POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Requirement Analysis and Specification Document

AUTHOR: *Hoda sadat Mousavi Tabar(10904806)*
Hananeh Asadiaghbolaghi(10962418)
Wang Kangfeng(11014059)
Shoayb Sobhani(10967761)

---

1. Introduction

1.1. Context and Motivations

Urban planning and management rely heavily on accurate and up-to-date data. Access to this data through intuitive and interactive visualizations can greatly enhance decision-making processes for city planners, administrators, and the general public. This project aims to develop a web application that presents detailed information about various cities, leveraging geographical and statistical data to create informative visualizations.

1.2. Definitions, Acronyms, Abbreviations

- **API**: Application Programming Interface - a set of functions allowing the creation of applications that access the features or data of an operating system, application, or other service.
- **CRUD**: Create, Read, Update, Delete - the four basic functions of persistent storage.
- **CSV**: Comma-Separated Values - a file format used to store tabular data.
- **DB**: Database - an organized collection of data.
- **GIS**: Geographic Information System - a system designed to capture, store, manipulate, analyze, manage, and present spatial or geographic data.

- **HTTP**: Hypertext Transfer Protocol - the foundation of any data exchange on the Web.
- **JSON**: JavaScript Object Notation - a lightweight data interchange format.
- **REST**: Representational State Transfer - an architectural style for designing networked applications.
- **SQL**: Structured Query Language - a standard language for accessing and manipulating databases.
- **UI**: User Interface - the space where interactions between humans and machines occur.
- **UID**: Unique Identifier - a unique string of characters used to identify a particular entity.
- **UTM**: Universal Transverse Mercator - a coordinate system that divides the world into a series of six-degree longitudinal zones.

1.3. Solution Overview

The proposed solution is a web application that consists of several integrated components:

- **Flask API**: A backend service that handles data retrieval and processing. It provides endpoints to fetch city data from a PostgreSQL database.
- **PostgreSQL Database**: A relational database that stores city data, including geographical coordinates, population statistics, family information, building counts, and surface areas. The PostGIS extension is used to handle spatial data and queries.
- **Dash Frontend**: An interactive front-end interface built with Dash, which allows users to visualize city data through various types of plots and maps.
- **Folium Maps**: A mapping library used to create interactive maps that display the geographical locations of cities along with relevant statistical information.

1.4. Scope and Limitations

Scope

- Provide an API for retrieving data for all cities or a specific city.
- Visualize city data using interactive plots (bar, line, scatter) and maps.
- Enable users to select different cities and plot types for customized visualizations.
- Display detailed statistics for each city, including population, family structures, buildings, and surface areas.

Limitations

- Data availability is limited to what is stored in the PostgreSQL database and fetched from external APIs.
- The application focuses primarily on data visualization rather than in-depth data analysis.
- Performance might be impacted by the volume of data and complexity of spatial queries.
- The initial implementation supports basic CRUD operations for city data but may need further enhancements for additional functionalities.

2. Requirements

2.1. Stakeholders

- **Urban Planners**: Use the application to analyze and plan urban development.
- **City Administrators**: Access and manage city data for administrative purposes.
- **Developers**: Maintain and enhance the application with new features and improvements.
- **Citizens**: Gain insights into city statistics and geographical information.

2.2. Actors

- **User**: Any individual interacting with the application to view city data.
- **Admin**: A user with special privileges to manage the database and API endpoints.
- **System**: The web application comprising the backend (Flask API) and frontend (Dash and Folium).

2.3. Domain Assumptions

- City data in the PostgreSQL database is accurate, regularly updated, and reliable.
- Users have internet access and can use a web browser to interact with the application.
- External APIs used for data enrichment are operational and accessible.

2.4. Requirements

Functional Requirements

- **FR1**: The system shall provide an API endpoint to retrieve all cities' data.

    - **FR1.1**: The endpoint /api/comune shall return a list of all cities with their data.
- **FR2**: The system shall provide an API endpoint to retrieve specific city data.
    - **FR2.1**: The endpoint /api/comune/<int:uid> shall return data for the specified city.
- **FR3**: The system shall visualize city data using interactive plots.
    - **FR3.1**: Users shall be able to select plot types (bar, line, scatter) for data visualization.
- **FR4**: The system shall visualize city data using interactive maps.
    - **FR4.1**: Users shall be able to view cities on an interactive map with relevant statistics.
- **FR5**: The system shall support user interactions for selecting cities and plot types.
    - **FR5.1**: Users shall be able to select a city from a dropdown menu.
    - **FR5.2**: Users shall be able to select a plot type from a dropdown menu.
- **FR6**: The system shall handle errors gracefully and provide meaningful messages.
    - **FR6.1**: The system shall display error messages when data retrieval fails.

Non-Functional Requirements

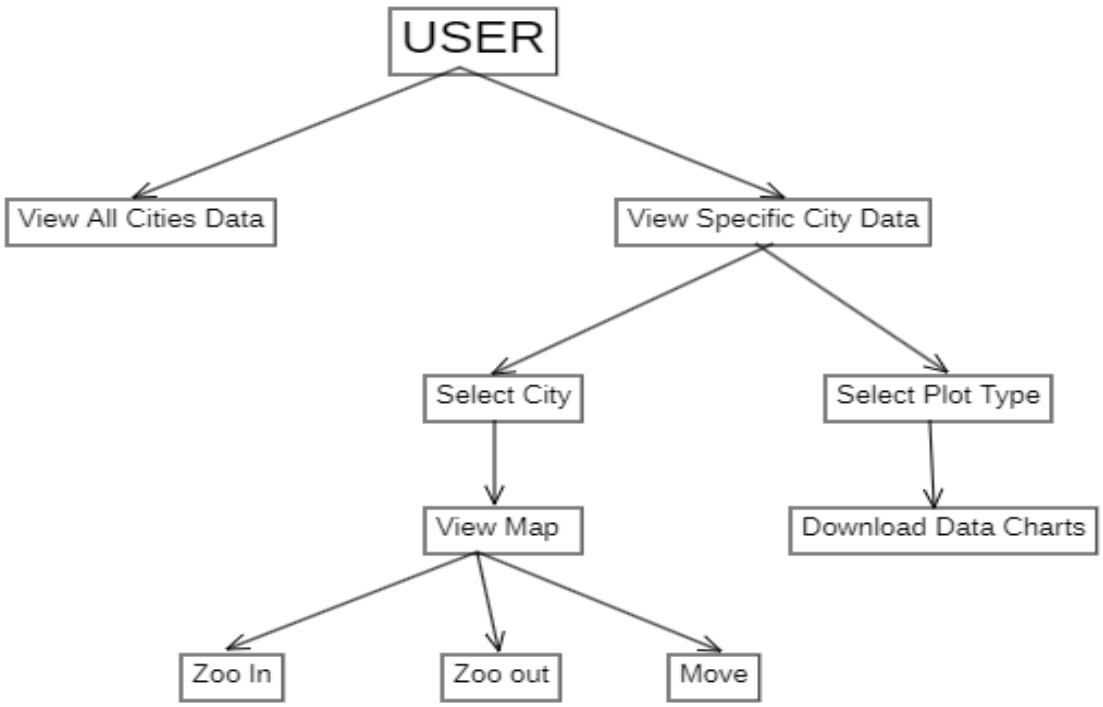
- **NFR1**: Performance: The system shall respond to user requests within 2 seconds under normal load.
- **NFR2**: Availability: The system shall be available 99.9% of the time, excluding scheduled maintenance.
- **NFR3**: Data Integrity: The system shall ensure the accuracy and consistency of data.
- **NFR4**: Usability: The system shall provide a user-friendly interface that is easy to navigate.
- **NFR5**: Security: The system shall secure sensitive data and prevent unauthorized access.

2.5. Use Case Diagram and Description

The use case diagram represents the interactions between users and the system, highlighting the key functionalities provided by the application.

Use Case Diagram Description

- **User** interacts with the **Web Application**.
  - **View All Cities Data**: Allows the user to view a comprehensive list of all cities and their data.
  - **View Specific City Data**: Enables the user to select and view detailed information about a specific city.
  - **Select City**: Users can choose a city from a dropdown menu.
  - **Select Plot Type**: Users can choose the type of plot (bar, line, scatter) for data visualization.
  - **View Map**: Users can view an interactive map displaying the city's location and statistics.



2.6. Use Cases

Use Case: View All Cities Data

Actors: User

Description: The user requests to view data for all cities.

Preconditions: The user is on the web application interface.

Postconditions: The system displays a list of all cities with relevant data.

Main Flow:

1. User navigates to the "View All Cities" section.
2. System retrieves data from the database.
3. System displays the list of cities and their data.

Use Case: View Specific City Data

Actors: User

Description: The user selects a specific city to view detailed data.

Preconditions: The user is on the web application interface.

Postconditions: The system displays detailed data for the selected city.

Main Flow:

1. User selects a city from the dropdown menu.
2. System retrieves data for the selected city from the database.
3. System displays detailed data for the selected city.

2.7. User Stories

User Story 1: As a user, I want to view a list of all cities so that I can see an overview of available data.

Acceptance Criteria:

- The user can access a list of all cities.
- The list displays relevant data for each city.

User Story 2: As a user, I want to select a specific city so that I can view detailed information about it.

Acceptance Criteria:

- The user can select a city from a dropdown menu.
- Detailed data for the selected city is displayed upon selection.

User Story 3: As a user, I want to view city data in different plot types so that I can better understand the information

n.

Acceptance Criteria:

- The user can choose between different plot types (line, bar, scatter).
- The selected plot type is applied to the data visualization.

User Story 4: As a user, I want to view the location of a city on a map so that I can understand its geographical context.

Acceptance Criteria:

- The user can view an interactive map displaying the city's location.
- The map provides relevant city statistics.

3. Bibliography

- Flask Documentation: Flask
- Dash Documentation: Dash
- psycopg2 Documentation: psycopg2
- Plotly Documentation: Plotly
- Folium Documentation: Folium
- PostgreSQL Documentation: PostgreSQL
- PostGIS Documentation: PostGIS

This detailed Requirements and Specification Document provides a comprehensive guide for understanding the project's context, scope, requirements, and planned functionalities. It serves as a reference for all stakeholders involved in the development and usage of the web application.