Ministry of Agriculture & Farmers Welfare, Government of India
- OpenStreetMap for map tiles
- data.gov.in for open datasets
- PostgreSQL and PostGIS communities

Built with 💚 for Indian farmers