Software Release Document (SRD)

This document aims to guide users in installing and running the Hazard Maps Project application and to outline the known limitations of the software. The project uses Python, Flask, GeoServer, and PostgreSQL with the PostGIS extension to provide hazard maps for disaster management.

Hazard Maps Project

This project aims to provide hazard maps for disaster management using Python, Flask,GeoServer, and PostgreSQL with PostGIS extension. The following instructions will guide you through setting up and running the project.

Prerequisites

- Python 3.8+
- PostgreSQL 12+
- Node.js and npm (for frontend dependencies)
- pgAdmin (for database management)
- GeoServer (for map layer publishing)

Setup Instructions

1. **Clone the repository**

   First, clone the repository to your local machine:

   ```
   git clone https://github.com/tanxiaoo/Hazard_Maps_Project-SE4GEO.git
   cd Hazard_Maps_Project-SE4GEO/APP
   ```

2. **Download large files**

   Due to GitHub's file size limitations, the `spatial_data`folders contain large files that need to be downloaded separately. Please download the necessary files from the following links and place them in the `backend/resources/geoserver/spatial_data`folders:

   - Download spatial_data folder

3. **Create the database and enable the PostGIS extension**

   In pgAdmin:

   1. Open pgAdmin and connect to your PostgreSQL server.
   2. Create a database named `se4g24`.
   3. Select the newly created database, open the query tool, and run the following command to enable the PostGIS extension:

```
    ```sql
    CREATE EXTENSION postgis;
    ```
4. Run the `fill_database.py` file to populate the database:
    ```sh
    cd backend
    python fill_database.py
    ```
```

4. **Configure GeoServer**

    1. Download and install GeoServer:

        - Visit the GeoServer download page and download the latest version of GeoServer.
        - Follow the installation instructions.

    2. Start GeoServer and access the GeoServer Web Interface at:

        ```
        http://localhost:8080/geoserver
        ```

    3. Publish layers and styles:

        - In the left menu, select `Data` -> `Layers`.
        - Click `Add new Layer`, select the data store, and choose the data from `backend/resources/geoserver/spatial_data` folder, then follow the prompts to complete the layer publishing.
        - In the left menu, select `Style` -> `SLD`.
        - Click `Add a new style`, upload the style files from `backend/resources/geoserver/SLD` directory, and apply them to the corresponding layers.

5. **Create a .env file**

    Create a `.env` file in the `APP` directory and add the following content:

    ```
    DATABASE_URL=postgresql://postgres:yourpassword@localhost:5432/se4g24
    GEOSERVER_URL=http://localhost:8080/geoserver
    ```

    Ensure to replace `yourpassword` with your PostgreSQL password.

6. **Install Python dependencies**

    Ensure you are in the `APP` directory, then run the following commands to create and activate a virtual environment and install the required Python packages:

```
python -m venv venv
source venv/bin/activate  # For Windows, use `venv\Scripts\activate`
pip install -r requirements.txt
```

7. **Install Node.js dependencies**

   In the `APP` directory, install the Node.js dependencies:

   ```
   npm install
   ```

8. **Run the Flask application**

   In the `APP` directory, run the Flask application:

   ```
   cd backend
   python app.py
   ```

9. **Run the frontend dashboard**

   In the `APP` directory, run the frontend dashboard:

   ```
   cd frontend
   python dashboard.py
   ```

Follow these steps to correctly set up and run the project. If you encounter any issues during the process, please refer to the project documentation or contact the project development team for assistance.