Design Document

HAZARD MAPS PROJECT SE4GEO IN ITALY
POLITECNICO DI MILANO

AUTHORS
Giovanni Montefoschi
Zhanbin Wu
Xiao Tan

INSTRUCTOR
Giovanni Quattrocchi

DATE PREPARED
31/05/2024

VERSION
0.0.0

Table of content

1. Introduction

1.1 Context and motivations

The Italian public authorities are responsible for collecting, processing, and analyzing datasets related to natural disasters to generate landslide hazard maps. By combining these hazard maps with data on population, households, buildings, industries and services, cultural heritage, and territories across various regions of Italy, a comprehensive assessment of regional exposure to landslides can be conducted. This multidimensional data analysis aids in evaluating the potential impact of landslides on different areas, providing critical information for decision-makers.

The analysis specifically covers the following data aspects:

- **Territory**: Geographical and physical characteristics of the region.
- **Population**: Distribution and density of the population in each region.
- **Families**: Structure and distribution of households.
- **Buildings**: Number and types of buildings.
- **Industries and Services**: Distribution and economic significance of industries and services.
- **Cultural Heritage**: Distribution and historical and cultural value of heritage sites.

The scope of this project includes both national and regional levels. By integrating these multidimensional datasets, decision-makers can more effectively access, visualize, and analyze relevant information, thereby developing scientific disaster management and prevention measures, enhancing disaster response capabilities and public safety levels.

1.2 Definitions and Acronyms

- **ISPRA**: Istituto Superiore per la Protezione e la Ricerca Ambientale (Italian Institute for Environmental Protection and Research). ISPRA is responsible for environmental protection and research, providing data related to natural disasters.
- **PIR**: Hazards and Risk Indicators. These indicators are used to assess and analyze the risk of natural disasters such as landslides.
- **REST API**: Representational State Transfer Application Programming Interface. An interface design style for web services allowing interaction between clients and servers.
- **GeoJSON**: Geospatial JavaScript Object Notation. An open standard format used for encoding geographic data structures, commonly used in web mapping applications.
- **WMS**: Web Map Service. A standard protocol used to request geographic information services over the internet to generate maps.

- **PostgreSQL**: An open-source object-relational database management system emphasizing extensibility and standards compliance, widely used for complex data analysis tasks.
- **PostGIS**: An extension to PostgreSQL adding support for geographic objects, enabling the handling of geographic information system (GIS) data.
- **Flask**: A lightweight web application framework written in Python, designed for simplicity and flexibility, suitable for rapid development of web services and REST APIs.
- **GeoServer**: An open-source server software allowing users to share and edit geospatial data. It is designed to interact with many different data sources and output data to various clients, including web maps.
- **Pandas**: A Python data analysis library providing high-performance, easy-to-use data structures, and data analysis tools.
- **GeoPandas**: An extension to Pandas supporting geospatial data types, enabling the handling and analysis of geospatial data.

1.3 Purpose

The aim of this project is to develop an interactive web application to support disaster management activities in Italy. The application will integrate PIR data from ISPRA, presenting landslide risk and exposure information related to population, buildings, households, industries and services, cultural heritage, and territories through a user-friendly interface, assisting decision-makers in effective disaster management.

The main interactive features of the application include:

- **Region Selection**: Users can select different regions from a dropdown menu to view detailed statistical information and exposure data for that region, including population, households, buildings, industries and services, cultural heritage, and territories.
- **Map View**: The application provides landslide risk maps showing the distribution of landslide risks across Italian regions. The maps support zooming and panning, allowing users to view detailed risk information for specific areas.
- **Data Display**: Statistical information and landslide risk data for the selected region are presented through data cards and stacked bar charts. Users can clearly see the distribution of different categories of data, such as population, households, buildings, industries and services, and cultural heritage across different risk levels.
- **Data Download**: The application allows users to download statistical information and landslide risk data for specific regions, supporting further analysis and use.
- **Legend and Styles**: The application includes a detailed legend to help users understand various risk symbols and data representations on the map.

- **Map Layer Selection**: Users can select different map layers to view more detailed information. These layers include:
  - OpenStreetMap: Provides a basic map view.
  - Carto Positron: Offers a clear and simple map style suitable for background maps.
  - Esri Topographic: Provides a detailed topographic map view.
  - Esri World Imagery: Offers satellite imagery views.
  - Landslide Hazard Map: Displays the landslide risk map layer, indicating areas at risk of landslides.
  - Population Density: Displays the population density distribution layer.
  - Risk Indicator Buildings: Shows the building risk indicator layer.
  - Risk Indicator Cultural Heritage: Displays the cultural heritage risk indicator layer.
  - Risk Indicator Families: Shows the family risk indicator layer.
  - Risk Indicator Industries and Services: Displays the industries and services risk indicator layer.
  - Risk Indicator Population: Shows the population risk indicator layer.
  - Risk Indicator Territory: Displays the territory risk indicator layer.
  - Regions: Displays region boundary layers.
  - Selected Region: Highlights the selected region.

These interactive features enable users to easily access and analyze relevant information, allowing them to develop scientific disaster management and prevention measures, enhancing disaster response capabilities and public safety levels.

1.4 Scope and Limitations

Scope:

Develop a client-server application comprising a database, REST API, and interactive dashboard. Data will be sourced from the ISPRA IdroGEO API, including landslide risk and census data on population, buildings, households, industries and services, cultural heritage, and territories. The application will support national and regional-level data presentation and analysis.

Limitations:

The current project does not support real-time data integration, though this can be a direction for future development. The design and functionality of the dashboard are limited by the technical capabilities of Dash and related graphical libraries. The analysis primarily focuses on regional levels, which may not sufficiently meet the needs of smaller geographic units (e.g., neighborhoods or communities).

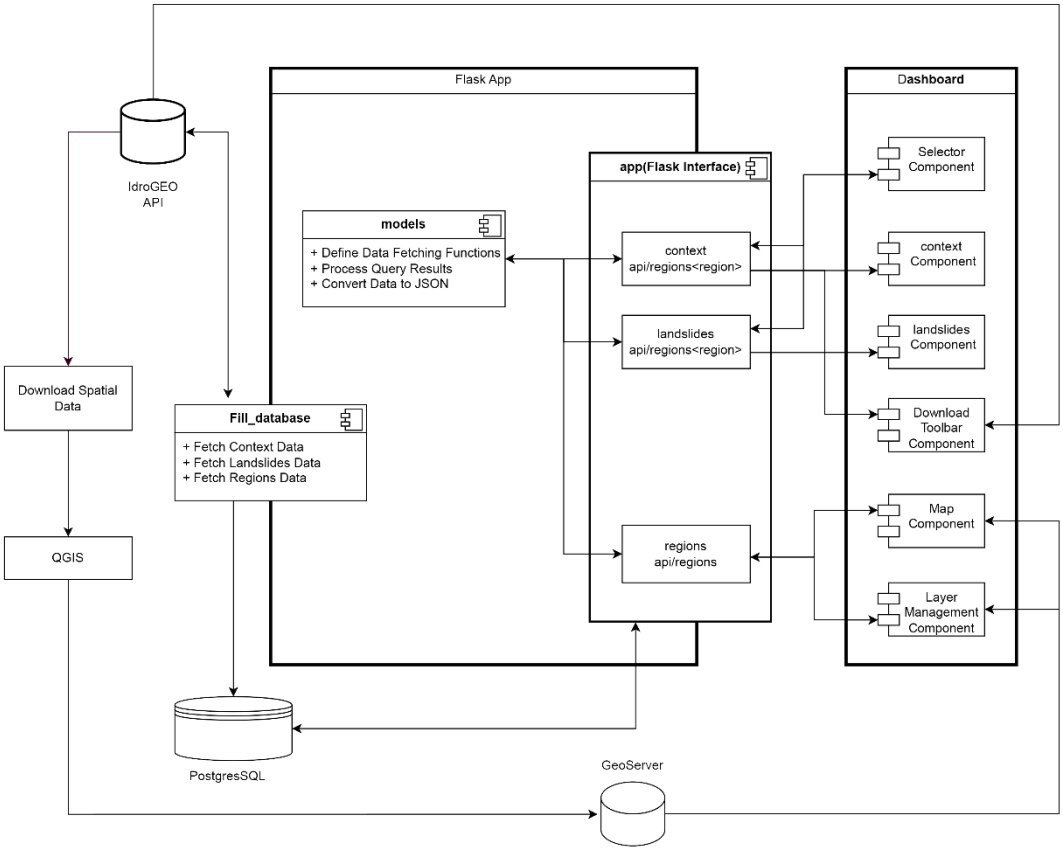
2. Architectural Design

2.1 Overview

The system is composed of three main components: the database, the web server (REST API), and the interactive dashboard (frontend). The database is responsible for storing and managing landslide risk data and related statistical information obtained from the ISPRA IdroGEO API. The web server provides query and retrieval services to the database through REST API interfaces. The interactive dashboard is used for visual data presentation, supporting users in selecting different regions, viewing detailed statistical information and risk analysis, thereby achieving interactive data analysis and decision support.

2.2 Component Diagrams

The component diagram below illustrates the relationship between the database, web server, and dashboard, as well as the interaction and data flow between each component.



Component Description:

- **IdroGEO API**: Provides API interfaces for landslide risk and related statistical data.

    - **Function**: Acquire landslide risk data and related statistical information through HTTP requests from the ISPRA IdroGEO API.
- **Download Spatial Data**: Provides spatial data download.
    - **Function**: Download spatial data from external sources for further geospatial analysis.
- **QGIS**: Processes and converts spatial data.
    - **Function**: Process downloaded spatial data using QGIS and publish these data through GeoServer.
- **Fill_database**: Acquires data from the API and stores it in the database.
    - **Function**: Use the fill_database.py script to acquire statistical data, landslide data, and regional data from the IdroGEO API and store them in the PostgreSQL database.
- **PostgreSQL with PostGIS**: Stores and manages geospatial data.
    - **Function**: Use the PostgreSQL database extended with PostGIS to store and manage geospatial data.
- **GeoServer**: Publishes geospatial data.
    - **Function**: Publish spatial data processed from QGIS through GeoServer, providing map layer services.
- **Flask App**: Manages data acquisition, processing, and API interface provision.
    - **Function**: Acts as the core of the system, responsible for data acquisition, processing, and provision of data services through REST APIs.
    - **Models**:
        - Define Data Fetching Functions: Define functions to fetch data from the database.
        - Process Query Results: Process query results.
        - Convert Data to JSON: Convert data to JSON format.
    - **API Interfaces**:
        - Context API:
            - context api/regions<region>: Acquires statistical data for a specific region.
        - Landslides API:
            - landslides api/regions<region>: Acquires landslide data for a specific region.
        - Regions API:
            - regions api/regions: Acquires data for all regions.
- **Dashboard**: Visualizes and interacts with data.
    - **Function**: Provides a user interface for selecting different regions and viewing detailed statistical information and risk analysis.
    - **Components**:
        - Selector Component: Allows users to select different regions.
        - Context Component: Displays statistical data for the selected region.

- Landslides Component: Displays landslide risk data for the selected region.
- Download Toolbar Component: Provides data download functionality.
- Map Component: Displays landslide risk maps and other related layers.
- Layer Management Component: Allows users to manage and switch between different map layers.

2.3 Technology Stack

- **Database**:
  - **PostgreSQL**: An open-source object-relational database management system with powerful extensibility and standards compliance, suitable for complex data management tasks.
  - **PostGIS**: An extension to PostgreSQL adding support for geospatial objects, enabling the handling and storage of geographic information system (GIS) data.
- **Web Server**:
  - **Flask**: A lightweight web application framework written in Python, designed for simplicity and flexibility, suitable for rapid development of web services and REST APIs.
  - **GeoServer**: An open-source server software for publishing geospatial data. GeoServer is designed to interact with many different data sources and output data to various clients, including web maps.
- **Data Processing**:
  - **Pandas**: A powerful Python data analysis library providing high-performance, easy-to-use data structures and data analysis tools, suitable for various data processing tasks.
  - **GeoPandas**: An extension to Pandas supporting geospatial data types, enabling the handling and analysis of geospatial data.
  - **Requests**: A simple and easy-to-use Python library for sending HTTP requests, suitable for interacting with APIs and acquiring data.
  - **QGIS**: An open-source geographic information system application for processing and analyzing spatial data and publishing data through GeoServer.
- **Dashboard**:
  - **Dash**: A Python framework for building analytical web applications based on Flask and Plotly, suitable for rapid development of interactive data visualization applications.
  - **Plotly**: A Python library for creating interactive charts, supporting various chart types, including scatter plots, line charts, bar charts, and more.

    - **Leaflet**: An open-source JavaScript library for building interactive maps, suitable for displaying and interacting with geospatial data.
    - **Dash Leaflet**: An extension library integrating Leaflet into Dash, for creating and managing interactive maps in Dash applications.
- **Version Control**:
    - **Git**: A distributed version control system for tracking code changes and collaborative development.
    - **GitHub**: A code hosting platform based on Git, providing version control and collaboration tools, suitable for team development and code management.
- **Development Environment**:
    - **PyCharm**: A professional Python integrated development environment developed by JetBrains, offering intelligent code assistance, debugging, integrated version control, and a variety of plugin support.

3. Software Design

3.1 Database Design

The database will include the following tables:

- **regions**: Stores geographic region data, including geometric information (geom), region name (den_reg), and other related attributes.

	id bigint	cod_rip bigint	cod_reg bigint	den_reg text	shape_leng double precision	shape_area double precision	geom geometry
1	1	1	1	Piemonte	1234644.01493	25391636805.5	0106000020787F000001000000010300000003000000EA13000060F5B9DA55F01B41645DDC72E7A05341B0F3FD94
2	2	1	2	Valle d'Aosta	311187.149916	3258608423.93	0106000020787F000001000000010300000001000000AE0400002073D792F2D71741D478E98A6D6C5341E0F21F927
3	3	1	3	Lombardia	1411360.47581	23862867853.3	0106000020787F000004000000010300000001000000040000000D0915CFE81A21D417E6ABC28990453412000E0AD
4	4	2	4	Trentino-Alto Adige	800731.661919	13605964490.5	0106000020787F000001000000010300000001000000720E0000E80434D1D3AE2641984C15BC6CE9534100098A9FI
5	5	2	5	Veneto	1056503.58204	18351494741.1	0106000020787F000001000000010300000008000000A1100001074366B0E7127415E4BC85F0FBE534168BC74931
6	6	2	6	Friuli-Venezia Giulia	659341.083253	7936833767.7	0106000020787F000002000000010300000001000000B0000004A83E575BF9942A41C898BB3604545341C0172653F
7	7	1	7	Liguria	821916.468558	5417714496.69	0106000020787F000009000000010300000001000000110000028A913B0475721412EB29DB78E98524128A913703
8	8	2	8	Emilia-Romagna	1176991.66838	22501817675.9	0106000020787F000004000000010300000001000000120000000E0C798DBCFDE2241EE5A426A48AB5241283A926B
9	9	3	9	Toscana	1304649.60395	22985007078.5	0106000020787F000002400000010300000001000000C000000D066D5475F952441782D2137F6DA514168D5E74A4
10	10	3	10	Umbria	619404.571083	8463968722.17	0106000020787F000002000000010300000001000000280000038EF38452CB6274184C0CA75F46E5241E024062219
11	11	3	11	Marche	618848.328436	9344542629.09	0106000020787F000001000000010300000002000000E3090000F80FE9777D7C28410EBE305DBB985241E0E995F24
12	12	3	12	Lazio	1056599.44	17236487921.2	0106000020787F000010000000010300000001000000A000000B88D06D0EBBB2A412063EE92A242514138EF38A5
13	13	4	13	Abruzzo	613572.801521	10828888286.1	0106000020787F000001000000010300000001000000EB090000381AC09B7D852B4198BB9640262952418048BF9DI
14	14	4	14	Molise	433630.82851	4459797834.28	0106000020787F000001000000010300000001000000BB08000008A3921A1A052E41FEB27B3A61D1514118D9CED7
15	15	4	15	Campania	890506.464536	13667847424.5	0106000020787F00000F000000010300000001000000090000078832FEC24A82F41FA5C6D9520F950414850FC580
16	16	4	16	Puglia	1182953.11178	19541033601.1	0106000020787F00001F000000010300000001000000D000000EC9E3C1C7C8A3341A245B6EF0EFE5041EC9E3C9C
17	17	4	17	Basilicata	613038.362707	10071594221.5	0106000020787F000003000000010300000001000000160000007424970F2B6230417EAEB60681F550417424970F2
18	18	4	18	Calabria	847415.624343	15212651631.1	0106000020787F000005000000010300000001000000C0000004C5986D8DE873041AEB6622FD7D850414C5986D8
19	19	5	19	Sicilia	1344687.69115	25824319687.6	0106000020787F000034000000010300000001000000120000000907EFBBAEA1C29411C38675C25064E41D044D8B0I
20	20	5	20	Sardegna	1449230.54457	24106296672	0106000020787F0000AA000000010300000001000000170000000E03D7998B6641B414AEA04B0F0685041F0065F98I

- **context_data_{id}**: Stores statistical data for a specific region, including information on population, households, buildings, industries and services, cultural heritage, and territories. Each region has an independent table, with the id representing the unique identifier of the region.

	Territory (km²) double precision	Industries and services bigint	Buildings bigint	Cultural heritage bigint	Population bigint	Families bigint	Children (0-14) % double precision	Adults (15-64) % double precision	Elderly (65+) % double precision
1	302068.253	4806014	14515795	213360	59433744	24611766	14.009	65.153	20.838

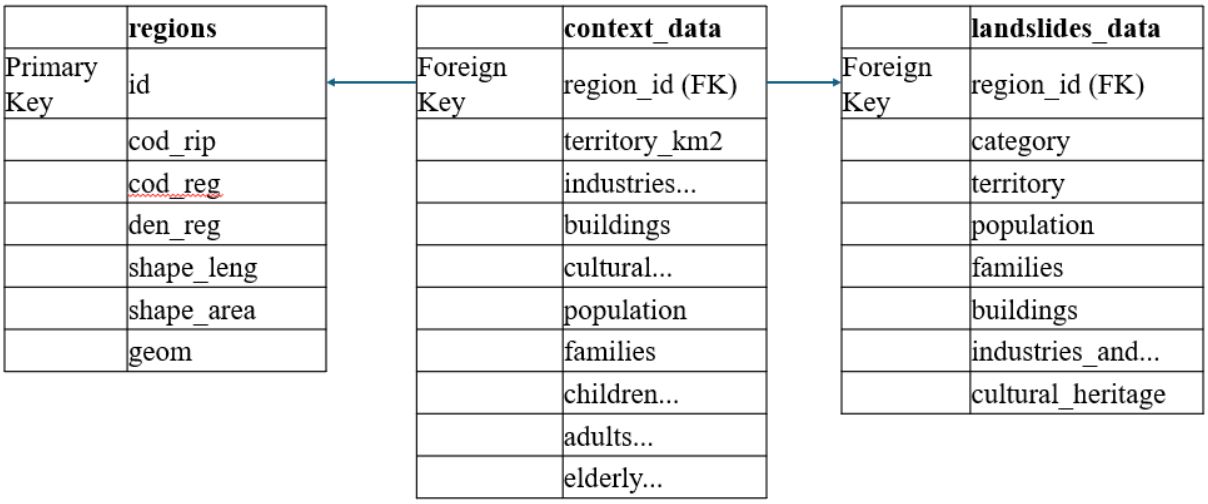
- **landslides_data_{id}**: Stores landslide risk data for a specific region, classified according to different risk categories. Each region has an independent table, with the id representing the unique identifier of the region.

	Category text	Territory text	Population text	Families text	Buildings text	Industries and services text	Cultural heritage text
1	Very high P4	9494 (3.143%)	499749 (0.841%)	206968 (0.841%)	223065 (1.537%)	31244 (0.65%)	5351 (2.508%)
2	High P3	16890 (5.592%)	803917 (1.353%)	340926 (1.385%)	342483 (2.359%)	53197 (1.107%)	7182 (3.366%)
3	Medium P2	14551 (4.817%)	1720208 (2.894%)	727315 (2.955%)	562800 (3.877%)	127356 (2.65%)	10728 (5.028%)
4	Moderate P1	12555 (4.157%)	2006643 (3.376%)	844536 (3.431%)	522206 (3.598%)	147766 (3.075%)	12390 (5.807%)
5	Attention zones AA	6987 (2.313%)	676948 (1.139%)	271208 (1.102%)	216540 (1.492%)	45677 (0.95%)	2502 (1.173%)
6	P4 + P3	26385 (8.735%)	0 (0%)	0 (0%)	565548 (3.896%)	0 (0%)	0 (0%)
7	None zones	241576 (79.978%)	53726722 (90.397%)	22222522 (90.286%)	12648136 (87.137%)	4400737 (91.568%)	175207 (82.118%)

Table Descriptions:

- **regions Table**:
  - **Fields**:
    - id: Primary key, identifying the unique ID of the region.
    - cod_rip: Regional code.
    - cod_reg: Area code.
    - den_reg: Region name.
    - shape_leng: Region boundary length.
    - shape_area: Region area.
    - geom: Geometric information, used to store the geographical shape of the region.
- **context_data_{id} Table**:
  - **Fields**:
    - Territory (km²): Territory area.
    - Industries and services: Number of industries and services.
    - Buildings: Number of buildings.
    - Cultural heritage: Number of cultural heritage sites.
    - Population: Number of people.
    - Families: Number of families.
    - Children (0-14) %: Percentage of children aged 0-14.
    - Adults (15-64) %: Percentage of adults aged 15-64.
    - Elderly (65+) %: Percentage of elderly aged 65 and above.
- **landslides_data_{id} Table**:
  - **Fields**:
    - Category: Risk category (e.g., very high risk P4, high risk P3, moderate risk P2, low risk P1, attention area AA).
    - Territory: Affected territory area (and percentage of the total area).
    - Population: Affected population (and percentage of the total population).
    - Families: Affected families (and percentage of the total number of families).
    - Buildings: Affected buildings (and percentage of the total number of buildings).
    - Industries and services: Affected industries and services (and percentage of the total number).
    - Cultural heritage: Affected cultural heritage sites (and percentage of the total number).

- **Database ER diagram:**



3.2 REST APIs

The REST API provides the following endpoints for client applications to communicate with the server, retrieve, and process data from the database:

3.2.1 GET /context/ string:id

- **Description**: Retrieve statistical data for a specific region.
- **Request Parameters**:

  id: Unique identifier of the region (string format).

- **Response**: Returns statistical data for the specified region, including information on population, households, buildings, industries and services, cultural heritage, and territories.
- **Code:**

```
@app.route("/context/string:id", methods=["GET"])
def get_context_endpoint(id):
    table_name = f'context_data_{id}'
    return get_context(conn=conn, table_name=table_name)

# models.py function
def get_context(conn, table_name):
    df = pd.read_sql_table(table_name=table_name, con=conn)
    df_json = df.to_json(orient="records")
    return df_json
```

3.2.2 GET /landslides/ string:id

- **Description:** Retrieve landslide risk data for a specific region.
- **Request Parameters:**

  id: Unique identifier of the region (string format).

- **Response:** Returns landslide risk data for the specified region, including data for different risk categories such as very high risk (P4), high risk (P3), moderate risk (P2), low risk (P1), and attention area (AA).
- **Code:**

```
@app.route("/context/string:id", methods=["GET"])
def get_context_endpoint(id):
    table_name = f'context_data_{id}'
    return get_context(conn=conn, table_name=table_name)

# models.py function
def get_context(conn, table_name):
    df = pd.read_sql_table(table_name=table_name, con=conn)
    df_json = df.to_json(orient="records")
    return df_json
```

3.2.3 GET /regions

- **Description:** Retrieve data for all regions.
- **Request Parameters:** None.
- **Response:** Returns basic information and geometric data for all regions.
- **Code:**

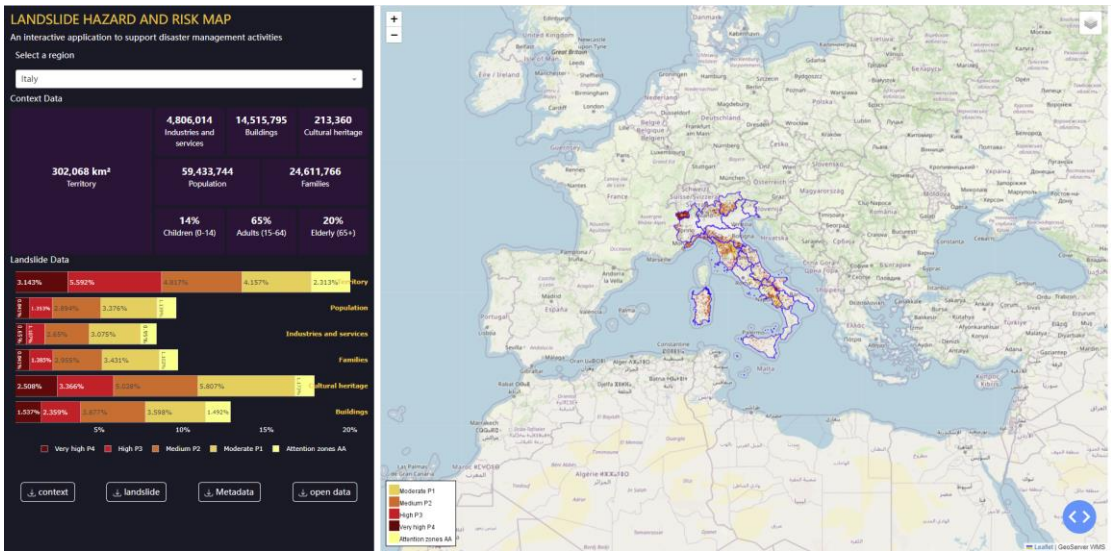
```
@app.route("/regions", methods=["GET"])
def get_regions_endpoint():
    return get_regions(conn=conn)

# models.py function
def get_regions(conn):
    df = pd.read_sql_table(table_name='regions', con=conn)
    df_json = df.to_json(orient="records")
    return df_json
```

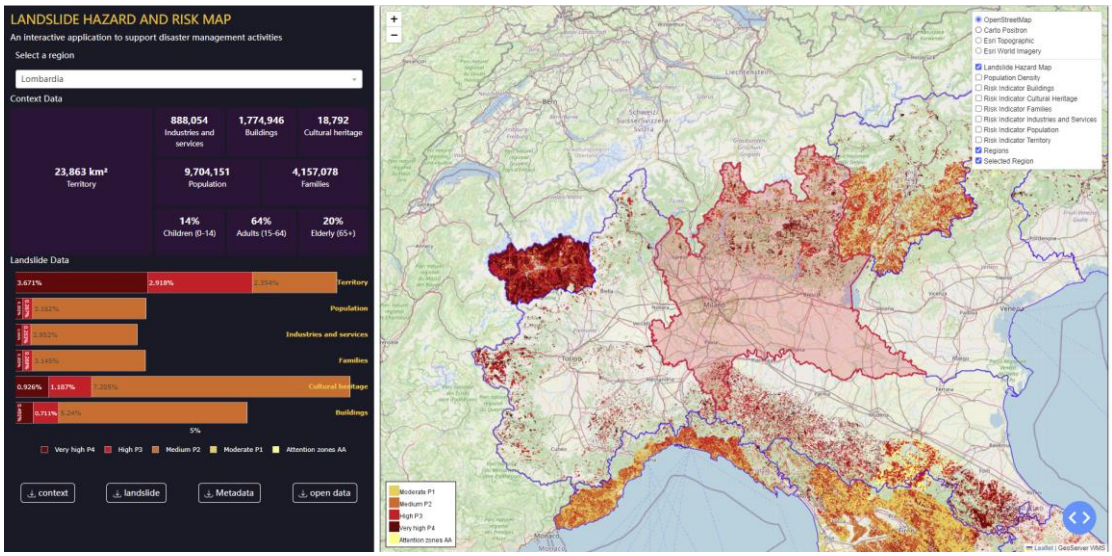
3.3 Dashboard Design

The dashboard plays a crucial role in disaster management and decision support. The design of this system's dashboard aims to enhance the efficiency of landslide risk analysis and management through interactive data visualization and an intuitive user

interface. The main features of the dashboard and their detailed descriptions are as follows:



Initial interface.



The interface for the user to select an area.

Main Features:

- **Map View**:
  - **Description**: Displays landslide risk maps, allowing users to select different regions to view detailed information. The map view is built on Leaflet and Dash Leaflet, providing highly interactive geospatial data presentation.
  - **Functionality**:

        - Region Selection and Highlighting: Users can click on regions on the map to highlight the boundaries and view detailed information.
        - Layer Switching: Provides multiple map layers, allowing users to switch between different types of data, including landslide risk layers, population density layers, industrial and services distribution layers, etc.
        - Map Zooming and Panning: Supports free zooming and panning of the map, allowing detailed viewing of specific area's geographic information.
    - **Technical Implementation**:
        - Uses the Leaflet library to build the basic map view, integrated seamlessly with the Dash framework using Dash Leaflet.
        - GeoJSON formatted data is used for drawing and highlighting region boundaries.
- **Data Display**:
    - **Description**: Displays statistical data and landslide data for the selected region in the form of charts and data cards. The data display part is built on Dash and Plotly, offering diverse charts and intuitive data cards.
    - **Functionality**:
        - Statistical Data Display: Shows detailed statistical data for the selected region, including territory area, population, number of households, number of buildings, number of industries and services, number of cultural heritage sites, percentage of children, percentage of adults, and percentage of elderly.
        - Landslide Risk Data Display: Displays landslide risk data for the selected region, covering different risk levels such as very high risk (P4), high risk (P3), moderate risk (P2), low risk (P1), and attention area (AA), with affected area, population, number of households, number of buildings, number of industries and services, and number of cultural heritage sites.
        - Data Charts: Provides stacked bar charts to intuitively display the detailed classification and proportion of landslide risk data.
    - **Technical Implementation**:
        - Uses the Dash framework to build an interactive data display interface.
        - Employs the Plotly library to create various types of charts for data visualization.
        - Data is dynamically rendered on the front end, fetched from the backend through REST APIs.
- **Download Functionality**:
    - **Description**: Allows users to download data for specific regions, facilitating offline analysis and report preparation. The download

functionality is implemented using Dash's download components, ensuring convenience and flexibility in data acquisition.

- **Functionality**:
  - Data Selection and Download: Users can select and download statistical data and landslide risk data for the selected region, with the downloaded data files formatted in CSV, suitable for further processing and analysis.
  - Download Button: Provides a download button on the data display interface, allowing users to easily obtain relevant data files with a single click.
- **Technical Implementation**:
  - Uses Dash's dcc.Download component to implement the download functionality.
  - The backend provides data interfaces that generate CSV files and respond to frontend requests.

With these features and layout design, the dashboard will provide users with an intuitive, user-friendly interface supporting landslide risk analysis and decision-making. Users can gain a comprehensive understanding and analysis of landslide risks and related statistical data through the map view, data display, and download functionality.

4. Implementation and Test Plan

4.1. Implementation Plan

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_datafolders 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_datafolders:

- Download spatial_data folder

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

In pgAdmin:

i. Open pgAdmin and connect to your PostgreSQL server.
ii. Create a database named se4g24.
iii. Select the newly created database, open the query tool, and run the following command to enable the PostGIS extension:
CREATE EXTENSION postgis;
iv. Run the fill_database.py file to populate the database:
cd backend
python fill_database.py

4. **Configure GeoServer**

i. Download and install GeoServer:
   - Visit the GeoServer download page and download the latest version of GeoServer.
   - Follow the installation instructions.
ii. Start GeoServer and access the GeoServer Web Interface at:
iii. http://localhost:8080/geoserver
iv. 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 the above steps and you will be able to get the project up and running properly. If you encounter any problems during the process, please refer to the project documentation or contact the project development team for assistance.

4.2 Test Plan

Unit Testing:

- Write unit tests to cover each module, ensuring the correctness of each function.
- Test database connections, API endpoints, data processing functions, etc.

Integration Testing:

- Test the integration between modules, ensuring correct data flow between different components.
- Test the database population script, API queries, and dashboard data display.

User Testing:

- Invite users to trial the dashboard, collect feedback, and make improvements.
- Test user interactions, data visualization effects, and download functionality.

5. Bibliography

- ISPRA IdroGEO API Documentation: ISPRA IdroGEO API Documentation
- GeoPandas Documentation: GeoPandas Documentation
- Flask Documentation: Flask Documentation
- Dash Documentation: Dash Documentation
- GeoServer Documentation: GeoServer Documentation
- PostgreSQL Documentation: PostgreSQL Documentation