Test Strategy Document

Scope

- The document will be verified and approved by Dr.Sachin Chaudhari(Client)
- Testing activities:
    1.Testing of the correct linking of pages on the webapp
    2.Testing the operations done on the database
    3.Testing the api calls using the database
    4.Testing the plotting of graphs

Test Approach

Process of testing

1.Setting up the test for individual components(feeding of data, database operations, api calls, graph plotting, testing readings)

2.Running the tests(running the operations)

3.Evaluating outputs(comparing test result with expected outcome)

Testing levels

1.Unit Testing of individual components(conversion of xls to db, db functions, graph plotting, pages, etc)

2.Integration testing for db creation and api calls, api calls and graph plotting, backend and frontend, etc

3.System Testing of the overall web app. Running it and checking all functionalities

Roles and responsibilities of each team member

1.Anshul Padhi-Database Management and Backend Management

2.Saransh Rajput-Consumption Patterns, Graph Plotting, Mailing Support

3.Fiza Husain-Frontend Management, UI Design, Login, Contacts Feature

4.Jayati Narang-Frontend Management, UI Design, Leakage Handling

Types of Testing

1.Functional Testing:

1.1)Test whether database is added and all the data is fed into it

1.2)Test whether the model functions in the webapp are working and giving correct results

1.3)Test whether app is running on localhost

1.4)Test whether all the links within the app are working

1.5)Test whether outgoing links are working

1.6)Test for broken links

1.7)Test whether the graphs are rendering on the webapp

1.8)Test whether the graphing has been done properly by comparing the values in the graph with the actual results

1.9)Test whether anomalies are being detected

1.10)Test whether the IIIT map is being displayed with the leakage values and pipelines

1.11)Test whether we are able to login once registered

1.12)Test whether the NRW percentage is correctly displayed

1.13)Test whether the pH value is correctly displayed

1.14)Test whether the TDS readings are correctly displayed

1.15)Test whether leakage points are highlighted on the map

1.16)Test to see that the mails are being sent

2.Interface Testing:

2.1)Check whether the site has a clean and minimal UI

2.2)Check whether the web app is easy to navigate

2.3)Check whether the instructions are easy to follow

2.4)Checking for errors in text

2.5)Checking to see how the overall web app looks. It should catch the eye.

Test Environment

Backend Management-

1.Excel-Input data is in the form of an excel sheet

2.MySQL-Converting the sheet to a database for better management of tables

3.Django-For running the app, creating api calls and adding functionalities

4.Python-For running the scripts to edit database and testing

Programs-

1.Text Editors-Sublime, VSCode, VIM

2.Browser-Google, Chrome, Firefox

Frontend Management-

1.Django-For allowing HTML pages with Python objects and for registering users and logging in

2.HTML/CSS-For designing the web pages

3.Plot.ly-For plotting of interactive graphs which are used to find consumption Patterns

4.Openstreet Map-Showing the IIIT map

5.d3.js-For making the water network pipeline

6.Leaflet.js-For presenting the leakage values and highlighting the leakage points

Testing Tools

1.MySQL to check the current state of database after CRUD operations
2.Django in-built shell to check output of the api calls and defined functionality
3.Excel to get actual outputs of the functions we wish to implement in our django app
4.Python Scripting to create scripts to compare the outputs of excel and django functions

Use Cases

1.Analysis of Water Consumption
2.Database Management
3.Ranking by Block per Person
4.Total Consumption Numbers
5.Anomaly Detection
6.pH Readings
7.TDS Readings
8.Leakage Detection
9.NRW Calculation
10.Register
11.Login
12.Mailing Support
13.Contact Support

Test Cases

1.
Test : On clicking 'Graphs' we check whether the desired graph is rendering

Expected Outcome : Desired graph should get rendered on the web app.

2.
Test : We can test operations on database using the 'select' command in MySQL

Expected Outcome : The correct numbers should be displayed in the view command

3.
Test : Using the consumption data and population of a block a score should be created and the ranking should be done on that basis

Expected Outcome : Ranking should be displayed.

4.
Test : We can use the django shell using manage.py to test the functions to get consumption patterns based on block, season, day of the week. We just run the functions in the models.py file to check

Expected Outcome : Consumption data should get displayed.

5.
Test : Total consumption is calculated per day. Shows error if anomaly detected.

Expected Outcome : A notice should be displayed on the total consumption section if anomaly exists

6.
Test : pH readings are taken from the IIIT server where the readings are stored.

Expected Outcome : pH value should be displayed on the home page.

7.
Test : TDS readings are taken from the thingSpeak server where the readings are stored.

Expected Outcome : pH value should be displayed on the home page.

8.
Test : When the total amount of water consumption is less than the total amount of water produced that means there is a leakage. So the location of the leak is found and highlighted on the map.

Expected Outcome : If there is a leakage, the leakage amount should show on the map and the location should be highlighted.

9.
Test : The percentage of NRW(Non Revenue Water) is calculated by comparing total consumption through meter readings and total usage through the block pumps.

Expected Outcome : The NRW percentage should be shown on the home page.

10.
Test : Using the django register feature we add people to the user database through a registration portal.

Expected Outcome : The user should get added to the database on registering.

11.
Test : Using the django login feature we authorise the use of the web app through a login portal

Expected Outcome : On entering credentials if they are correct the user should get redirected to the main page.

12.
Test : The users should be sent a monthly mail which should include the consumption reports.

Expected Outcome : Every month the registered users should be sent a mail which contains the report or water consumption

13.
Test : We need to provide contacts for users so they can ask queries or report issues.

Expected Outcome : On clicking on the contact button the contact info should be displayed.