POLITECNICO
MILANO 1863

Software Engineering for Geoinformatics

## Software Design Document & Test Plan

# Web-Based Geospatial Data Analysis Application on North Rupununi Wetland Guyana, South America

**Authors:**
Ali B. Mohmmed
Hussamalden M. Shereif
Mazin A. Saeed
Nizar O. Fadl Elseed
Nshwan A. Khairalla

**Version 2.0**
June 6, 2021

# Contents

# 1 Introduction:

The design document is the phase of software organization that follows the RASD (Requirement Analysis and Specification Document ) phase. It describes what our software system should perform, how it should interact with the external environment, and what tasks are expected to be assigned to the actors in the external environment. The RASD, design, and coding processes are closely linked.

In principle, the design document is a critical part that will lead producers and group members through the future development route of a piece of software, which will involve system component execution, interaction, and testing.

It's worth noting that the design document is linked to the RASD by meeting all of the defined requirements and avoiding missing any critical functionality.

Since the design document contains the technical part of the project, it is not easy to be understood by any actor, and it requires special experience from the external interactor.

Software design document describes software structure in general, the structure of database, modules, packages, functions and libraries, furthermore how these components interact with each other.

# 2 Software structure:

The software structure consists of 3 components:

## 2.1 Database server:

Database management system (DBMS): is a software package designed to define, manipulate, retrieve, and manage data in a database.

The software uses SQL postgres10 as database management system (DBMS)that create tables, manage, retrieve, and store data that work in local machine.

The program uses strings that interpreter on database management system to create two tables (system table, comments table), these tables store data inserted by the members such as comment, username and password.

The software requires a connection to the Epicollect5 and get the raw data by using REST API, these raw data will be transformed from JSON format to pandas data frame to be implemented on web application. Specifically, displayed as analysis figures (for example bar chart) and the coordinate as point on map.

### 2.2 Web server:

This server takes the request from client and provide the response.

The web browser interacts with web server (full path URL) so it is explicitly or implicitly specifying the HTTP protocol that will be used for transferring client request to the server, so when clicking enter a GET message will reach software web server (by URL address), the web server will return the html page with CSS style.

### 2.3 WSGI server:

WSGI (Web Server Gateway Interface) is the standard interface that the web server must implement in order to be able to execute python code.

# 3 Data base design:

### 3.1 Database:

A database is a collection of information that is organized, so that it can be easily accessed, managed and updated. Computer databases typically contain aggregations of data records or files.

### 3.2 EPICOLLECT5:

The epicollect5 provides a web and mobile app for the generation of forms (questionnaires) and freely hosted project websites for data collection. Data are collected by special technicians North Rupununi started from 18th of

September 2020 and still collecting data until now. It contains the following attribute:

| |
|---|
| Date of entry (dd/mm/yyyy) |
| Time of entry (hh:mm) |
| Technician name |
| Site ID |
| Mobile Coordinate (Longitude, Latitude) |
| Habitat Type: include terrestrial vegetation type, grassland. |
| Vegetation Type: here the area is divided to forest, Grass/shrub mix, Shrub/Trees Mix. |
| Height Grass (centimeter). |
| Density Grass |
| Moisture content |
| Water source |
| Temperature |
| Humidity |

## 3.3 Database schema:

A database schema represents the logical configuration of all or part of a relational database. It can exist both as a visual representation and as a set of formulas known as integrity constraints that govern a database.

### 3.3.1 The database tables:

A table is a collection of related data held in a table format within a database. It consists of columns and rows.

#### 3.3.1.1 The comment table:

The comment table is defined by five columns:

| | |
|---|---|
| Comment ID | Contains an integer number (Serial primary key) |
| User ID | Contains a unique integer number and it should not be null. This column is linked with the system table. |
| Time stamp | Contains time of stored comment |
| Title | Contains the header and it should not be null |
| Body | Contains the main comment and it should not be null |

#### 3.3.1.2 The data table:

The data table contains the dataset being used from the epicollect5 and it is defined by these columns:

| | |
|---|---|
| date and time | The column contains the time of the creation of the data. |
| ID | The column contains the ID and username of the technician who added the data to the data set. |
| Name of the technician | The column contains the real name of technician who created the data. |
| Coordinates | The column contains the locations (mobile coordinates) (Longitude, Latitude). |

| | |
|---|---|
| Standing water | The column contains if the habitat has standing water (yes or no). |
| Habitat type | The column contains the type of the habitat of the area. |
| Water source | The column contains if there is an available water source around the area. |
| Vegetation | The column contains the type of vegetation available in the area. |
| Height of grass | The column contains the height of the grass in centimeters. |
| Density Grass | The column contains the percentage of the availability of the grass (low, medium, high and full). |
| Moisture content | The column contains the availability of the moisture in the area (low, medium, high) and the vegetation color. |
| Temperature | The column contains the temperature of the area in Celsius. |
| Humidity | The column contains the humidity of the area as percentage. |

### 3.3.1.3 The system table:

| | | |
|---|---|---|
| User ID | The column contains the ID of the user | Mandatory. |
| User name | The column contains a name chosen by the user to be used to login | Mandatory. |
| Email | The column contains the email of the user | Optional |
| Password | The column contains a password being chosen by the user in order to login | Mandatory. |

### 3.3.1.4 System table characteristics:

| Name | Data type | length | restrictions | keys |
|---|---|---|---|---|
| User ID | serial | | Not null | Primary |
| User name | varchar | 255 | Not null | |
| Email | varchar | 255 | Null | |
| Password | varchar | 255 | Not null | |

# 4 Modules and packages:

## 4.1 Flask:

Flask contains the main application codes which are responsible for the interaction between the template engine, the database, the server and the functions in between, so it is the backbone of this web application.

Flask application works on: Create/Modify the database structure with functions that are suitable to define structures, interact with the database in such a way that users can make some requests containing queries of posting or retrieving information back from the database to the user's interface.

There are two ways to interact with the function (GET and POST method) so both types of requests can be received.

### 4.1.1 POST message:

The one that allows the client to transfer pieces of information to the server.

### 4.1.2 GET message:

The typical message that is used when the client wants to receive from the server some content.

If nothing is specified explicitly it means the request of type GET.

### 4.1.3 Redirect:

Flask class has a redirect() function. When called, it returns a response object and redirects the user to another target location with specific status code. Brings the user to a new location in their browser (Pages).

## 4.2 Jinja (Template Engine)

Jinja is a fast, expressive and extensible templating engine. Special placeholders in the template allow writing code like Python syntax. The Jinja template engine allows customization of tags, filters, tests and globules. It uses mainly HTML pages as user interface, it is dynamically building HTML pages using familiar Python.

### 4.3 psycopg2:

Psycopg is the most popular PostgreSQL database adapter for the Python programming language. It is designed for multi-threaded applications and manages its own connection pool. The software use psycopj2 to connect with the database.

### 4.4 render_template():

render_template is a flask function from the flask.templating package used to generate output from a template file based on the Jinja2 engine that is found in the application's templates folder.

### 4.5 session:

The session object lets you to keep track of parameters across requests, so track and remember data from request to request.

### 4.6 Request:

The requests module allows you to send HTTP requests using Python. The HTTP request returns a response object with all the response data (content, encoding, status, etc.)

## 5 Libraries and statistical and geospatial analysis:

### 5.1 JSON:

It is an open standard file format and data interchange format that stores and transmits data objects made up of attribute-value pairs and array data types using human-readable text (or any other serializable value). The software uses Json for acquiring data from Epicollect5 and transfer it to pandas data frame.

## 5.2 Bokeh:

Bokeh is a Python library for creating interactive visualizations for modern web browsers.

Bokeh helps you build powerful data applications with a wide array of widgets, plot tools, and UI events that can trigger real Python callbacks, the Bokeh server is the bridge that lets you connect these tools to rich, interactive visualizations in the browser.

The goal of bokeh in the project is to create bar charts describing columns (habitat, etc.) from the data being retrieved and cleaned up from epicollect5 raw data.

The member can utilize this bar charts in his research in any way he/she wants to use it.

## 5.3 Geopandas:

The goal of GeoPandas is to make working with geospatial data in Python easier. It combines the capabilities of pandas and shapely, providing geospatial operations in pandas and a high-level interface to multiple geometries to shapely. GeoPandas enables you to easily do operations in Python that would otherwise require a spatial database such as PostGIS.

The goal of geopandas in the project is to import geospatial data from Json file and offer ability of manipulation operations.

# 6 Application Functions:

A function is a block of code which only runs when it is called.
@app.route: when we call this route the function is called.

## 6.1 Member opens the webpage:

@app.route('/')
@app.route('/Home')
This function renders the Homepage of the web application (/Home)

## 6.2 Users registration:

@app.route('/register',methods=['GET','POST']) This function must answer two HTTP requests:

1. If the request is POST:
    a. The software gets the credential information that is sent by the user (username, email, and password)
    b. The software checks if these credentials have been inserted correctly.
    c. The software checks the username in the database.
    d. If it exists, the software sends an error message to the user." Username is already registered".
    e. If it does not exist, the software saves the credentials in the database
    f. The software redirects the member to login page.
2. If the request is GET:
   The software redirects the user to the registration page.

## 6.3 Member login:

@app.route('/login', methods=['GET', 'POST']).
This function must answer two HTTP requests:

1. If the request is POST:
    a. The software gets the credential information that is sent by the member (username and password).
    b. The software checks the username in the database.
    c. If it does not exist, the software sends an error message to the member "Incorrect username."
    d. If it exists, the software checks the password.
    e. If the password does not correspond to the one on the database, the software sends an error message "Incorrect password.".
    f. If it is a correspondent, the software saves user ID of the member.
2. If the request is GET:
   The software redirects the member to the index.

## 6.4 Member logout:

@app.route('/logout').
The software returns redirecting the member to the homepage.

## 6.5 Member Adds Comment:

@app.route('/addcomment', methods=('GET','POST')). The
function has two cases of answering to HTTP requests:

1. If the request is POST:
    a. The software acquires the comment title and body from the member.
    b. The software stores the comment in the database.

2. If the request is GET:

The software redirects the member to the add comment page.

## 6.6 Member Edit his Comment:

@app.route('/editcomment', methods=('GET','POST')).

The function has two cases of answering to HTTP requests:

1. IF the request is POST:
    a. The software checks if the user ID of member is stored in the database.
    b. If the user ID is not stored, the software will return an error message "Only loggedin users can insert comments!".
    c. If the user ID is stored, the software will acquire the comment from the database.
    d. The software allows the member to edit the comment.
    e. The new comment will be stored in the database with the same user ID.
2. If the request is GET:

The software redirects the member to the add comment page.

## 6.7 Member deletes his comment:

@app.route('/deletecomment', methods=('GET','POST')

The function has two cases of answering HTTP requests:

1. If the request is POST:
    a. The software checks if the user ID of member is stored in the database.
    b. If the user ID of member is stored in the database , the software will acquire the comment from the database.
    c. The software allows the member to delete the comment.
    d. The software deletes the comment from the database.
    e. The software redirects the member to the add comment page.
2. If the request is GET:

The software redirects the member to the add comment page

## 6.8 Member adds data to the database:

@app.route(‘/add data’, methods=(‘GET’,’POST’)).
The function has two cases of answering to HTTP requests:

1. If the request is POST:

   a. The software acquires the database table from the database.
   b. The software allows the member to add data.
   c. The software stores the new data in the data tables.
   d. The software redirects the member to the add data page.

2. If the request is GET:
The software redirects the member to the analysis page.

## 6.9 Visualization:

@app.route('/visualize'):
1. The software redirects the member to the base map page.
2. The software displays specific details about the point (ID, coordinates).

# 7 HTML Templates:

## 7.1 (/Home):

This is the first page of the web application which contains a summary of the study area “North Rupununi wetland”, it is background, characteristics, and some information that users can find on our web application, It contains link for Login, Home is the main redirecting page in the code.

## 7.2 (/RegisterUser):

Accessed when the user chooses to create a new account from the home page. The user must enter his information such as username, password, and email (optional). Then “username " is cross checked for matching with other usernames in the database if no match is found then it will then restore within the database. Also, it redirects the user to the Login page (Login.html).

## 7.3 (/Login):

Users will be directed to this page from the “home” page with (‘/login’) function. They will be asked to provide their login information (username and password), after hitting the login button a database function will compare this information with the database and return with allowing the user to login if it matches any of database information, the user is redirected again to (/index) page and now some extra URLs will be available for the user (visualization, analysis, logout, …)

### 7.4 (/Index):

The page that follows the login page. It contains mainly the visualizations and analysis tools which is the essential privilege of the member. It is redirecting page from (‘/visualization) and (‘/analysis) functions that contains the (‘/logout) function.

### 7.5 (/Logout):

Users will be directed to this page from the “index” page with (‘/logout’) function. After hitting the logout button to confirm logout choices the function will then delete the user’s “name” from the session in (/index) page and redirect back to (/home).

### 7.6 (/Visualization):

Accessed from the index page where the (‘/visualize) function works on displaying base map with location information, the user then can access information displayed in the map based on location.

### 7.7 (/AddData):

By logging into the application, the user can now access this page from (‘/adddata’) function in the Visualization page, database table is retrieved and displayed, inputs will then be stored in the database table with a unique identifier with the username as (User Id).

### 7.8 (/Analysis):

This page can be accessed from the index page that will include some analysis results: statistical analysis (charts) applied by dynamic modules like bokeh to be displayed in the viewer of the page.

## 7.9 (/AddComment):

By logging into the application, the user can now access this page from ('/addComment') function in the analysis page, a comment window is displayed, beside the [post] button is all shown in user interface, inputs will then be stored in the comment table in the database. with the username as (User Id), the time of the comment (timestamp) and every comment will be assigned a unique identifier (Comment_Id).

## 7.10 (/EditComment):

This page is accessed only if the member is logged-in from the analysis page. It is accessed from the ('/addcomment') function, it allows the user to edit the comment only if the User Id of the user is the same as in the comment table in the database. After the comment is modified it redirects the user to the add comment page.

## 7.11 (/DeleteComment):

By logging into the application, the user can now access this page from the ('/deletecomment') function in the analysis page. if the UserId of the user is the same as in the comment_table in the database, the user will be allowed to delete the assigned comment and the user will be redirected to the add comment page.

# 8 Testing Plan

## 8.1 UC1: Registration to the website:

| Use Case Number | 1 |
|---|---|
| Test Case ID | TC_1 |
| Inputs | Username Maldini and password 12345 |
| Hypothesis | The username exists in the database with password 12345 |
| Expected Results | The system informs the user "Username Exist" |

| Use Case Number | 2 |
|---|---|
| Test Case ID | TC_2 |
| Inputs | Username Omer, password 524d |
| Hypothesis | The username and password same as in the database |
| Expected Results | the system brings the user in the home2 page and welcome him |

## 8.2 UC2: login:

| Use Case Number | 2 |
|---|---|
| Test Case ID | Login_3 |
| Inputs | Username Maldini and password not entered |
| Hypothesis | User Maldini exists password should be 1234 |
| Expected Results | The system informs the user to "please enter password" |

| Use Case Number | 2 |
|---|---|
| Test Case ID | Login_4 |
| Inputs | Username Nesta and password cd55j |
| Hypothesis | User Nesta exists, password should be c2356 |
| Expected Results | The system informs the user to “incorrect username or password” |

| Use Case Number | 2 |
|---|---|
| Test Case ID | Login_5 |
| Inputs | Username “Null” and password “Null” |
| Hypothesis | User does not exist password does not exist user must enter his unique IDs to get accessed |
| Expected Results | The system informs the user to “Please enter username and a password” |

| Use Case Number | 2 |
|---|---|
| Test Case ID | Login_6 |
| Inputs | Username Omer and password 5462 |
| Hypothesis | User does not exist in database so password not exist |
| Expected Results | The system informs the user to “incorrect username or password” |

| Use Case Number | 2 |
|---|---|
| Test Case ID | Login_7 |
| Inputs | Username Zakab and password Polimi2 |
| Hypothesis | User exists in database so the password |
| Expected Results | The system allows the user to inter and direct him to home2 page |

## 8.3 UC 5: Enter Comment:

| | |
|---|---|
| Use Case Number | 5 |
| Test Case ID | TC_8 |
| Inputs | User enters “6520027554” |
| Hypothesis | The member enters data do not match the field format in the database |
| Expected Results | The system informs the user to “Comment Format not appropriate” |

## 8.4 UC7: Delete comment:

| | |
|---|---|
| Use Case Number | 7 |
| Test Case ID | TC_9 |
| Inputs | The member Zkzk try to delete comment with Title “weather update” |
| Hypothesis | The user ID of the member in the comment database “User Id” does not match the id of the Titled comment.<br>The system will allow only if they match |
| Expected Results | The system denies the delete command and inform the user “not allowed to delete” |

## 8.5 UC6: Edit comment:

| | |
|---|---|
| Use Case Number | 6 |
| Test Case ID | TC_10 |
| Inputs | The member Omer tries to edit comment with Title "weather update" |
| Hypothesis | The member id in the comment database "User Id" does not match the id of the Titled comment.<br>The system will allow only if they match |
| Expected Results | The system denies the delete command and inform the user "not possible to edit" |

## 8.6 UC 8: Add data to the database:

| | |
|---|---|
| Use Case Number | 8 |
| Test Case ID | TC_11 |
| Inputs | The member enters "too heigh" to the data base under the Colum "heigh of grass" |
| Hypothesis | The column format is integer |
| Expected Results | The system will tell the user "Incorrect Format" |

## Efforts:

| | |
|---|---|
| Ali B. Mohmmed | 12 hours |
| Hussamalden M. Shereif | 12 hours |
| Mazin A. Saeed | 12 hours |
| Nizar O. Fadl Elseed | 12 hours |
| Nshwan A. Khairalla | 12 hours |