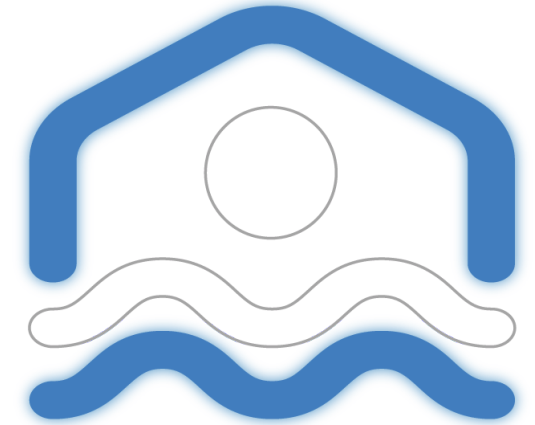
Design Document and Test Plan

__Emilia-Romagna region__ exposure to potential risk and hazard to __flooding__ and its impact on __households__



Sarmiento Ospina, Nataly Alejandra

Saud-Miño, Claudia Isabela

Wang, Xinmeng

Sayegh, John Cullen

Deliverable: DD

Title: Design Document and Test Plan

Authors: Sarmiento Ospina, N. A., Saud-Miño, C. I., Sayegh, J.C., Wang, X.

Version: 2.0

Date: 01/07/2024

Table of Contents

1. Introduction

The Design Document is an important document in the implementation of the project, as it ensures quality-control and sets the expectations for the final end-use product. The following pages contain a step-by-step process explaining the design and development of the project, starting with high-level aims and goals of the application, followed by a detailed analysis of the project database, its software structure, UI design, use cases and stakeholders, and finishing with implementation and team contributions. The application's purpose and scope described below are a summary of the Requirements Analysis Specification Document (RASD), already developed earlier in the project timeline to delineate the overall aim of the group's work.

1.1 Purpose

The purpose of this document is to outline the design of a software system for analyzing flooding in high-risk areas for vulnerable populations in the Emilia-Romagna region of Italy. Given that climate-change-related flooding events are increasingly common across the world and threaten the lives of millions of people each year, the topic is relevant for several use categories and scenarios both in Italy and abroad. Within the Italian context, the region of Emilia-Romagna was chosen as the use case for several reasons; the region's flat topography, its relative proximity to the Po River valley which winds its way throughout the region, and its high-density cities and towns with vulnerable populations render it vulnerable to excessive flooding events. Future flooding has the potential to threaten the lives of thousands of people and cause billions of euros in economic damage across Emilia-Romagna. For these reasons, it is a useful exercise to develop software that can easily highlight and identify areas where potential flood threat levels, concerning vulnerable populations, are at an elevated risk within the region. It is hoped that this project could be expanded to other Italian regions for a large, composite view of flood risk across the country. This data and the method it is accessed and communicated is important for future considerations regarding regional urban planning, flood mitigation schemes, sustainable agricultural practices, and natural biodiversity in the Emilia-Romagna region.

1.2 Scope

This project aims to develop a web-based application that provides real-time flood risk analysis, data visualization, and alerts to assist in disaster management and planning. The ability to visualize the flood mapping data—which is already provided by Idrogeo, an open-source data platform for hydrological risk—will allow organizations and individuals to better understand flood risk in their immediate vicinity clearly and effectively. The

application allows users to easily query flood risk data related to a specific geospatial coordinate and visualize it using GIS software. In doing so, it is hoped this tool empowers users to combat “information asymmetry”, that is, the imbalance of power and transactions between individuals and public and private entities, so that citizens and local groups are more aware of population vulnerability and flood risks.

1. Flood Risk Analysis
    - The software will allow users to select specific communes and run a detailed visualization of flood risk data related to the following categories: surface area, general population, families, buildings, local business units, and cultural heritage sites at risk.
2. Data Visualization
    - Interactive Maps: Create detailed and interactive maps that display flood-prone areas.
    - Bar Graph of Risk Indicators: Highlight high-risk categories per comune in bar graph format.
3. User Interaction and Customization
    - Query and Analysis Tools: Provide a dropdown for users to perform custom queries and analyses, generating tailored visualizations according to their needs.
4. Stakeholder Integration
    - Local Government: Offer functionalities to support local authorities in planning and decision-making processes regarding flood management and emergency response.
    - Community Engagement: Engage local communities by providing accessible information and tools to help them understand flood risks and prepare accordingly.
5. Technological and Design Considerations
    - Scalability: Design the system to handle large volumes of data and a growing number of users, ensuring performance remains robust under heavy load.
    - Usability: Focus on creating an intuitive and user-friendly interface that is accessible to a wide range of users, including those with limited technical expertise.

By addressing these objectives, the project aims to provide a valuable tool for analyzing and managing flood risks in the Emilia-Romagna region, ultimately contributing to improved safety and preparedness for vulnerable populations.

1.3 Definitions, Acronyms, and Abbreviations

- GIS: Geographic Information System
- DBMS: Database Management System
- The app: the application that processes and visualizes the data
- JSON: JavaScript Object Notation
- GPD: GeoPandas abbreviation
- Matplotlib.pyplot: plotting function for Python
- Bokeh: Visualization plug-in for Python
- OSM: OpenStreetMap
- PostgreSQL: Open-source relational database management system
- Flask: A web framework tool for Python
- API: Application Programming Interface
- ISPRA: Superior Institute of Environmental Protection and Research of Italy
- RASD: Requirements Analysis and Specification Document
- Leaflet: API used for interactive mapping

2. System Overview

2.1 Project Description

The project involves creating a system to collect, store, analyze, and visualize flood data for the Emilia-Romagna region. It will focus on areas most at risk and vulnerable populations. The application will leverage historical data to map areas of low, medium, and high flood risk while overlaying demographic information related to populations and built areas that are at high risk in the event of a major flood event. These categories are broken down into low, medium, and high risk as per the IdroGEO API data. The categories that will be analyzed and shown via the system are as follows: surface area at risk, population at risk, families at risk, buildings at risk, local business units (industry) at risk, and cultural heritage sites at risk.

2.2 Key Features

- Data Integration: The program will fetch pre-existing data collated between 2011 and 2021 from IdroGEO's open-source API platform.

- Interactive maps: The user will be able to select a comune by name anywhere in Emilia-Romagna, which will provide analysis and numerical values related to flood risk.
- Risk Analysis Visualization: The six different categories taken from IdroGEO will be shown with a corresponding bar graph to highlight percentages of flood risk in the selected comune.

2.3 Stakeholders

- Local Government: Responsible for regional planning, disaster management, and policy-making, requiring detailed flood risk analyses and reliable data to make informed decisions.
- Emergency Response Teams: Need accurate and timely information to effectively coordinate and execute flood response and mitigation efforts.
- Residents of High-Risk Areas: Require accessible information about flood risks and alerts to take preventive measures and ensure safety.
- Environmental Researchers: Benefit from access to detailed flood data and analysis tools for research and development purposes.
- Non-Governmental Organizations (NGOs): Focus on community awareness and disaster preparedness, utilizing the system to educate and assist vulnerable populations.

2.4 Constraints

- Data Availability: The system's effectiveness depends on the availability and accuracy of real-time and historical flood data.
- Internet Access: Users need reliable internet access to utilize the web-based application and receive real-time alerts.
- Resource Accuracy: Census and flood risk assessment data can only be traced to the commune level (Italy's equivalent lowest level of administrative organization), which means the analysis will not provide a granular, specific analysis of an area or plot smaller than the municipal level.
- Compliance with Regulations: The system must comply with local, national, and international regulations regarding data privacy and security.

By outlining these aspects, the System Overview provides a comprehensive understanding of the project's goals, features, stakeholders, constraints, and assumptions, setting a clear direction for the development and implementation of the flood risk analysis system.

3. Architecture Design

3.1 Overview

This software is developed using Flask, Dash, and PostgreSQL by separating the front-end and back-end and realizing that the results of the user query on the user interface show the corresponding Emilia Romagna region's flood risk per respective category.

The project will use Flask as the backend development framework, and employ Python's library dashboard for frontend development. The database will be designed according to the attributes of retrieved data from IdroGEO and OSM, and established through PostgreSQL, data exchange and storage will be completed in JSON and GeoJSON format.

3.2 System Components

1.) **Database:** Storing data and providing a data access interface. The database component will use PostgreSQL, which is an open-source relational database management system (RDBMS) that maintains extensibility (allowing data customization) and utilizes the standard language SQL (Structured Query Language) of RDBMS. In this way, it will be easy to update, retrieve, and add/remove data quickly from multiple sources. PostgreSQL's extension, PostGIS, is useful for transforming data into geospatial polygons and points, which is necessary for the final map-based product offered to the end user.

2.) Dash: Dash is a Python library used for developing the front end of the software. Dash provides an interactive user interface, displaying data and performing data analysis. Dash's components are comprised of HTML elements, input controls such as sliders and dropdowns, and plotly graphs (which were used to display the bar graph charts indicating risk breakdowns). Dash runs on a web server, ensuring it is user-friendly and can perform actions called by a user, efficiently without reloading the web page.

3.) **Flask**: Flask is used for handling data requests and responses, and calling database query operations. Flask is an easy and simple-to-use Python framework that can build out web applications and works well with SQL-based databases such as PostgreSQL.

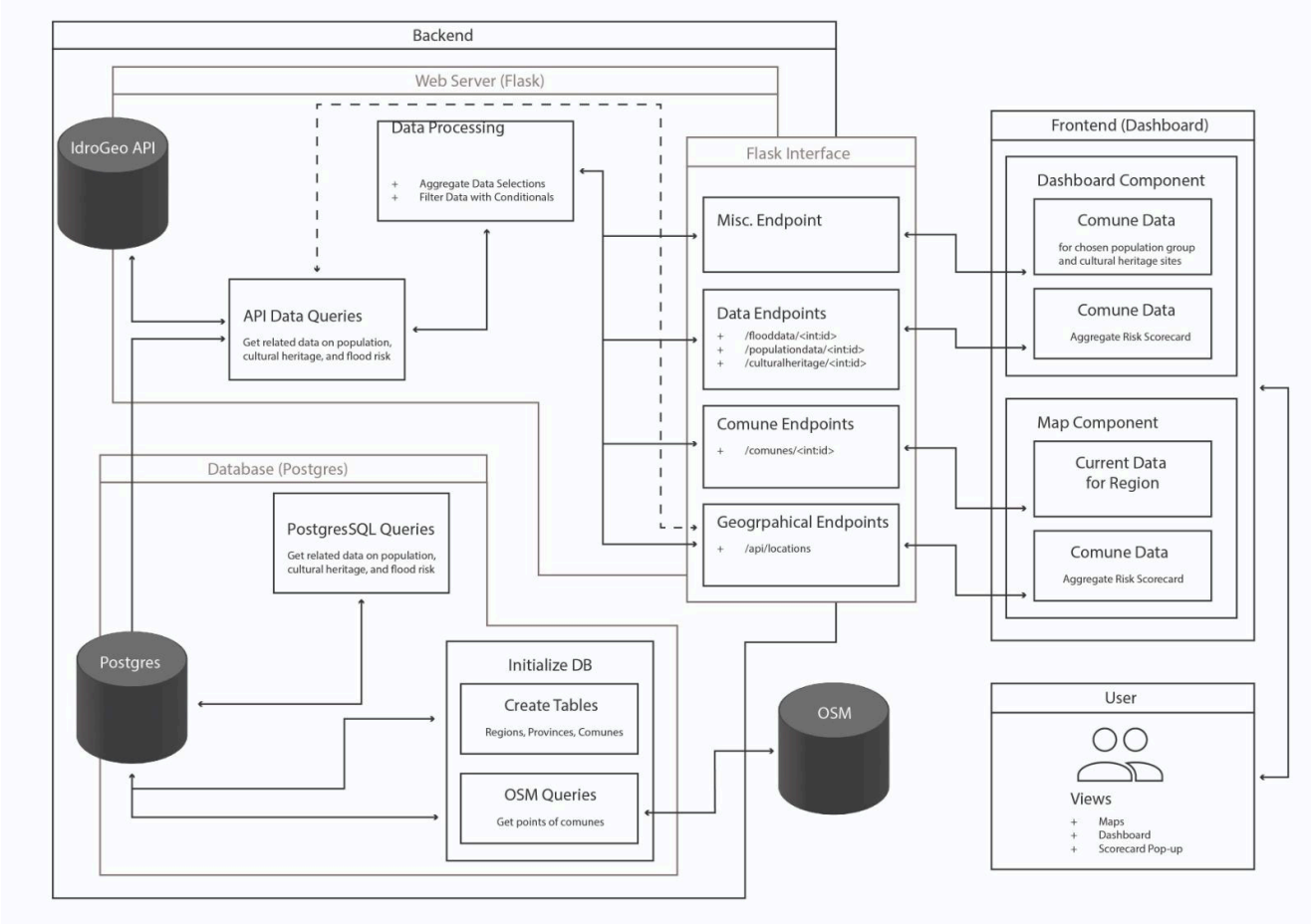


Figure 1: System components

3.3 Data flow

The software follows the methodology of separating the front-end and back-end functions as a development technique. The general idea of data process flow is as follows; when a user initiates a request, the Dash application receives the request and sends it to the Flask application. The Flask application will interact with the database (PostgreSQL), retrieving the data, reconfiguring data, and returning to Dash for visualization according to the user's input. The data that is being recalled is obtained from IdroGEO and OSM. Figure 2 below illustrates the data flow process.

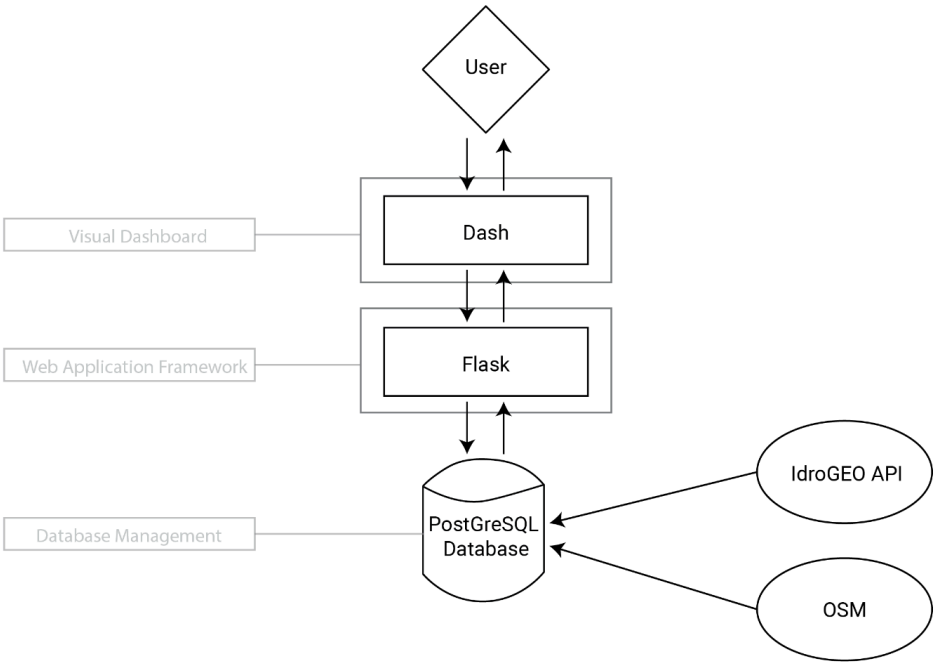


Figure 2: Data Flow Chart

3.4 Presentation Tier

The presentation tier is responsible for rendering and presenting interactive user interfaces. This project will use Dash for creating the interactive interface. The UI of the dashboard will consist of several key components:

1.) **Dashboard:** Displays an overview of the flood index data for Emilia-Romagna, and provides controls for querying and filtering.
2.) **Interactive Map:** A central feature, displaying required data on a map using Geopandas and OpenStreetMap (OSM) as the base map.
3.) **Data Visualization:** A bar chart with flood risk per given category will be shown as well as numerical values related to each comune per given category of flood risk.
4.) **Filter and Search Options:** Users can filter data in two ways, by selecting specific comuni and by selecting attributes in the search bar to visualize a bar chart of the selected category per comune.

The presentation tier communicates with the application tier (Flask) using RESTful API, which retrieves the data related to the communes. Data requests from the frontend are

sent to the Flask backend, which retrieves and processes the data before sending it back to the frontend for display.

4.2 Application Tier

The application layer is built with Flask and Dash. Flask acts as a bridge between the database and front-end, while Dash is the interface provided to users to perform queries and data visualization. The Flask application exposes two REST API endpoints:

Endpoint	HTTP Method	Function	Outcome
'/communes'	'GET'	'getComunes'	performs a SQL query to select distinct records based on the name (apd.nome) and returns the results as JSON
'/communes/<int:id>	'GET'	'getComunesById'	performs a SQL query to select distinct records based on the name (apd.nome) and the unique identifier (apd.uid) provided in the URL and returns the results as JSON.

The team added some styles for the front end, which can be seen here:

```
/* styles.css */

body {
    font-family: Arial, sans-serif;
    background-color: #171b26;
    color: #9fa6b7;
    padding: 20px;
}

#control-row1, #content{
    display: flex;
    justify-content: space-between;
}

#checklist-container {
    display: inline-block;
```

```
    overflow-y: scroll;
    margin-right: 10px;
    height: 200px;
}

#banner{
    font-size: 40px;
    height: 3.5rem;
    align-items: center;
}

#upper-right, #upper-left{
    width: 50%;
    padding: 10px;
}
```

4.3 Data Tier

The data tier consists of the PostgreSQL database, which stores all the relevant data for the application. This includes data retrieved from IdroGEO, specifically the API <pir/comuni/{id}>. To connect all the metadata of each comune, the ID of each comune needed to be specified in the code. Given that there was no existing list of comune IDs from the IdroGEO website, the /pir/comuni API was used to obtain the ID for each comune in the region. After connecting the <pir/comuni/{id}> API with the <pir/comuni/>, the communes and their corresponding risk scores could be collated in the PostgreSQL database. Regarding the geometry attribute, OSM data was obtained through a built-in library. The database schema is designed to store and query flood risk data along with other related geometric information. The key tables and their structures are located below:

The spatial database table of each comune:

```
def create_table(cur):
    # delete table if already exists
    cur.execute("DROP TABLE IF EXISTS comuni CASCADE;")
    cur.execute("""
    CREATE TABLE comuni (
        id SERIAL PRIMARY KEY,
        name VARCHAR(255),
        geom GEOMETRY(MultiPolygon, 4326)
    );
    """)
    conn.commit()
```

The geometry attributes are inserted as a new column of the table:

```
def insert_comuni_data(cur, comuni):
    for idx, row in comuni.iterrows():
        name = row.get('name', 'Unknown')
        geom = row['geometry']

        # print(f"Geometry type before conversion: {type(geom)}")

        # convert Polygon to MultiPolygon
        if isinstance(geom, Polygon):
            geom = MultiPolygon([geom])

        # print(f"Geometry type after conversion: {type(geom)}")

        # only insert Polygon or MultiPolygon type of geometry data
        if isinstance(geom, (Polygon, MultiPolygon)):
            wkt_geom = geom.wkt
            cur.execute("""
                INSERT INTO comuni (name, geom)
                VALUES (%s, ST_SetSRID(ST_GeomFromText(%s), 4326));
            """, (name, wkt_geom))
    conn.commit()
```

The table is created using the following attributes:

```
popidp2_p NUMERIC,
popidp1_p NUMERIC,
fam_tot INTEGER,
fam_idr_p3 INTEGER,
fam_idr_p2 INTEGER,
fam_idr_p1 INTEGER,
famidp3_p NUMERIC,
famidp2_p NUMERIC,
famidp1_p NUMERIC,
ed_tot INTEGER,
ed_idr_p3 INTEGER,
ed_idr_p2 INTEGER,
ed_idr_p1 INTEGER,
edidp3_p NUMERIC,
edidp2_p NUMERIC,
edidp1_p NUMERIC,
im_tot INTEGER,
im_idr_p3 INTEGER,
im_idr_p2 INTEGER,
im_idr_p1 INTEGER,
imidp3_p NUMERIC,
imidp2_p NUMERIC,
imidp1_p NUMERIC,
n_vir INTEGER,
```

```
popfrp2_p NUMERIC,
popfrp1_p NUMERIC,
popfraa_p NUMERIC,
popfrp3p4p NUMERIC,
fam_fr_p4 NUMERIC,
fam_fr_p3 NUMERIC,
fam_fr_p2 NUMERIC,
fam_fr_p1 NUMERIC,
fam_fr_aa NUMERIC,
famfr_p3p4 NUMERIC,
famfrp4_p NUMERIC,
famfrp3_p NUMERIC,
famfrp2_p NUMERIC,
famfrp1_p NUMERIC,
famfraa_p NUMERIC,
famfrp3p4p NUMERIC,
ed_fr_p4 NUMERIC,
ed_fr_p3 NUMERIC,
ed_fr_p2 NUMERIC,
ed_fr_p1 NUMERIC,
ed_fr_aa NUMERIC,
ed_fr_p3p4 NUMERIC,
edfrp4_p NUMERIC,
edfrp3_p NUMERIC,
edfrp2_p NUMERIC,
edfrp1_p NUMERIC,
```

```
edfraa_p NUMERIC,
edfrp3p4p NUMERIC,
im_fr_p4 NUMERIC,
im_fr_p3 NUMERIC,
im_fr_p2 NUMERIC,
im_fr_p1 NUMERIC,
im_fr_aa NUMERIC,
imfr_p3p4 NUMERIC,
imfrp4_p NUMERIC,
imfrp3_p NUMERIC,
imfrp2_p NUMERIC,
imfrp1_p NUMERIC,
imfraa_p NUMERIC,
imfrp3p4p NUMERIC,
bbcc_fr_p4 NUMERIC,
bbcc_fr_p3 NUMERIC,
bbcc_fr_p2 NUMERIC,
bbcc_fr_p1 NUMERIC,
bbcc_fr_aa NUMERIC,
bbccfrp3p4 NUMERIC,
bbccfrp4_p NUMERIC,
bbccfrp3_p NUMERIC,
bbccfrp2_p NUMERIC,
bbccfrp1_p NUMERIC,
```

```
bbcc_id_p3 INTEGER,
bbcc_id_p2 INTEGER,
bbcc_id_p1 INTEGER,
bbccidp3_p NUMERIC,
bbccidp2_p NUMERIC,
bbccidp1_p NUMERIC,
ar_fr_p4 NUMERIC,
ar_fr_p3 NUMERIC,
ar_fr_p2 NUMERIC,
ar_fr_p1 NUMERIC,
ar_fr_aa NUMERIC,
ar_fr_p3p4 NUMERIC,
ar_frp4_p NUMERIC,
ar_frp3_p NUMERIC,
ar_frp2_p NUMERIC,
ar_frp1_p NUMERIC,
ar_fraa_p NUMERIC,
ar_frp3p4p NUMERIC,
pop_fr_p4 NUMERIC,
pop_fr_p3 NUMERIC,
pop_fr_p2 NUMERIC,
pop_fr_p1 NUMERIC,
pop_fr_aa NUMERIC,
popfr_p3p4 NUMERIC,
popfrp4_p NUMERIC,
popfrp3_p NUMERIC,
```

5. User Interface Design

5.1 Overview

The user interface will use a template to display information related to the flood risk of each commune. The layout will provide a dashboard view that contains a map of Emilia Romagna, a zoomed-in map of the selected comune, a third window that provides info on all six selected categories and their respective risk levels (surface area at risk, population at risk, families at risk, buildings at risk, local business units (industry) at risk, and cultural heritage sites at risk), a fourth section that highlights the specific breakdown of risk for each category in bar graph form, and a fifth area that assigns an overall risk category to the comune and recommends specific policy solutions to mitigate flood vulnerability. An overview of the proposed UI design, which was used as a template to iterate through the software development process, can be seen in Figure 3.

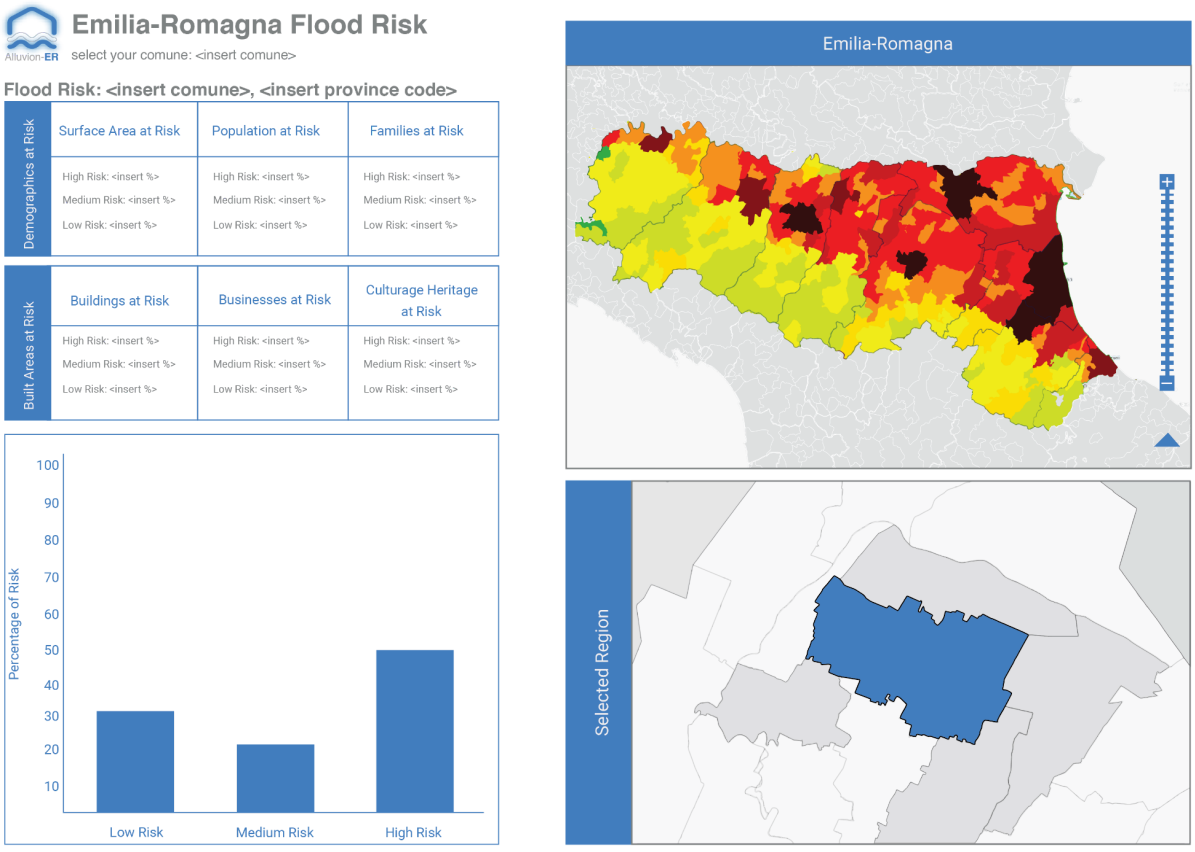


Figure 3: Initial UI Design for determining eventual data visualization

5.2 Data Visualization

Data visualization was an important consideration in the development of the project and one that occupied a significant amount of the team's time. The goal was to highlight and show data in a comprehensible manner that allows comparisons between communes across the region. The large map of Emila-Romagna will use OpenStreetMap (OSM), which provides an easily understandable and user-friendly interface for a diverse set of users. The OSM design was chosen because it is the world's largest open-source geographic database and visualization tool, and it is familiar to most users. This is in keeping with the idea of "usability heuristics", which offer the user an easy-to-use and understandable UI system. The OSM feature allows the user to easily select the specific comune he or she would like to analyze and offers the ability to zoom in and zoom out of the region to view Emilia-Romagna in a wider context. The software UI is shown below in Figure 4.

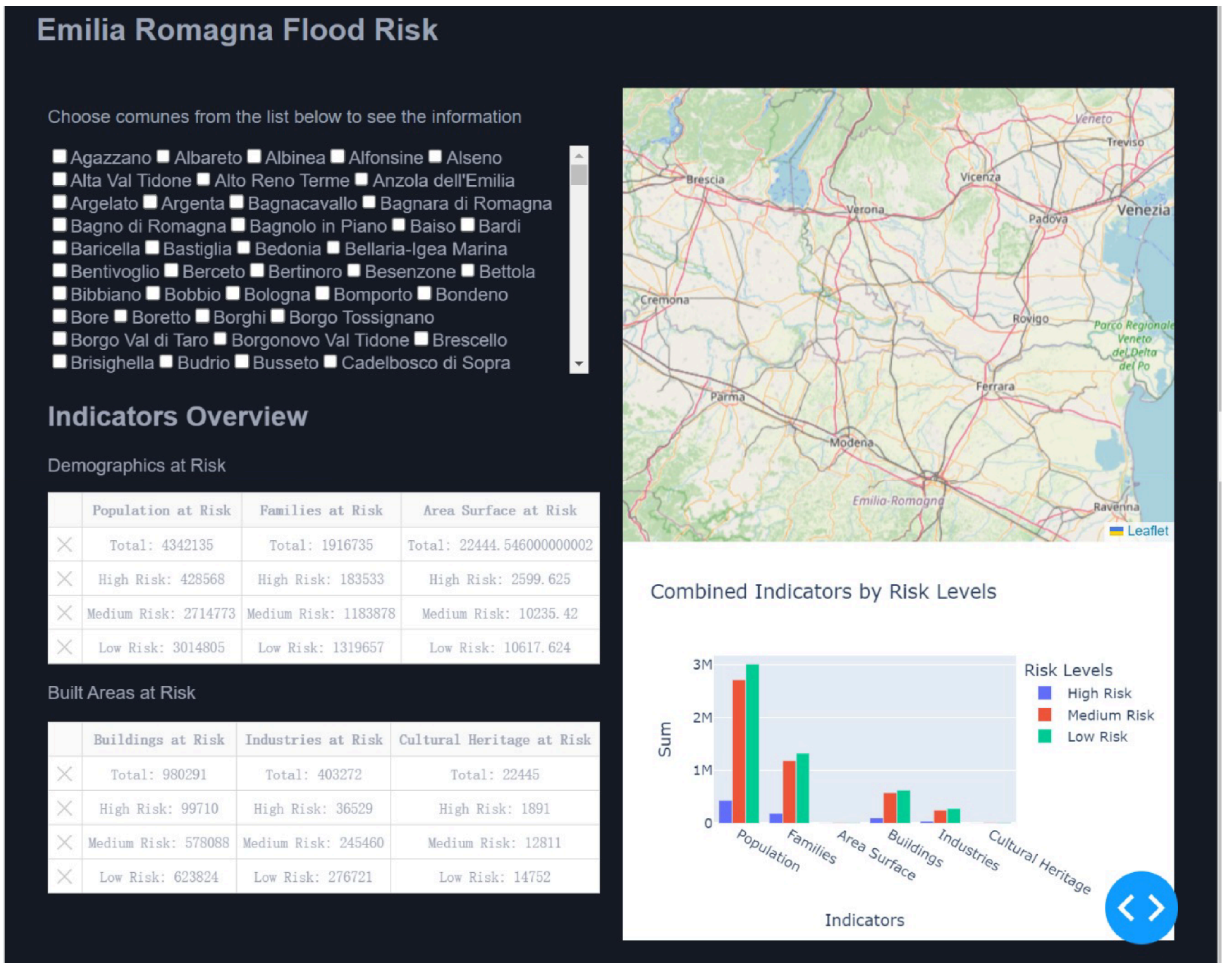


Figure 4: Software Interface

5.3 Analysis Tools

The comune-level map allows the user to view the specific comune at a higher level of fidelity, while the infographics bar displays each risk of the six categories divided into risk (low, medium, high) in numerical format. The bar graph feature also gives users a visual idea of the breakdown of risk for each category, highlighting each according to low, medium, and high vulnerability. In this way, the user can make comparisons between risk categories and communes together to demonstrate what areas are vulnerable to flooding and what specific category is most at risk.

6. Use Cases

6.1 Use Cases

A use case summary helps to define the interaction between users and software in the development process. This helps developers assess the participating actors, entry conditions, flow of events, exit conditions, exceptions, and requirements necessary in the creation of a functional software product.

Use Case No. 1	
Name	Data Query
Primary Actors	User
Entry	Hazard Risk Identification
Flow of Events	**1.** User accesses the search field to start the query **2.** User indicates the desire to search for a specific place and its flooding risk **3.** User selects the scenario and season **4.** System displays search results matching users' supplied criteria **5.** User acknowledges
Exit	User acknowledges information retrieved
Exceptional Cases	**a.** User searches an area out of the study zone

	b. User decides to change zone before querying information
Special Requirements	User only receives the information searched and not other
User Story	As a user, I want to query data so that I can identify possible hazard risks in Emilia-Romagna's zones

Use Case No. 2	
Name	Data Analysis
Primary Actors	User
Entry	Analytical assessment with provided data
Flow of Events	**1.** User sorts the relevant information **2.** User asks the system to display the metadata and attribute tables **3.** System displays search results matching users' supplied criteria **4.** User acknowledges
Exit	User acknowledges information retrieved
Exceptional Cases	**a.** User searches for additional information that is not provided by the software **b.** User decides to change zone while querying information
Special Requirements	User only receives the information searched and not other
User Story	As a user, I want to analyze data in a particular area, so that I can identify possible hazard risks in Emilia-Romagna's zones.

6.2 Use Case Diagram

A use case diagram (Figure 5) was developed to explain how a user will navigate the software once he or she accesses the site.

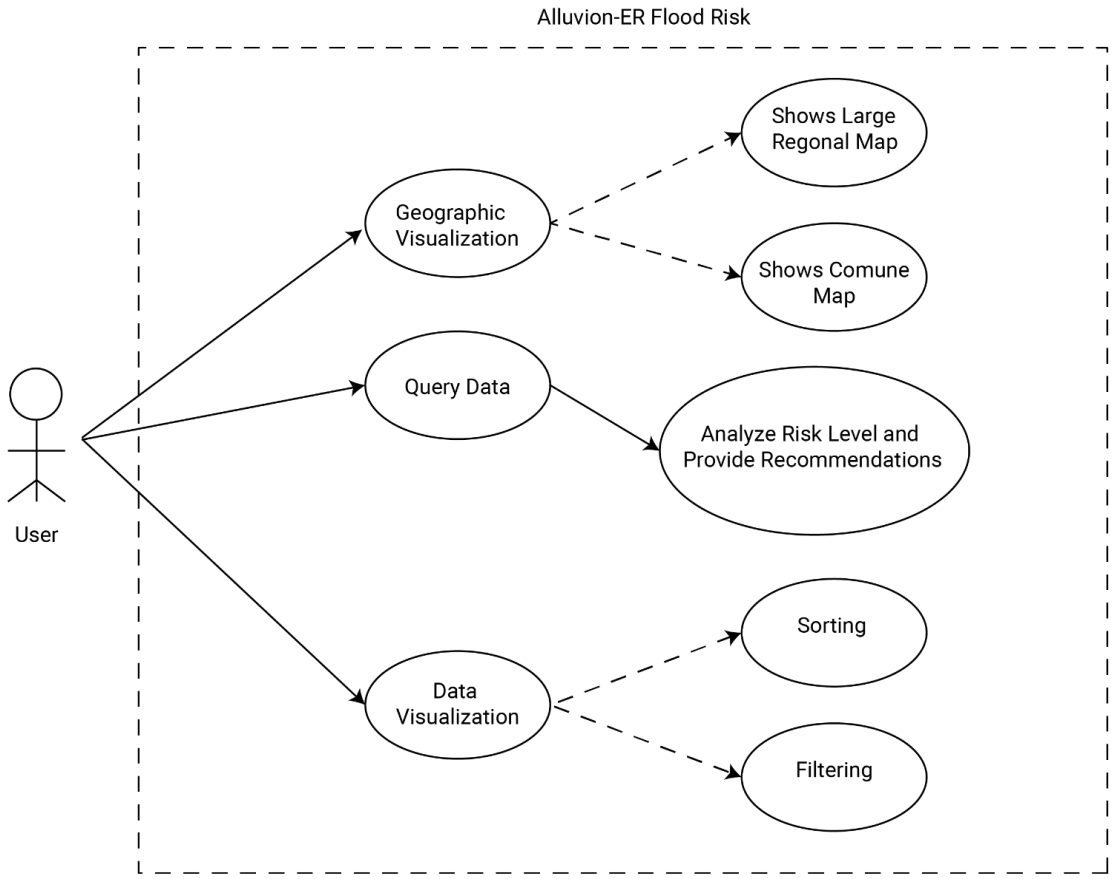


Figure 5: Use Case Diagram

7. Implementation and Test Plan

7.1 Implementation Plan

The test plan of this system will mainly focus on the Flask backend and its interaction with the database. The primary focus is to ensure the API endpoints in the software system, data is stored and retrieved from the local database, and ensure the system is reliable under different circumstances.

The test strategy will include unit tests and end-to-end tests. The unit tests will be used to test the components of Flask, particularly its functions. The end-to-end tests will test the entire workflow of this system, from API requests to database operations and responses.

7.2 Test Plan

Function	Test	Hypothesis	Expected Outcome
Map	Navigate to the geometric center of the region	Set-up Complete	Map component displays the base map.
Pop-up	Select a comune from the list.	Set-up Complete	Map zooms in and displays the selected commune.
Category Table	The risk indicator of cross-bounding categories are visualized	Set-up Complete	The risk category is displayed on the interface.
Infographic Bar Graph	Hover over the graph to see risk indicators for each category	Set-up Complete	The graph displays the percentage value of the risk category.

8. Organization

8.1 Team Roles and Responsibilities

The team developed the software over several months in the spring of 2024. Assuming the notion of a software engineering “sprint”, the team worked rapidly in short periods to develop the code and associated documentation to minimize errors and changes in the development process. Documentation, such as the Requirements Analysis and Specification Document (RASD) was updated as the coding progressed and as the project became more defined.

8.2 Jira Scrum

Jira, a Agile Scrum software application, was used to organize the team’s workflow. Figure 6 below shows the expected “sprint” for the completion of the software product.

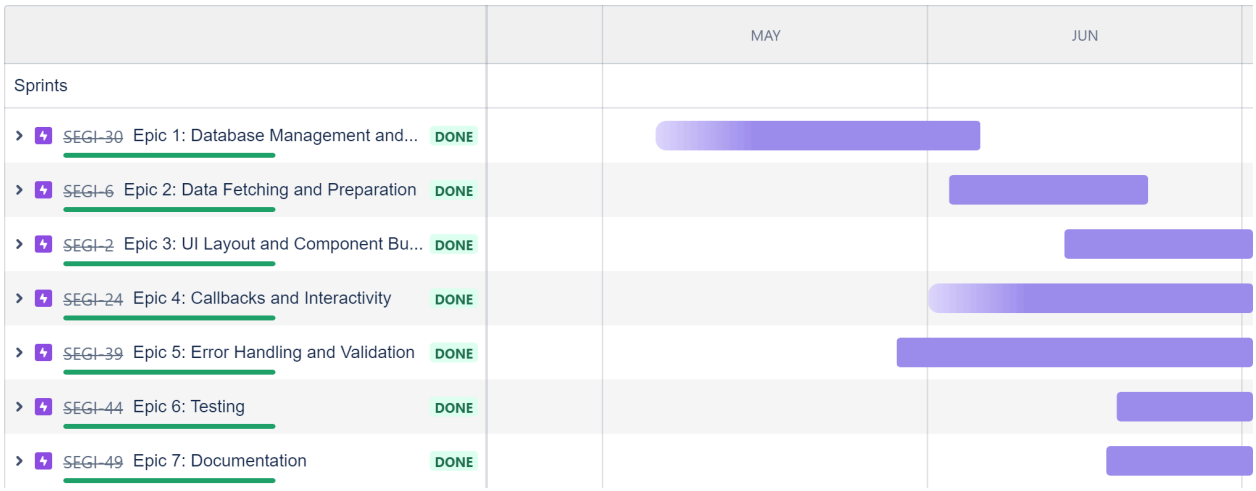


Figure 6: Jira Scrum Handbook Calendar