Logo, company name

Description automatically generated

**"CARBON FOOTPRINT CALCULATOR"**

A PROJECT REPORT SUBMITTED TO

**THE NATIONAL INSTITUTE OF ENGINEERING, MYSURU**

(An Autonomous Institute under VTU, Belagavi)

In partial fulfillment of the requirements for Project work (Database Laboratory CS5L02),   
fifth semester

**Bachelor of Engineering**

**in**

**Computer Science and Engineering**

*Submitted by*

**VARUN M (4NI19CS120)**

**SURAJ PRAKASH (4NI19CS108)**

**TANAY VERMA (4NI19CS112)**

Under the Guidance of

Mrs. Smitha B

Assistant Professor

Department of CS&E

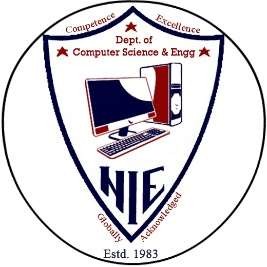
NIE Mysuru

Ms. Padmini MS

Assistant Professor

Department of CS&E

NIE Mysuru



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

2020-2021

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**THE NATIONAL INSTITUTE OF ENGINEERING**



***CERTIFICATE***

This is to certify that the project work entitled “**CARBON FOOTPRINT CALCULATOR**” is a work carried out by **VARUN M(4NI19CS120), SURAJ PRAKASH(4NI19CS108), TANAY VERMA(4NI19CS112)** in partial fulfillment for the project work (Database Laboratory – CS5L02), fifth semester, Computer Science & Engineering, The National Institute of Engineering **(**Autonomous Institution under Visvesvaraya Technological University, Belagavi) during the academic year 2021-2022. It is certified that all corrections and suggestions indicated for the Internal Assessment have been incorporated in the report deposited in the department library. The project work report has been approved in partial fulfillment as per academic regulations of The National Institute of Engineering, Mysuru.

**Signature of the Internal Guides** **Signature of the HoD**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Ms. Padmini MS  
Assistant Professor

Mrs. Smitha B

Assistant professor

Dr. V K Annapurna  
Professor and Head  
Dept. of CS&E  
NIE, Mysuru

**Signature of the Examiners with date**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Name: Mrs. Smitha B   
Designation: Assistant Professor

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Name: Ms. Padmini MS  
Designation: Assistant Professor

**Acknowledgements**

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it possible whose constant guidance and encouragement crown all efforts.

First and foremost, we would like to thank our beloved **principal Dr. N V Raghavendra** for being the patron and the beacon light for this project.

We would like to express our sincere gratitude to **Dr. V K Annapurna, Professor and Head of Department of CSE, NIE, Mysuru** for her relentless support and encouragement.

It gives us immense pleasure to thank our guide **Ms. Padmini MS, Assistant Professor in Department of CSE, NIE** and **Mrs. Smitha B, Assistant Professor in Department of CSE, NIE** for her valuable suggestions and assistance in keeping our project on schedule.

**-Varun M**

**-Suraj Prakash**

**-Tanay Verma**

**Table of Contents**

**Contents** **Page**

1. Introduction 5
2. System Analysis 2
3. System Requirements 3
4. System Design 4
5. System Implementation 10
6. Tools used for Implementation 12
7. System Testing 18
8. Results 20
9. Conclusion and future Enhancements 36
10. References 37

**List of Figures**

Figure No.Description Page

|  |  |  |
| --- | --- | --- |
| 3.1 | ER Diagram |  |
| 3.2 | Motor Schema |  |
| 3.3 | Data Schema |  |
| 4.1 | Software Development Life Cycle |  |
| 6.1 | Starter Page |  |
| 6.2 | Home Page |  |
| 6.3 | Cars Page |  |
| 6.4 | Flights Page |  |
| 6.5 | Motorbike Page |  |
| 6.6 | Public Transport Page |  |
| 6.7 | Results Page |  |
| 6.8 | Reducing Footprints Page |  |

**Chapter 1**

**Introduction**

**Carbon footprint** is the total amount of greenhouse gases (including carbon dioxide and methane) that are generated by our actions.

A **carbon footprint calculator** measures greenhouse gas emissions for a snapshot in time. Carbon calculators are used to calculate **greenhouse gas inventories** of facilities or operations in order to determine the amount of greenhouse gases produced.

The average carbon footprint for a person in the India is 16 tons, one of the highest rates in the world. Globally, the average carbon footprint is closer to 4 tons. To have the best chance of avoiding a 2℃ rise in global temperatures, the average global carbon footprint per year needs to drop to under 2 tons by 2050.

In order to create awareness among people about the rising issues we made a carbon footprint calculator.

This platform provides a user interface which is ease to use. This project uses non-relational database (NoSQL) MongoDB. While a relational database is a viable option most times, it is unsuited for large datasets and big data analysis. This is the major reason for the popularity of the NoSQL database systems in major Internet companies like Google, Yahoo, Amazon, etc. Server-side architecture uses Node.js with Express, EJS. MongoDB will be connected to the server via a popular layer of abstraction called Mongoose.

All the data transmissions happen in JSON format. EJS and CSS will be used to portray the data

extracted from database beautifully to the client side. We will be following MVC (Model View

Controller) architecture in the server side to keep the development simple and manageable.

Overall this project is used to implement a fully functional database. Which includes CRUD (Create

Read Update Delete) operations. Almost all of the queries will be implemented to filter data in the most

efficient way and provide highly secured authorization and logins with the help of Passport.js.

# Chapter 2

**System Analysis**

**Existing System Disadvantages**

1. Existing calculator used to calculate the data generically.

2. The calculation are made based on previously calculated data (decades ago).

3. Calculating data using minimal user input with not so convincing user interface.

4. They aren’t storing users data for future references.

**Proposed System Explanation**

1. We are creating a country specific (India) calculator with elegant user interface and we are

storing users data for past and future references by providing multiple entries.

2. We are calculating the footprint with the help of data provided by Ministry of Environment,

Forest and Climate Change (MoE.F.C.C.) stored in our database.

3. Improvements are made for convenient data analytics with the help of pie charts.

4. Providing detailed analysis of carbon footprint for providing better understanding to the user

about their total carbon emission in order to create awareness among them about the rising

issue of global warming.

**System Requirements Explanation**

### **Client-side hardware requirements**

PROCESSOR : PENTIUM III 866 MHz

RAM : 1GB RAM

MONITOR : 15” COLOR

HARD DISK : 20GB

KEYBOARD : STANDARD 102 KEYS

MOUSE : 3 BUTTONS HARD

DISK : 40GB

**Software Requirements**

Operating System : Windows / Linux / MacOS Front End : HTML, CSS and Server

Side Scripting : Node.js

Database : MongoDB

# Chapter 3

**System Design**

## **System Architecture and Explanation**

As discussed above, this is made using Nodejs, ExpressJS, MongoDB with Mongoose and the front end

is made using JavaScript, HTML and CSS.

### **Node**

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web-application development around a single programming language, rather than different languages for server-side and client-side scripts.

Though .js is the standard filename extension for JavaScript code, the name "Node.js" does not refer to a particular file in this context and is merely the name of the product. Node.js has an event-driven architecture capable of asynchronous I/O. These design choices aim to optimize throughput and scalability in web applications with many input/output operations, as well as for real-time Web applications (e.g., real-time communication programs and browser games).

The Node.js distributed development project was previously governed by the Node.js Foundation, and has now merged with the JS Foundation to form the Open Foundation, which is facilitated by the Linux Foundation's Collaborative Projects program.

Corporate users of Node.js software include GoDaddy, Groupon, IBM, LinkedIn, Microsoft, Netflix, PayPal, Rakuten, SAP, Walmart, Yahoo!, and Amazon Web Services.

### **Express**

Express.js, or simply Express, is a back end web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js.

The original author, TJ Holowaychuk, described it as a Sinatra-inspired server, meaning that it is relatively minimal with many features available as plugins. Express is the back-end component of popular development stacks like the MEAN, MERN or MEVN stack, together with the MongoDB database software and a JavaScript front-end framework or library.

### **MongoDB**

MongoDB is an open source NoSQL database management program. NoSQL is used as an alternative to traditional relational databases. NoSQL databases are quite useful for working with large sets of distributed data. MongoDB is a tool that can manage document-oriented information, store or retrieve information.

MongoDB supports various forms of data. It is one of the many nonrelational database technologies that arose in the mid-2000s under the NoSQL banner -- normally, for use in big data applications and other processing jobs involving data that doesn't fit well in a rigid relational model. Instead of using tables and rows as in relational databases, the MongoDB architecture is made up of collections and documents. Organizations can use Mongo DB for its ad-hoc queries, indexing, load balancing, aggregation, server- side JavaScript execution and other features.

**How it works**

MongoDB makes use of records which are made up of documents that contain a data structure composed of field and value pairs. Documents are the basic unit of data in MongoDB. The documents are similar to JavaScript Object Notation, but use a variant called Binary JSON (BSON). The benefit of using BSON is that it accommodates more data types. The fields in these documents are similar to the columns in a relational database. Values contained can be a variety of data types, including other documents, arrays and arrays of documents, according to the MongoDB user manual. Documents will also incorporate a primary key as a unique identifier.

Sets of documents are called collections, which function as the equivalent of relational database tables. Collections can contain any type of data, but the restriction is the data in a collection cannot be spread across different databases.

The mongo shell is a standard component of the open source distributions of MongoDB. Once MongoDB is installed, users connect the mongo shell to their running MongoDB instances. The mongo shell acts as an interactive JavaScript interface to MongoDB, which allows users to query and update data, and conduct administrative operations.

A binary representation of JSON-like documents is provided by the BSON document storage and data interchange format. Automatic sharding is another key feature that enables data in a MongoDB collection to be distributed across multiple systems for horizontal scalability, as data volumes and throughput requirements increase. The NoSQL DBMS uses a single master architecture for data consistency, with secondary databases that maintain copies of the primary database. Operations are automatically replicated to those secondary databases for automatic failover.

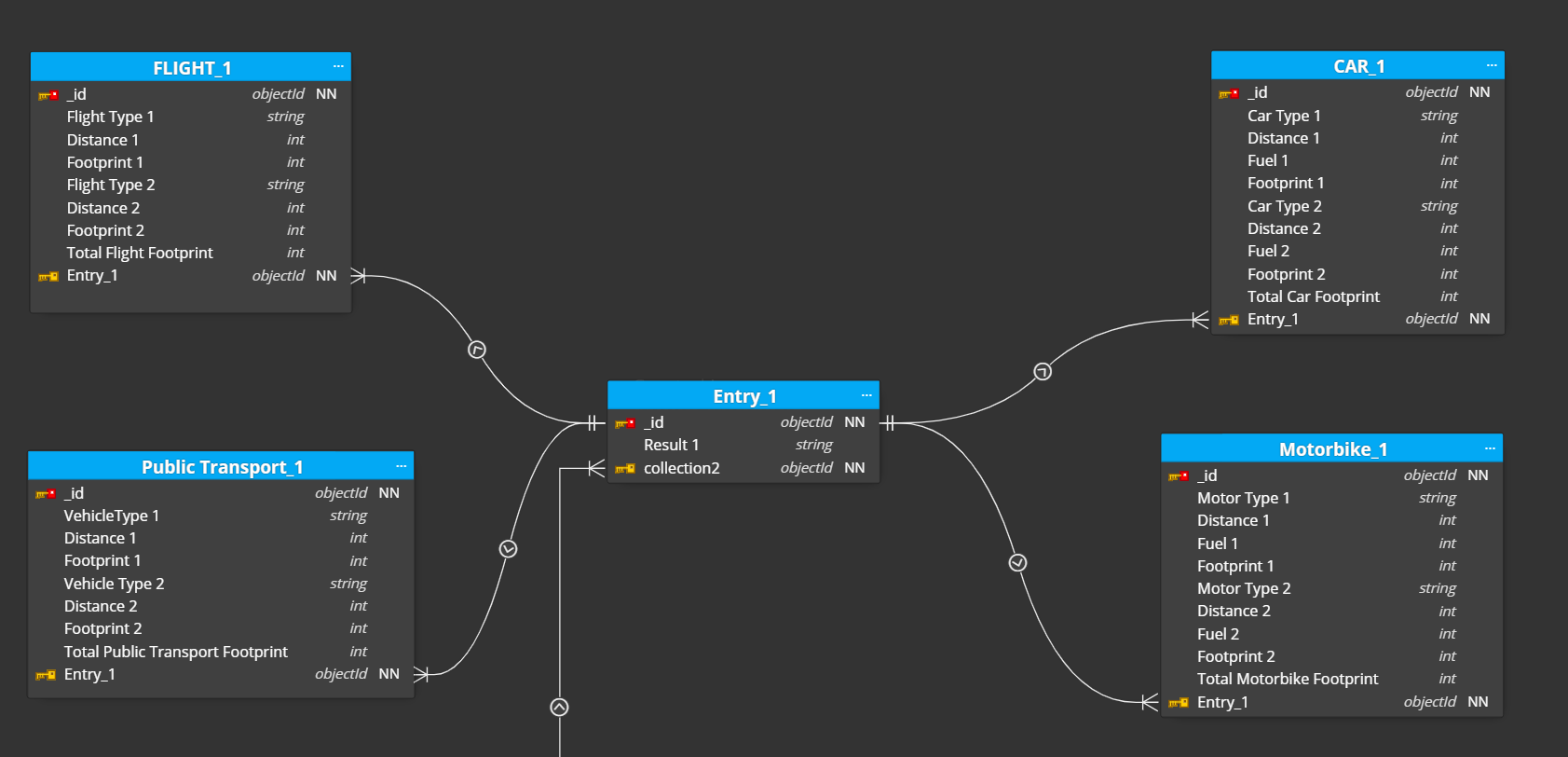
# ER Diagram

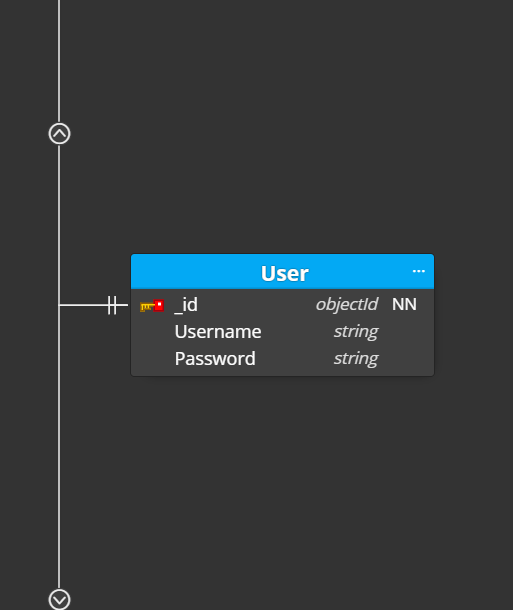
ER diagram stands for Entity Relationship diagram, also known as ERD is a diagram that displays the

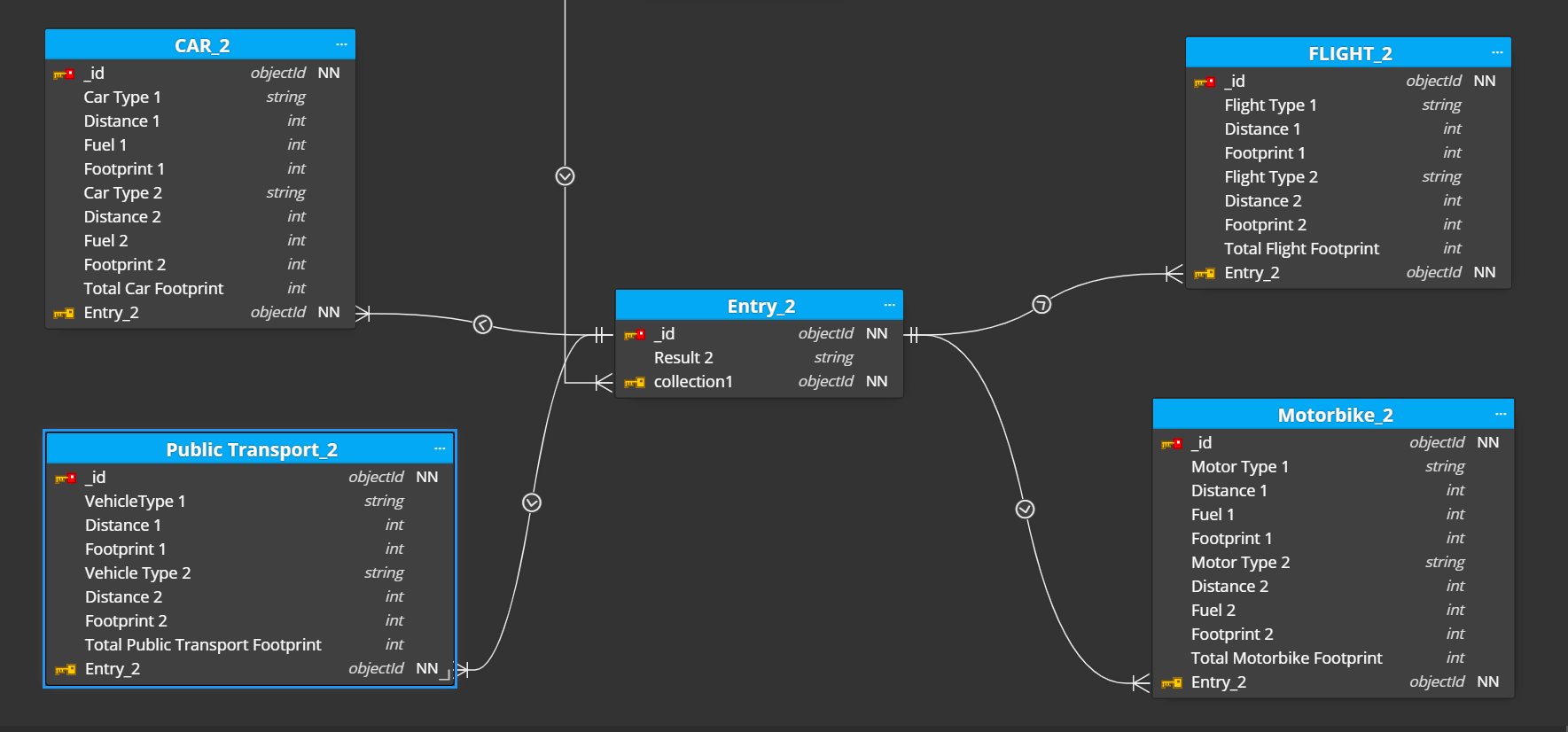
relationship of entity sets stored in a database. ER diagrams help to explain the logical structure of

databases. ER diagrams are created based on three basic concepts.

* 1. Entity
  2. Attributes
  3. Relationship

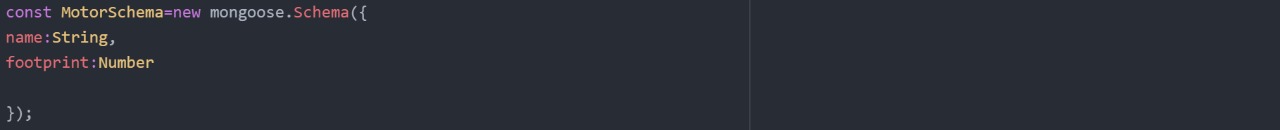
****

****

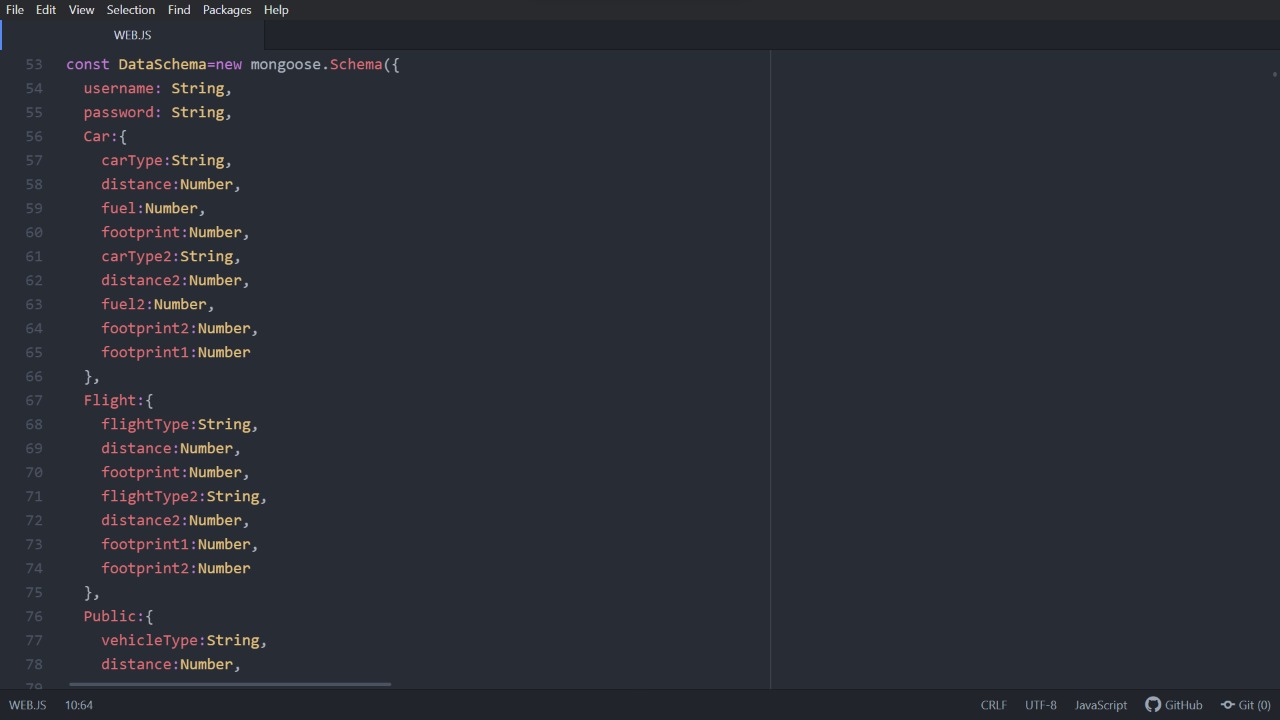


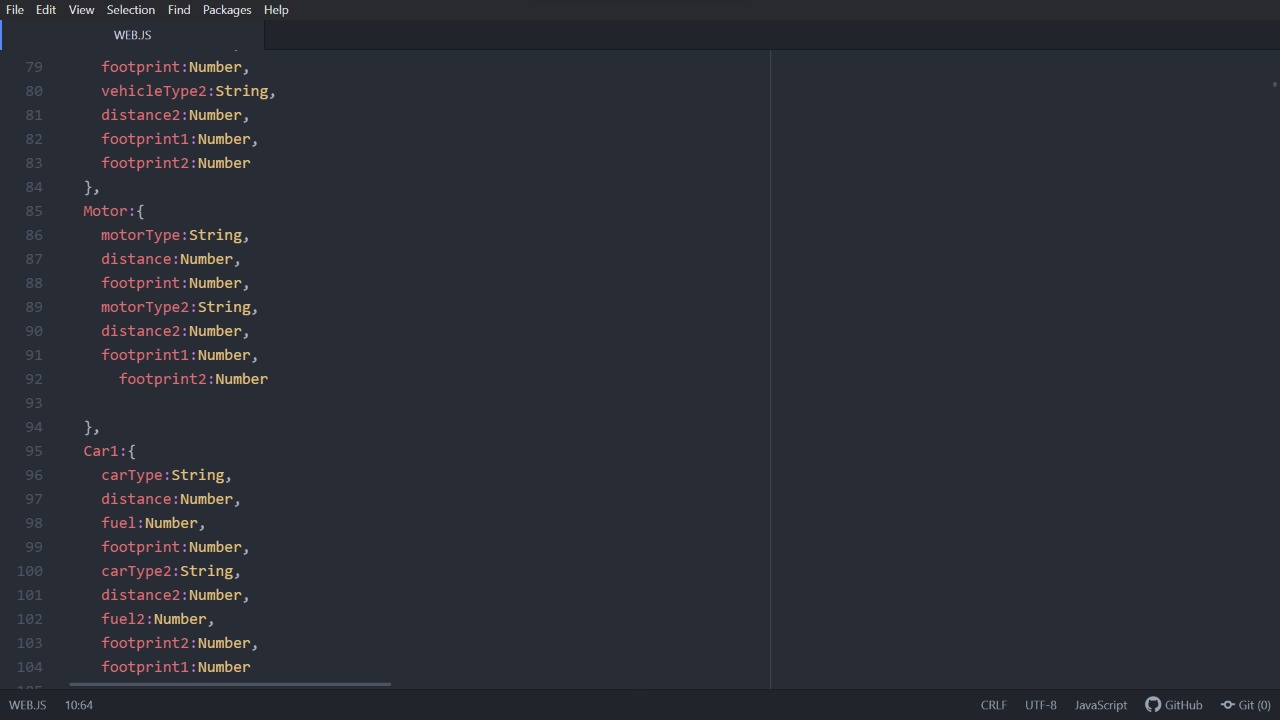
**Fig 3.1: ER Diagram**

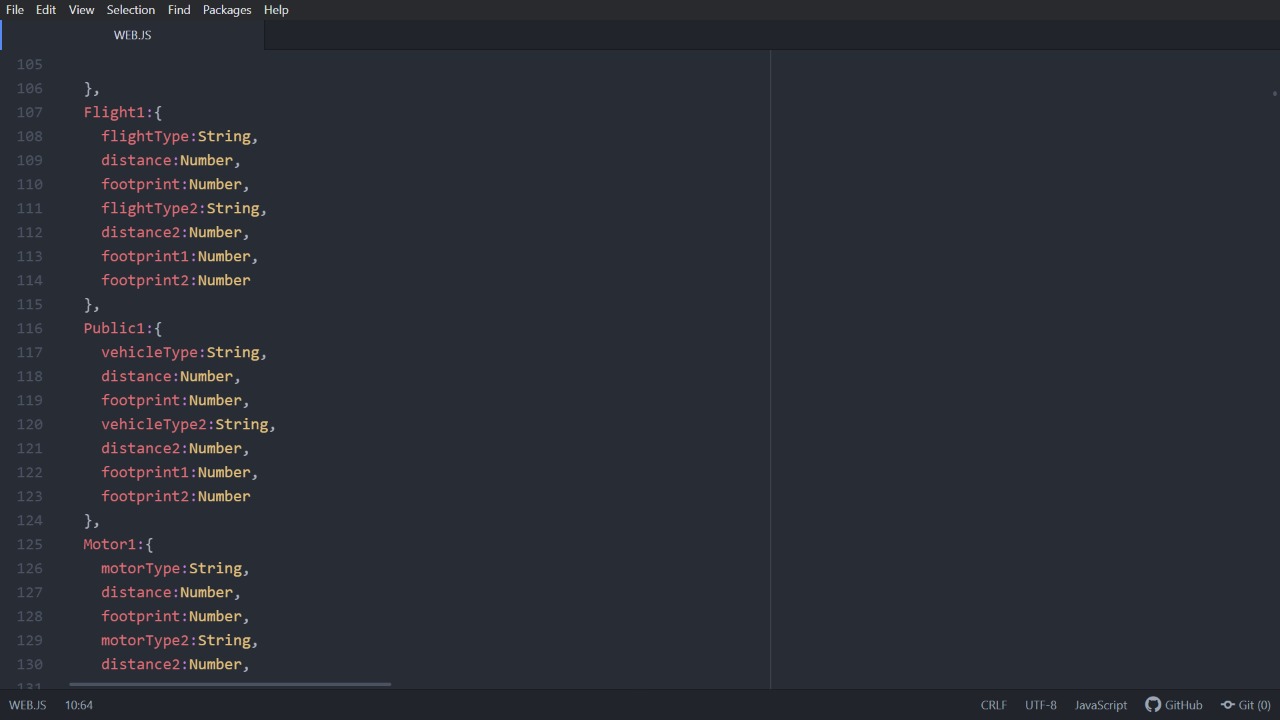
**Schemas**

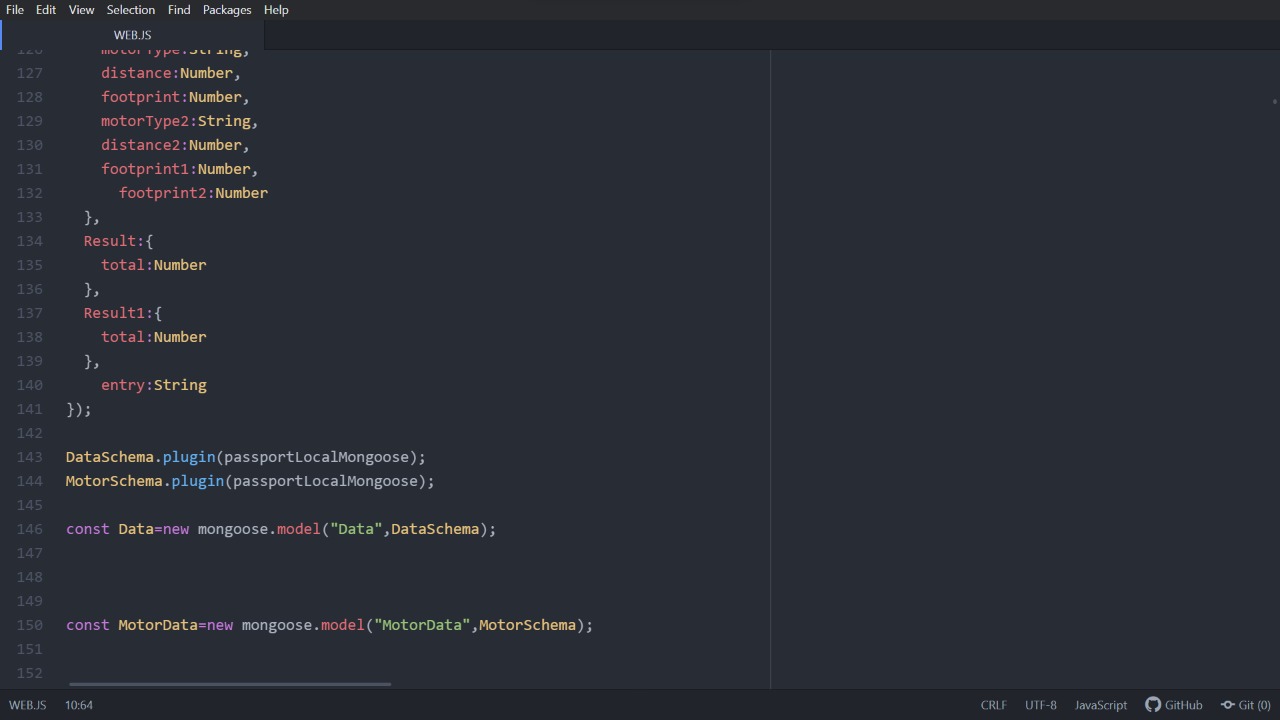


**Fig 3.2 Motor Schema**









**Fig 3.2: Data Schema**

# Chapter 4

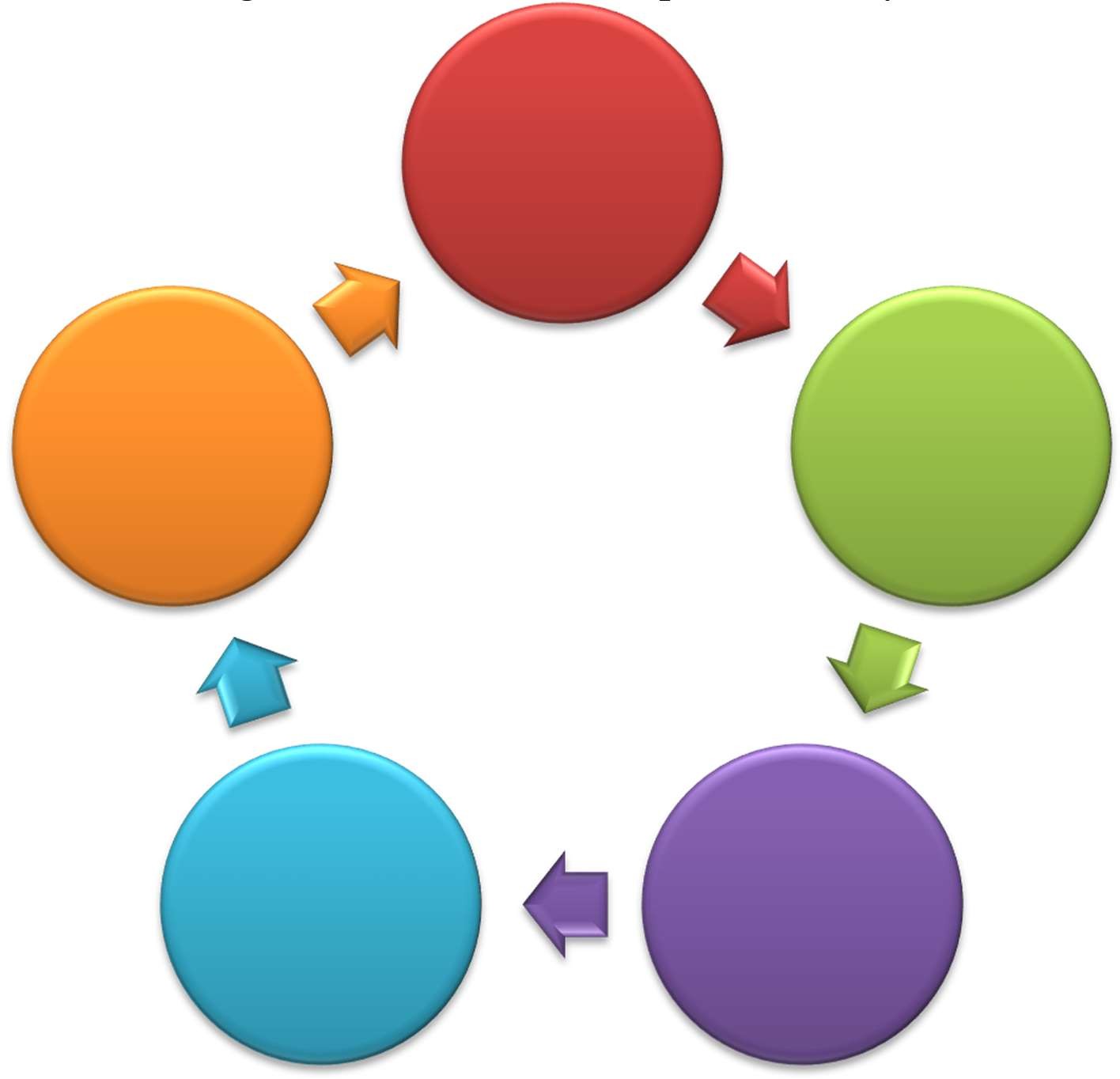
**System Implementation**

## **Explanation**

Implementation is the process of bringing a developed system into operational use and turning it over to the user. Implementation activities extend from planning through conversion from the old system to the new. There are 5 stages in the software development life cycle they are

* Planning
* Analysis
* Design
* Development
* Testing

### Figure 4.1 Software Development Life Cycle



Planning

Testing

Analysis

Development

Design

**1. Planning**

At the beginning of the Software Development Life Cycle (SDLC), planning phase was done. The questions raised in the phase are “What system we are going to build? “, “How we build the System?”, “Who will build it?” and “What are the risks associated?”, then it was decided that we are going to use HTML, CSS, Java Script, Bootstrap, jQuery, EJS at the front end, Node JS at the backend and MongoDB database. Then at the end of the phase work was divided among ourself then we started the analysis phase.

### **2. Analysis**

In the analysis phase the problems associated with the existing system was analyzed. Requirement analysis which is one of the important stages in a SDLC was done and the requirements needed to develop our project was determined.

### **3. Design**

This phase includes bringing down all the knowledge of planning and analysis phase. Based on Software Requirement Specification (SRS) the design was decided. The project should be designed in such a way that each and every people should be able to easily use the application.

### **4. Development**

In this phase the actual implementation of the project was started. First the front end was designed and after completion of each page it was being connected to the backend. Both the frontend and backend work were done parallelly.

### **5. Testing**

After development of each and every page unit testing was performed to test the functionality of the page. After integration of other pages integration testing was done to ensure that the integration of one page does not affect the functionality of the other page. At last verification testing was done to ensure that the project meets our requirements.

**Tools used for Implementation**

The tools used for development are **Visual Studio Code** which as an Integrated Development Environment, **GIT** which is an application for Microsoft Windows environments which provides an emulation layer for a Git command line experience, **chrome developer tools** which is used to find bugs in the application, **bootstrap** for frontend designing, MongoDB for storing the data, Mongoose for accessing the MongoDB, Nodejs with the help of express framework was used at the backend.

**Visual Studio Code** is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, Node.js and has a rich ecosystem of extensions for other languages.

Visual Studio Code combines the simplicity of a source code editor with powerful developer tooling, like IntelliSense code completion and debugging. First and foremost, it is an editor that gets out of your way. The delightfully frictionless edit-build-debug cycle means less time fiddling with your environment, and more time executing on your ideas.

**Web development tools** allow web developers to test and debug their code. They are different from website builders and integrated development environments (IDEs) in that they do not assist in the direct creation of a webpage, rather they are tools used for testing the user interface of a website or web application.

Web development tools come as browser add-ons or built-in features in web browsers. Most popular web browsers, such as Google Chrome, Firefox, Internet Explorer, Safari, Microsoft Edge and Opera, have built-in tools to help web developers, and many additional add-ons can be found in their respective plugin download centers.

**Nodemon** is a tool that helps develop node.js based applications by automatically restarting the node application when file changes in the directory are detected.

**Passport is authentication middleware for Node.js. Extremely flexible and modular, Passport can be unobtrusively dropped in to any Express-based web application. A comprehensive set of strategies support authentication using a username and password, Facebook, Twitter, and more.**

**Git** is a fast, scalable, distributed revision control system with an unusually rich command set that provides both high-level operations and full access to internals. Git which is an application for Microsoft Windows environments which provides an emulation layer for a Git command line experience.

**Bootstrap** is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

**EJS** is a simple templating language that lets you generate HTML markup with plain JavaScript.

**NPM** is the package manager for the Node JavaScript platform. It puts modules in place so that node can find them and manages dependency conflicts intelligently. It is extremely configurable to support a wide variety of use cases.

**Body-parser** is the Node.js body parsing middleware. It is responsible for parsing the incoming request bodies in a middleware before you handle it.

**Cookie-parser** is a middleware which parses cookies attached to the client request object.

**Express** is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node based Web applications.

**Express Session** is a session management done in node.js by using this module. It helps in saving the data in the key-value form.

**TECHNOLOGIES USED**

**Hypertext Markup Language (HTML)** is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

**Cascading Style Sheets** is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

**JavaScript**, often abbreviated as JS, is a programming language that conforms to the ECMAScript specification. JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly- bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.

**jQuery is a JavaScript library designed to simplify HTML DOM tree traversal and manipulation, as well as event handling, CSS animation, and Ajax. It is free, open-source software using the permissive MIT License. As of May 2019, jQuery is used by 73% of the 10 million most popular websites.**

**Node.js** is an open-source, cross-platform, back-end JavaScript runtime environment that executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser.

**Feature of Node.js**

* + **Asynchronous and Event Driven**: All APIs of Node.js library are asynchronous, that is, non- blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.
  + **Very Fast** − Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
  + **Single Threaded but Highly Scalable:** Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.
  + **No Buffering:** Node.js applications never buffer any data. These applications simply output the data in chunks.

Express is a popular unopinionated web framework, written in JavaScript and hosted within the Node.js runtime environment. This module explains some of the key benefits of the framework, how to set up your development environment and how to perform common web development and deployment tasks.

**Features of express:**

#### 1. Faster Server side development

#### 2. Middleware

#### 3. Advanced Routing mechanism

#### 4. Provides templating engines

#### 5. Makes debugging easier

**MongoDB** is a document-oriented NoSQL database used for high volume data storage. Instead of using tables and rows as in the traditional relational databases, MongoDB makes use of collections and documents. Documents consist of key-value pairs which are the basic unit of data in MongoDB. Collections contain sets of documents and function which is the equivalent of relational database tables. MongoDB is a database which came into light around the mid-2000s.

MongoDB is becoming so popular because of many good reasons:

1. Easy to use
2. Easy to get started
3. Great documentation
4. Free online training courses
5. A strong viable company
6. Constantly improving the software
7. Fast
8. Flexible
9. Can use lots of programming languages
10. Can do almost anything you can think of for most basic applications

**Mongoose is a JavaScript object-oriented programming library that creates a connection between MongoDB and the Express web application framework.**

# Chapter 5

**System Testing**

## **Details of extensive testing done to verify the system functionality.**

Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free.

### **UNIT TESTING**

Unit testing verification efforts on the smallest unit of software design, module. This is known as “Module

Testing”. The modules are tested separately. This testing is carried out during programming stage itself. In

these testing steps, each module is found to be working satisfactorily as regard to the expected output from the

module.

**Test Cases**

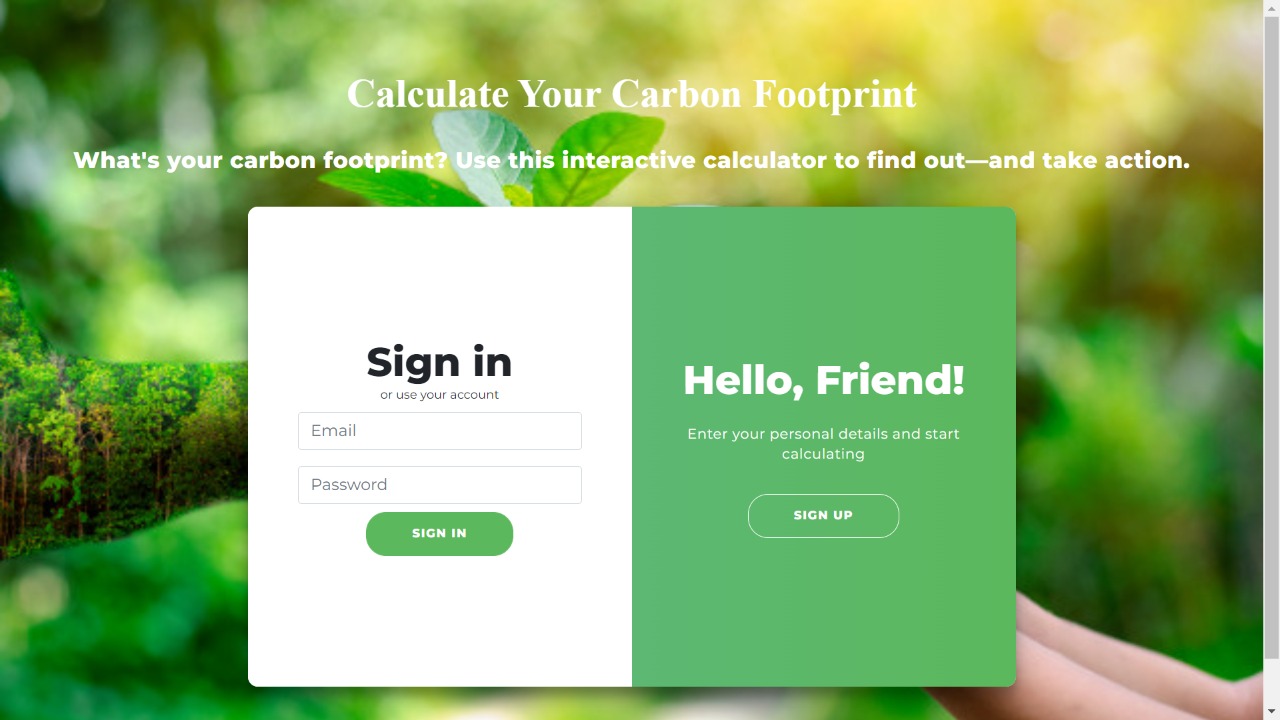
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test**  **Case ID** | **Test**  **Case** | **Steps** | **Expected Result** | **Actual**  **Result** |
| TC01 | User Sign IN | Enter user email and password and click login. | Registered users should be redirected to  their home page. | Registered user was redirected to home page. |
| TCO2 | User Sign UP | Enter Name, Email, Password and confirm the password after adding this much click signup. | User’s data should be created in database. | User’s data was created in the database. |
| TC03 | Entries | Select the entry to calculate the footprint and click “Let’s Calculate” to begin the calculation. | Redirect to the Cars Carbon Footprint Calculator page of corresponding entry. | Redirecting to the Cars Carbon Footprint Calculator page. |

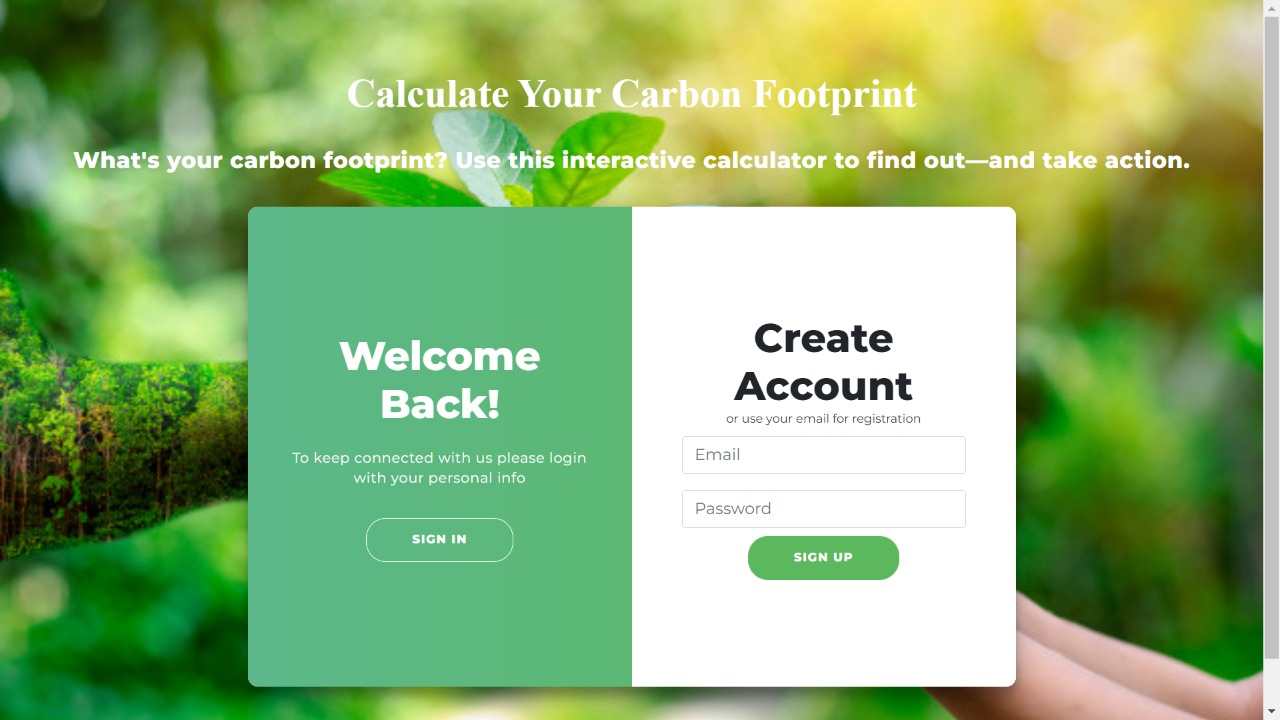
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TC04 | Inputs  (Car Type,  Distance Travelled, Mileage) | Insert the input.  After insertion click on calculate button. | Carbon footprint result will be shown along with the pie chart for comparison. | Carbon footprint result is displayed along with the pie chart for comparison. |
| TC05 | Inputs  (Flight Type,  Distance Travelled) | Insert the input.  After insertion click on calculate button. | Carbon footprint result will be shown along with the pie chart for comparison. | Carbon footprint result is displayed along with the pie chart for comparison. |
| TC06 | Inputs  (Motorbike Type,  Distance Travelled) | Insert the input.  After insertion click on calculate button. | Carbon footprint result will be shown along with the pie chart for comparison. | Carbon footprint result is displayed along with the pie chart for comparison. |
| TC07 | Inputs  (Public Transport Type,  Distance Travelled) | Insert the input.  After insertion click on calculate button. | Carbon footprint result will be shown along with the pie chart for comparison. | Carbon footprint result is displayed along with the pie chart for comparison. |
| TC08 | Previous | Click on previous button. | Upon clicking previous button, user will be redirected to previous page. | Upon clicking previous button, user is redirected to previous page. |
| TC09 | Next | Click on next button. | Upon clicking next button, user will be redirected to next page. | Upon clicking next button, user is redirected to next page. |
| TC10 | Clear | Click on clear button. | Upon clicking clear button, it will clear all the entries made by the user in the current page as well as carbon footprint . | Upon clicking clear button, clears all the entries made by the user in the current page as well as carbon footprint. |
| TC11 | Add Another Entry | Click on Add Another Entry. | On clicking another entry for present page will be created. | On clicking another entry for present page is being created. |
| TC12 | For Detailed Analysis.. Submit | Click on “For Detailed Analysis.. Submit”. | User will be redirected to the detailed result page where every carbon footprint will be displayed with other details. | User is being redirected to the detailed result page where every carbon footprint is being displayed. |
| TC13 | Logout | Click on logout button. | When clicked the user should be signed out. | When clicked the user had signed out. |

# Chapter 6

**Results**

## Screenshots of the system functionality with explanation for each screenshot.





**Fig 6.1: Sign IN/Sign UP Page**

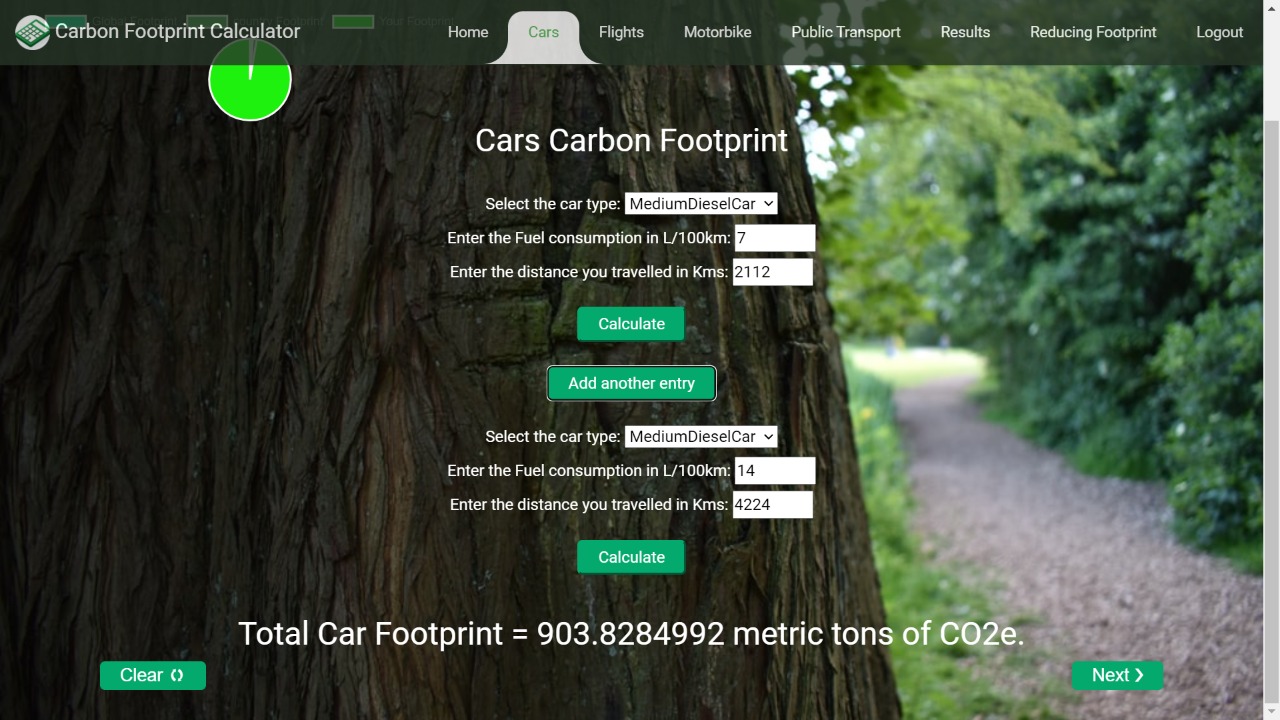
The user can register by filling the signup form. After signing up new user and his/her details are added into the database and then the page is redirected to User Sign in Page.

The user can login by entering the valid credentials in this page, if the credentials are valid, he will be redirected to the home page or else the page won’t be redirected and stays in sign in page itself. In order to login the user should be a verified user. The user can also log in using his/her gmail account.



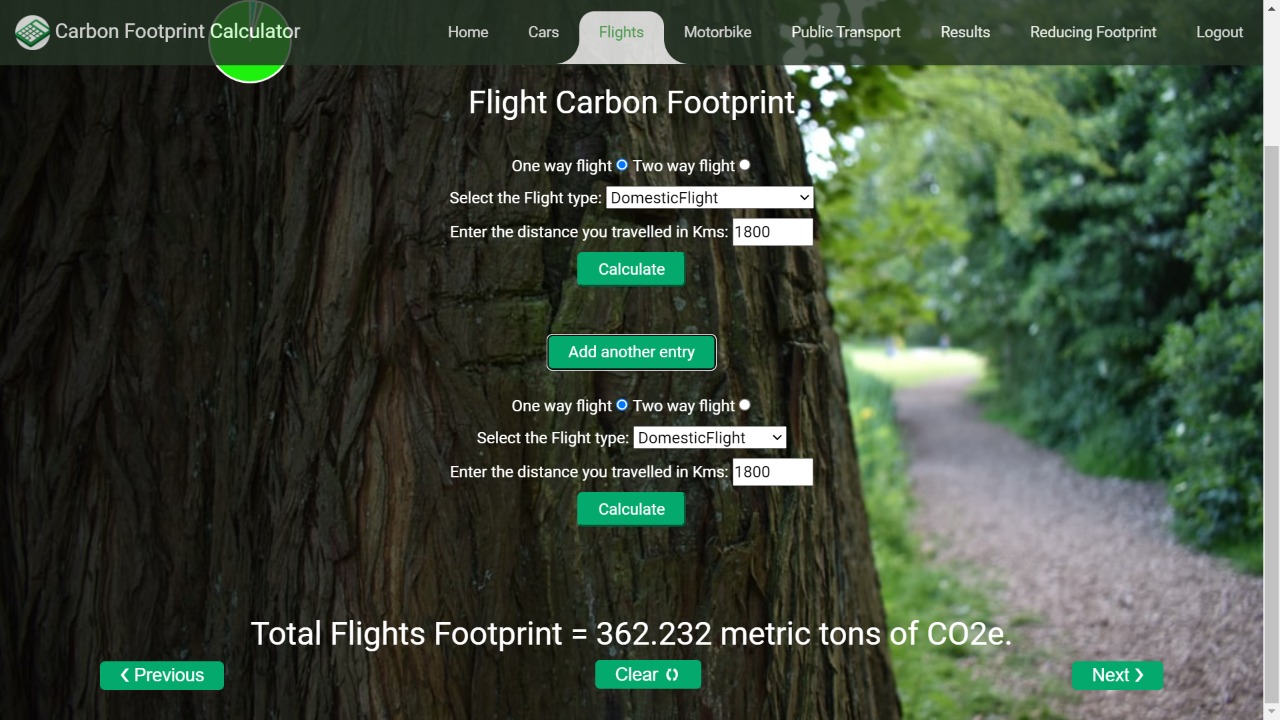
**Fig 6.2: Home page**

By clicking “Let’s Calculate” button user can start calculating their carbon footprint.



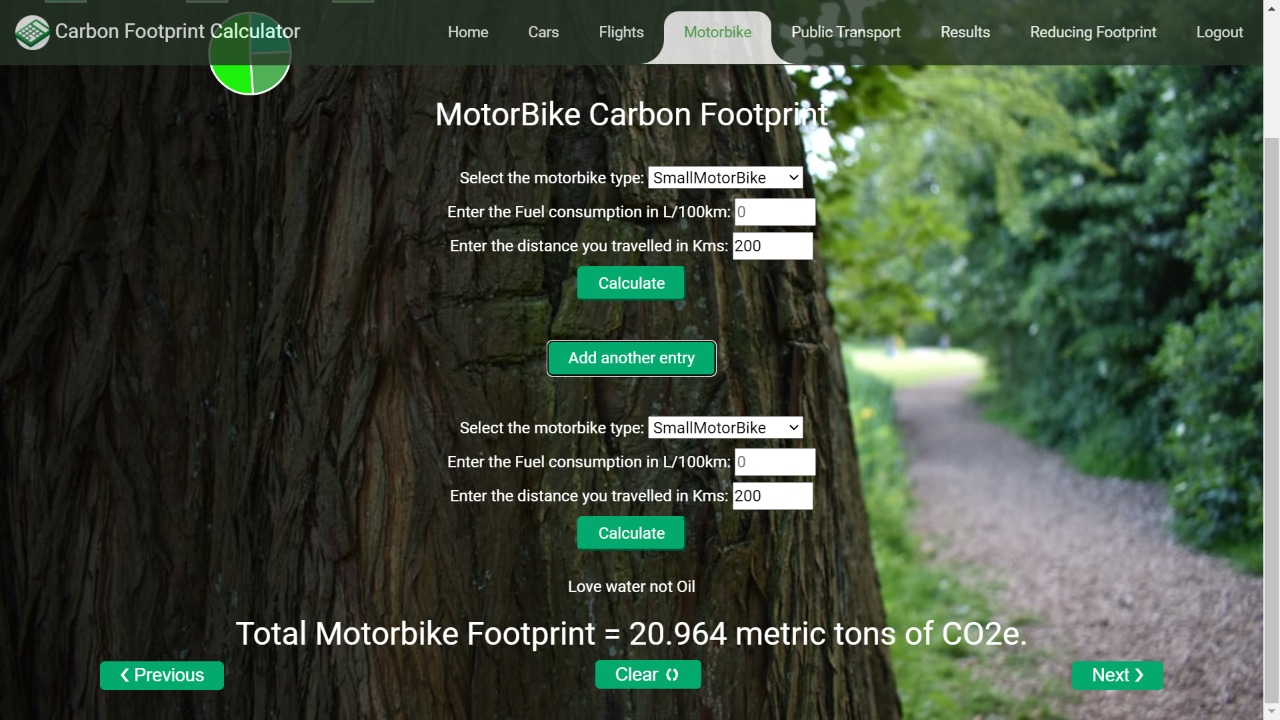
**Fig 6.3: Car Carbon Footprint**

The car calculator determines the CO2 emissions of a car that arise during a car journey. The calculation is made per car and not per person.



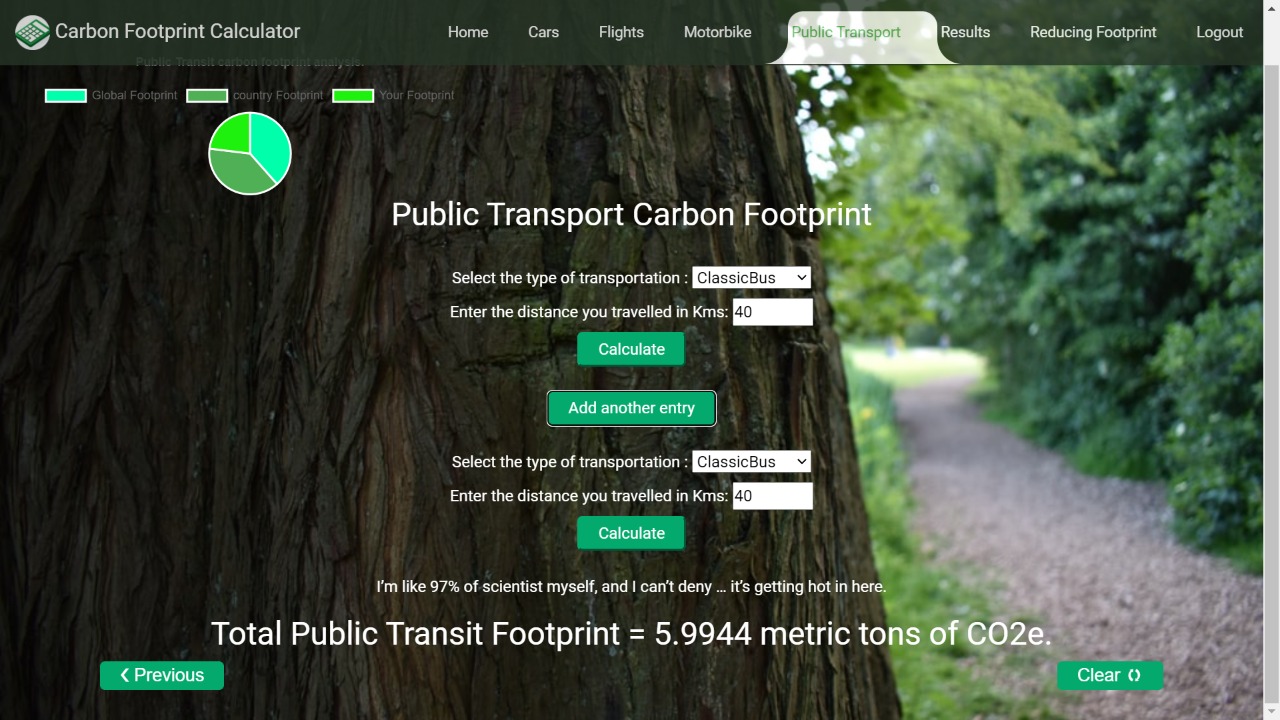
**Fig 6.4: Flight Carbon Footprint**

Use the flight calculator to determine the carbon footprint of your flight as well as the amount that is required for carbon offsetting**.**



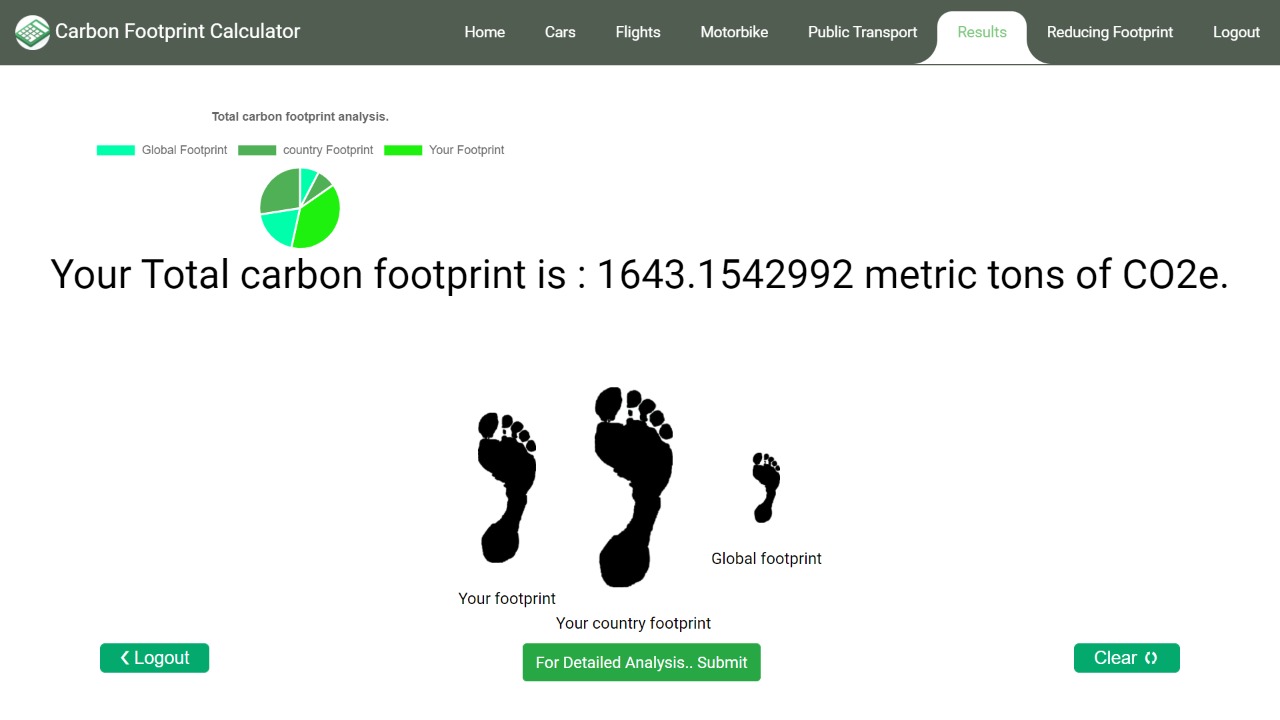
**Fig 6.5: Motorbike Carbon Footprint**

The motorbike calculator determines the CO2 emissions of a motorbike that arise during a journey. The calculation is made per motorbike and not per person.



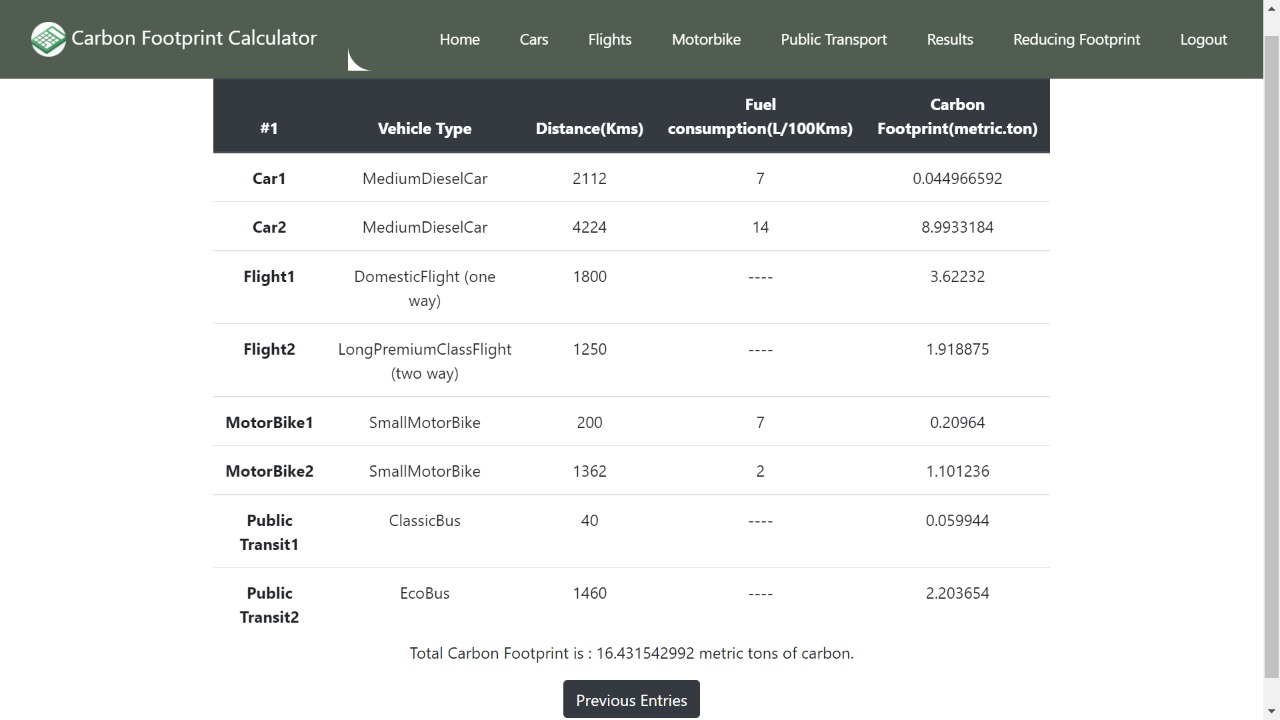
**Fig 6.6: Public Transport Carbon Footprint(Bus)**

The public transport calculator determines the CO2 emissions of a public transport(Bus,Taxi etc) that arise during a journey. The calculation is made per vehicle and not per person.



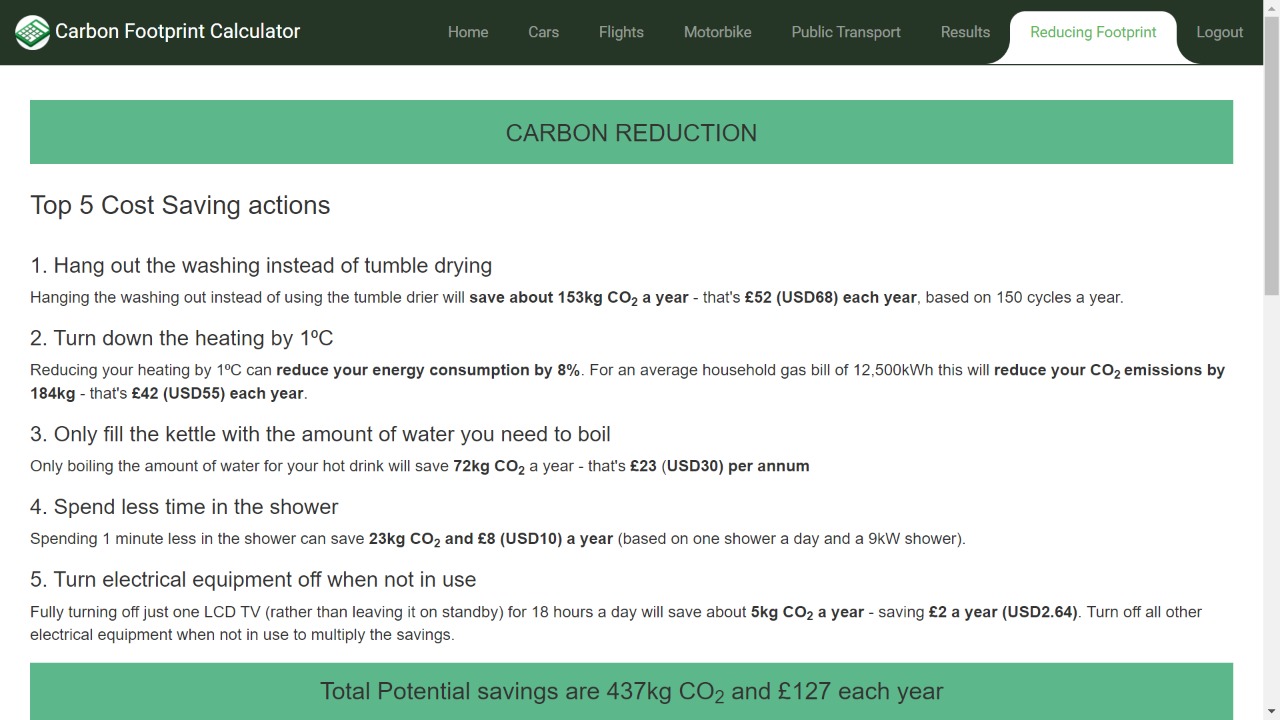
**Fig 6.7: Result**

This shows the total carbon footprint generated by the user by all means of transportation and compares the result with global and country’s carbon footprint.

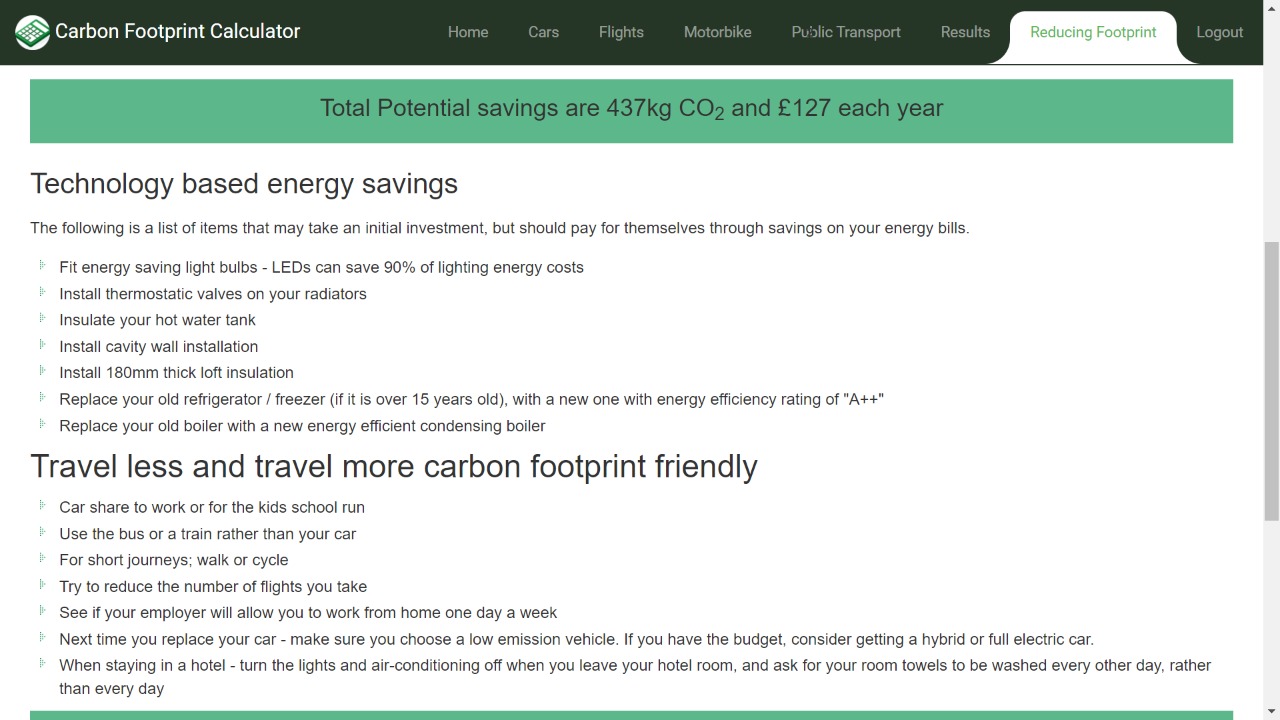


**Fig 6.8: Detailed Analysis of Result**

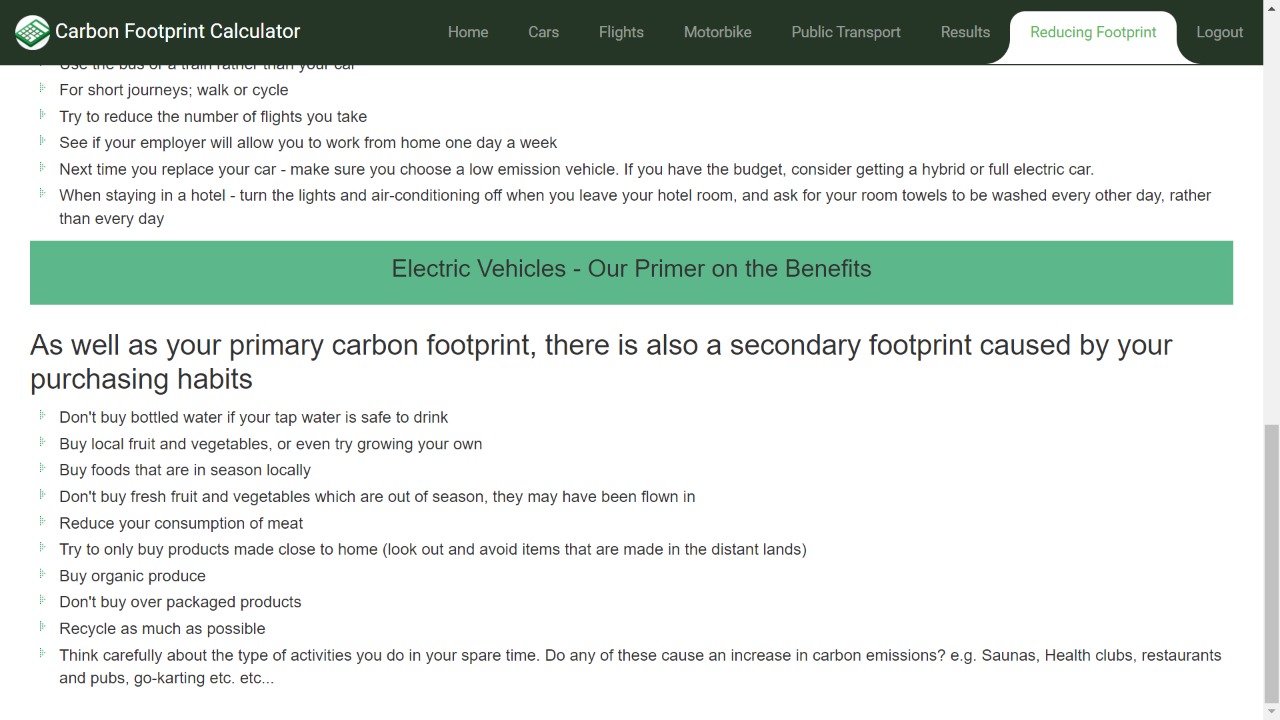
This gives a detailed explanation about the footprint generated by the user through various means of transportation.



**Fig 6.9: Reducing Carbon Footprint Techniques**



**Fig 6.10: Reducing Carbon Footprint Techniques**



**Fig 6.11: Reducing Carbon Footprint Techniques**

# 

# Conclusion and Future Enhancements

## **Conclusion**

The main objective of the proposed calculator is to provide user a basic web application where they

can enter their automobiles usage data and get their respective carbon footprint and can compare with

global footprint and country’s footprint.

**Future Enhancements**

1. To add more fields like cruise, households etc.

2. To add more input fields for user convenience.

3. Improve user interface.

4. To expand the existing calculator’s domain globally.

**Reference**

Node JS - <https://nodejs.org/en/>

Bootstrap – <https://getbootstrap.com/>

MongoDB -- <https://www.mongodb.com/>

Mongoose -- <https://mongoosejs.com/>

HTML MDN – <https://developer.mozilla.org/en-US/docs/Web/HTML>

CSS MDN – <https://developer.mozilla.org/en-US/docs/Web/CSS>

Java Script MDN – <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

jQuery – <https://jquery.com/>

Google Fonts – <https://fonts.google.com/>

Express-ejs-layout -- <https://www.npmjs.com/package/express-ejs-layouts>

Express Session – <https://www.npmjs.com/package/express-session>

Cookie Parser – <https://www.npmjs.com/package/cookie-parser>

EJS – <https://ejs.co/>

Express –<https://www.npmjs.com/package/express-session>

Body-Parser – <https://www.npmjs.com/package/body-parser>

Git – <https://git-scm.com/downloads>

Nodemon – <https://www.npmjs.com/package/nodemon>

Authentication & Authorization -- <https://www.passportjs.org/>

Icons -- <https://fontawesome.com/>