5/25/2022

DESIGN DOCUMENT AND TEST PLAN

Politecnico di Milano – Software Engineering for Geoinformatics

WEB-BASED APPLICATION FOR THE VISUALISATION AND ANALYSIS OF SOLID WASTE MANAGEMENT IN INDIA (SWM)

Authors:

Israa Ahmed Mohammed Abdalla

Ahmed Mohamed Eltahir Yassin

Mohamed Ridaeldin Mukhtar Mohamed

TABLE OF CONTENTS

1.INTRODUCTION:

1.1. PROJECT DESIGN:

A design clarification and documentation are critical stages is crucial before implementing solutions to previously identified client needs.

The study area is in India. It is intended to create a web page demonstrating the environmental damage caused by waste dumping sites. This may lead to some people becoming more aware of this issue and draw the attention of organizations as well as government agencies that may be able to offer support and solutions.

In its primary role, the software design and test plan document provides guidance to outline the overall architecture of the system. as it provides guidance to the development team to clarify what needs to be built and how and the relationship and connection between various components will be illustrated.

1.2. OVERVIEW OF THE DOCUMENT:

The structure and details of the following project areas will be addressed in the document:

Project Database:

An explanation of how Epicollect5 data is retrieved, edited, and synchronized with the web-application's database, as well as how the latter is structured and managed.

System architecture:

The web-server-side application's architecture is divided into three layers: the database (server) and database management system (DBMS), a WSGI compliant web server, and a WSGI application server.

User cases:

The RASD user cases are now described in terms of which and how software components are activated /used in the various use case scenarios.

1.3. PROJECT DESCRIPTION:

A solid waste management website requires spatial and geographic data, a web page demands implementation as a dynamic website, and the primary purpose of a web page visualize the data in an understandable way. A software framework can help with the development of such a website because it provides libraries such as templating engines or session management, as well as predefined classes or functions that can be used to process user input or interact with databases. Furthermore, as specified in the RASD, the application must be developed in Python, for these reasons, the Flask framework was chosen to help with the development of the solid waste management webpage, Python is easily extensible. It is based on the Jinja template engine, which allows the generation of dynamic HTML pages, the Werkzeug toolkit, which is required to write WSGI-compatible Python applications, and it does not specify a database backend, preserving the system's flexibility. Essentially, Flask provides all the tools required to complete the project.

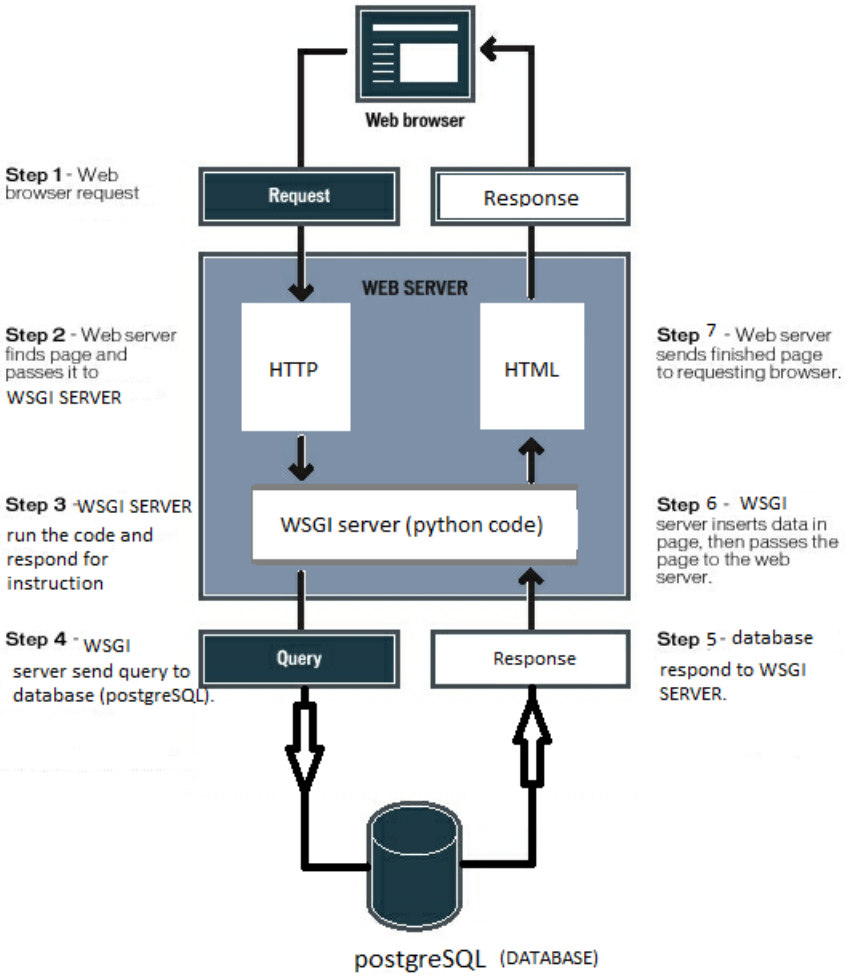
2. DATABASE ARCHITECTURE:

The webpage aims to display data collected from Epicollect5. As a result, developers must filter data and connect databases. The data collected from Epicollect5 for a region in India highlighting solid waste management will be emphasized. A critical decision in the web-application system design is where, when, and how the data from Epicollect5 is retrieved. As mentioned in RASD real-time data will be retrieved from Epicollect5 via REST API which stands for Representational State Transfer, is an architecture proposed by Dr. Roy Fielding by HTTP protocol, process the raw data and returned the results to the client as an HTML response, What matters to the user is to get a quick reaction, and therefore it must be worked on by increasing the interaction between the components of the program and the use of appropriate tools, models and templates to satisfy the user.

the project's data will be stored during development by using pgAdmin is a free and open-source graphical user interface for the administration of PostgreSQL databases, PostgreSQL is a powerful, object-relational database management system (DBMS), PostgreSQL has several administrative fronts ends. The primary one is psql, a command-line tool for entering SQL queries. All queries done in pgAdmin can also be done on the command-line with psql. The PostGIS extension will be used to enable spatial data handling PostgreSQL, because it is built on PostgreSQL, PostGIS automatically inherits important "enterprise" features as well as open standards for implementation.

3. SYSTEM ARCHITECTURE:

The web application architecture describes how applications, databases, and middleware systems interact on the web. It ensures that multiple applications can run at the same time. Staring from the database represented by PostgreSQL which the system data will be stored and going through WSGI where Your application code is running, It interacts with the web server, passes HTTP requests, runs the system code, and implements the functions within the framework of FLASK, which in turn depends on the pre-existing tools Jinja2 and Werkzeug. The web server serves the HTML file and any CSS files required for the client.



4. General scenario: when the user makes a request to the web server, the web server will pass it to the application server (WSGI server), the application server sends a query to the database (PostgreSQL), Database will pass the HTTP response carrying the resource—the HTML document, image, or output of a program and returned to the application server and send the result to the user.

5.Epicollect5, Database, WSGI server, Web server:

5.1. Interaction between them:

The raw data collected by Epicollect5 will be stored in our PostgreSQL database on the local host server and retrieved via the website's REST API, which serves the requested raw data in the form of a JSON file. The dataset for solid waste management was compiled by taking photos of various types of dumping areas and their locations, along with spatial annotations, and the web page will display the data in a map view. The PostgreSQL database is accessed by the web application via the DBMS via CRUD operations, WSGI-server, which runs the python application, which will also return an HTML, which can then be served to the client by the webserver, just like a static HTML. The webserver also stores and serves the CSS files that are referenced in the HTML responses. The flask application object contains all of the app's data as well as the objective functions that govern the app's behavior.

5.2. Filtering and mapping tools:

For the target users in this project, government agencies, municipalities, and environmental organizations, it would be better to clarify the locations and types of waste dump to guide the user, on whether he wants to collect or dump according to the type of waste, so we will have to display the filter options and results to the user in the Map.html (view map) according to the following:

Type of dumping area:

- Open dumping
- Metal bin
- Waste collection Vehicle
- Any other

when the user clicks on a datapoint on the Map.html page will get the type of dumping area.

5.3. Registration Function:

The user provides his data (username, password, address) on the registration HTML page, the HTTP page gets a request and the user’s inputs are passed to the registration function what the function does is check if the user data frame matches with any existing data in the database. If it does not find a match, it returns an error message, and if it writes the user input to the user data frame of the database and then leads the user to the login HTML page.

5.4. Login Function:

The login function must get and post HTTP requests this HTTP request calls the function that returns the login HTML page where the user is asked to provide a username and password. What the function does is to check if all necessary user input was provided and then review the user data frame in the database, if the function matches the provided user input to an entry of the data frame, it returns the HOME. HTML page and if does not it returns an error.

5.5. Logout Function:

This function clears the user session and returns the user to the login HTML page.

6. USE CASES AND IMPLEMENTED REQUIREMENTS:

This section describes the activities executed by the software and the user in a list of instances that are useful for describing the internal operations of the application to explain the functionalities of the software, interactions between the components, and any exceptions.

It describes what happens on the server and client sides when user cases occur, highlighting the numerous activities that take place in these scenarios.

UC1: Registration

• Registration.html page

• The individual user must enter personal data such as a username, password, and address.

• The registration function verifies that the entered username and password do not conflict with any other username stored in PostgreSQL (system table).

• If it doesn't, the registration function saves the user data frame with the user input (username and password) in PostgreSQL (database).

• If this occurs, the software will display an error notice to the user that says, "Invalid username or password."

• The user now has the opportunity to change the password.

UC2: Login:

• Login.html page

• The login function receives the user's username and password and compares it to the information stored in the PostgreSQL (system table).

• If the user's login credentials match those in the database, the function redirects the user to the HOME.html page.

UC3: Map view

• Map.html page

• Because the filtering tool has not yet received input, the filter function is used to extract data entries from the geodata frame.

• The function gets a list of all data entries and passes it to the mapping function.

• The mapping function generates a plot of coordinates on a map and then sends that to the jinja template engine in the form of a variable.

UC4: Filtering

• The user is looking at the Map.html page, which was rendered to show just the data points that the user had chosen via the use of filters.

• When the user clicks on any data point, an HTTP request is sent to the flask application, which then performs a function.

7.Test cases:

UC1 Registration:

TC1:

Inputs: user accesses the registration page and inserts username, password, and address confirming password and category.

Hypothesis on the internal state: username already exists in the system

Expected result: the system informs the user that the username already exists.

TC2:

Inputs: user accesses the registration page and inserts username, password, and address to confirm password and category.

Hypothesis on the internal state: password and confirm password doesn't exist in the system

Expected result: the system informs the user that password and confirms password do not match.

TC3:(Extension)

Inputs: Institution accesses the registration page and inserts username, institute's identification code

Hypothesis on the internal state: username and password match the institute's name and identification code

Expected result: the system informs the user that institution's name and password match what is stored in its database

UC2: Login

The system must allow user authentication

TC1:

Inputs: user accesses the login page and inserts a username and password

Hypothesis on the internal state: user exists in the system with the same password

Expected result: the system brings the user to the home page and welcomes him/her

TC2:

Inputs: user accesses the /login page and inserts a username and password

Hypothesis on the internal state: user exists in the system, with another password

Expected result: the system informs the user that the username or password is incorrect