Sustain-Web: Website Carbon Emission Estimator

SOFTWARE REQUIREMENTS SPECIFICATION

Submitted by:

Anmol Virdi (19103017)
Nandini Jaryal (19103070)

Under the guidance of:

Dr. Geeta Sikka
CSE Department, NIT Jalandhar

A report submitted for the partial fulfillment of the course:

MAJOR PROJECT

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

DR. B.R. AMBEDKAR NATIONAL INSTITUTE OF TECHNOLOGY, JALANDHAR

(144011)

ACADEMIC YEAR: 2022-2023

TABLE OF CONTENTS

DOCUMENTATION HISTORY

Sr. No.	Revision history	Date
1.	First	10.12.2022

GROUP MEMBERS’ DETAILS

Sr. No.	Roll No.	Name	Branch	Contact
1.	19103017	Anmol Virdi	B-Tech. (CSE)	anmolv.cs.19@nitj.ac.in
2.	19103070	Nandini Jaryal	B-Tech. (CSE)	nandinij.cs.19@nitj.ac.in

1. Introduction

Requirements identification is the first step of any software development project. Until the requirements of a client have been clearly identified, and verified, no other task (design, coding, testing) could begin. Usually, business analysts having domain knowledge on the subject matter discuss with clients and decide what features are to be implemented.
Functional and non-functional requirements are the primary components of a Software Requirements Specification.

2. Theory

a. Requirements

Sommerville defines "requirement" as a specification of what should be implemented. Requirements engineering refers to the process of understanding what a customer expects from the system to be developed, and documenting them in a standard and easily readable, and understandable format. Requirements specify how the target system should behave. It specifies what to do, but not how to do it. This documentation will serve as a reference for the subsequent design, implementation, and verification of the system.
It is necessary and important that before we start planning, designing, and implementing the software system for our client, we are clear about its requirements. If we don't have a clear vision of what is to be developed and what all features are expected, there would be serious problems, and customer dissatisfaction as well.

b. Characteristics of Requirements

Requirements gathered for any new system to be developed should exhibit the following three properties:

- Unambiguity: There should not be any ambiguity about what a system to be developed should do. For example, consider developing a web application for your client. The client requires that enough people should be able to access the application simultaneously. What's the "enough number of people"? That could mean 10 to you, but, perhaps, 100 to the client. There's an ambiguity.
- Consistency: To illustrate this, consider the automation of a nuclear plant. Suppose one of the clients says that if the radiation level inside the plant exceeds R1, all reactors should be shut down. However, another person from the client side suggests that the threshold radiation level should be R2. Thus, there is an inconsistency between the two end users regarding what they consider as the threshold level of radiation.
- Completeness: A particular requirement for a system should specify what the system should do and also what it should not. For example, consider software to be

developed for an ATM. If a customer enters an amount greater than the maximum permissible withdrawal amount, the ATM should display an error message, and it should not dispense any cash.

c. Categorization of Requirements

Based on the target audience or subject matter, requirements can be classified into different types, as stated below:

- User requirements: They are written in natural language so that both customers can verify their requirements have been correctly identified.
- System requirements: They are written involving technical terms and/or specifications, and are meant for the development or testing teams.

Requirements can be classified into two groups based on what they describe:

- Functional requirements (FRs): These describe the functionality of a system -- how a system should react to a particular set of inputs and what should be the corresponding output.
- Non-functional requirements (NFRs): They are not directly related to what functionalities are expected from the system. However, NFRs could typically define how the system should behave under certain situations. For example, a NFR could say that the system should work with 128MB RAM. Under such conditions, a NFR could be more critical than a FR.

3. Case Study

“Sustain-Web” is a web app that calculates the environmental impact of a web page and displays the breakdown and measures that can be taken to improve it.
Sustain-Web tests 115 best practices from the reference book published by Eyrolles, with contributions by more than 34 business experts. It also considers the underlying subsea cable factors.

The factors being considered in the carbon emission estimation process include page transfer size and location of the data center and underlying subsea cable length. Not only does this app display the estimated carbon footprint but also suggests ways for developers to reduce these emissions associated with their website

This free app connects developers with sustainable web designing fundamentals. Clients can directly submit their website’s URL and hit the analyze button. This triggers a backend program that first calculates all the factors and then applies appropriate formulae to calculate the appx carbon emissions associated with the website.

The factors include stuff like the number of empty tags, wrong tags, number of script files linked, number of style sheets linked, API Calls, empty image tags, etc. Then it tries to find the data center of the website. If found, it checks if the underlying subsea cable length data is present. If present, it calculates the carbon emission due to underlying subsea cable as well. This gives more accurate prediction for carbon footprint of that website.

Not just this, this app guides developers regarding how they can reduce these factors. After the results are displayed, it suggests clients remove the tags/factors, etc. This web application will also have a blog section where it will mention all the best practices which promote sustainable web design.

Altogether, the purpose of this app is to reduce carbon emissions associated with websites by altering website contents to improve data center life.

The final deliverable would be a web application (Flask App). To reduce the security risk of the software to a large extent, care should be taken when confidential information (eg., passwords) is stored in plain text.

Identification of Functional Requirements:

- **Analyze Facility:** Any User can use the website analyzer facility to check the carbon footprint of their website. There is no need to log in. The reason behind this is that as developers of this app, we’ll focus only on the number of times this analyzer has been used.
- **Suggestions facility:** All the Users will be presented with ways in which they can reduce the carbon footprint of their website. The processing of this part will be done in the backend. For example, if some empty image tags are detected on the website, it will prompt the developer to remove such tags in his/her code.
- **Consulting Facility:** The Users will have the option to book a one-on-one session with consultants of this Sustain-Dev in which the consultants would provide real-time suggestions to reduce the carbon footprint of the client’s website and ways to implement them. Before that, the user will be directed to a payment page where he will be given different payment options to pay the amount against the desired service.
- **Payment Facility:** Users can pay for the optional one-on-one consultation session with sustainable web development experts. The payment gateway would be created using Razorpay to provide a quick and safe payment process.

Identification of Non-Functional Requirements:

Performance Requirements:

- This system should remain accessible 24x7
- The listed equipment should be visible to customers only in a specific range.

- **Security Requirements:**
  - This system should be accessible only within the permitted area.
  - The backend should not store any password in plain text -- a hashed value has to be stored.
- **Software Quality Attributes**
- **Database Requirements**
- **Design Constraints:**
  - The project has to be developed as a web application, which should work with Firefox 5, Internet Explorer 8, Google Chrome 12, Opera 10
  - The system should be preferably developed using a microservices architecture.