DON BOSCO INSTITUTE OF TECHNOLOGY

Bengaluru, Karnataka – 74

TEAM COUNT: THREE

PROJECT TITLE:

Carbon Footprint Tracker for Offices: A Tool to Calculate and Display Carbon Emissions for Office Operations and Suggest Sustainable Practices

TEAM LEAD NAME: B N KUSHAL

TEAM LEAD CAN ID: CAN_33198144

1) NAME: Bindu G
CAN ID: CAN_33659435
ROLE: Frontend Developer

2) NAME: Akshatha B M
CAN ID: CAN_33688521
ROLE: Backend Developer

3) NAME: B N Kushal
CAN ID: CAN_33198144
ROLE: Database Administrator

GitHub:- https://github.com/kushal1929/carbon_footprint_tracker_1

Phase 4: Performance and Deployment

1. Final Front-End

Overview:

Improving responsiveness and usability of the user interface, so that it is intuitive and easy to use for all types of users, but also so that it provides all the tools for carbon footprint tracking and analysis in an accessible and interesting way.

Implementation:

- **UI/UX Improvements:**
  - Improve the existing components with **Material-UI** or **Bootstrap** to make a clean, easy-to-use and attractive dashboard to allow users to effectively track and analyze their carbon footprint.
  - We will make the user interface aesthetically pleasing while keeping maximum functionality and ease of use.
- **State Management:**
  - Use Redux or Context API to manage session states, user inputs, and dynamic content updates across the app so that users can interact with the platform in a natural way, and their progress and inputs are reflected everywhere.
- **Real-Time Updates:**

    - Using **WebSockets** or other real-time technology, display the results of the carbon footprint analysis in real time as users log their activities, so that it looks dynamic and interactive.
    - This will make the platform feel responsive and engaging, as users will be able to see the results of their actions immediately.
- **Cross-Platform:**
    - Make sure it's responsive across desktop, tablet and mobile – that way as many people as possible can access it no matter what device they're using and they get the same great experience across all screen sizes.

---

2. Back-End Optimization

Overview:

Fix the server-side so it can handle multiple users' requests smoothly while providing secure authentication and make the API respond quickly and efficiently so users can track their carbon footprint without any delays and so all the data is processed correctly.

Implementation:

- **API Optimization:** Optimize and streamline the RESTful APIs to make it easy to retrieve carbon footprint data, log activities, and receive personalized sustainability recommendations. By optimizing the APIs, we'll be able to exchange data between the front-end and back-end faster and more reliably.
- **Real-Time Data Processing: Real-time updates** – dynamically recalculate carbon footprints based on user input. As users log their activities, their carbon footprint will be instantly recalculated, giving them immediate feedback and insight.
- **Authentication & Authorization:** Secure user access with **JWT authentication** so that only authorized users can track, store, and access their environmental impact data. This will ensure that sensitive user data remains protected while also making the login process seamless and easy.
- **Data Security & Storage:** Store user activity logs and emissions data in Uses Firebase Firestore encrypted and with strict access control policies, so that the privacy and integrity of user data is protected from unauthorized access and so that it can be securely stored for convenient retrieval.

Result:

A secure, scalable, and high-performance back-end will enable real-time carbon footprint tracking, quick and accurate data retrieval, personalized sustainability insights, and high security and data privacy.

3. Database Refinement

- **Scalable:** Users and activities are stored in sub-collections, allowing easy expansion.
- **Efficient Queries:** Indexed fields (e.g., timestamp, emission) optimize performance.
- **Flexibility:** Easily extendable with more categories (e.g., flights, electricity usage).

Since Firestore is a **NoSQL document-based database**, it follows a **hierarchical structure** using **collections** and **documents** instead of traditional relational tables. Below is the **schema structure** of the Eco Footprint Tracker.

Benefits:

- **Faster Queries** (Retrieves relevant data efficiently).
- **Optimized Reads** (Reduces Firestore read costs).
- **Better Sorting** (Avoids Firestore's "Index Not Found" errors).

Outcome

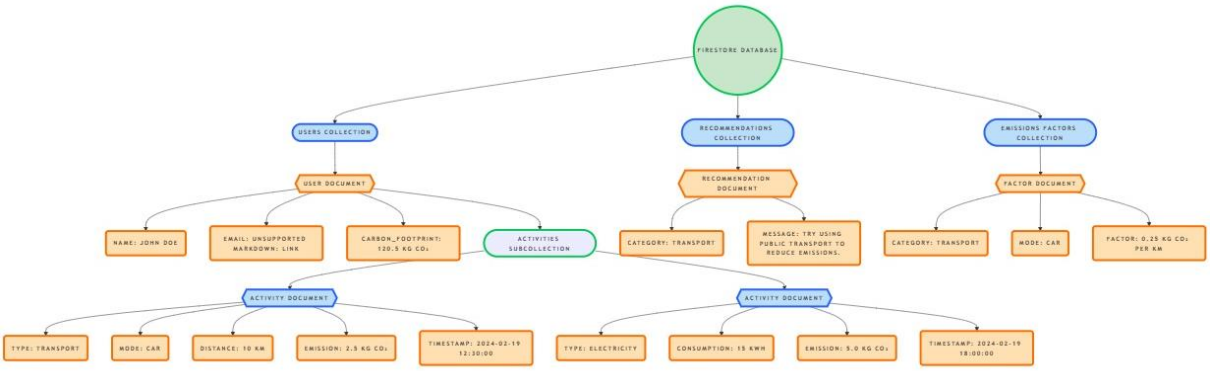
◇ **Faster Query Performance:** Optimized Firestore queries ensure rapid data retrieval, especially for user activities and emission factor lookups.

◇ **Reduced Firestore Read Costs:** Indexed queries prevent unnecessary document scans, reducing Firestore read operations and saving costs.

◇ **Efficient Filtering & Sorting:** Composite indexes allow filtering by multiple fields, such as activity type and timestamp, without performance bottlenecks.

◇ **Scalable Database Operations:** The database can handle growing user data efficiently without degrading response time.

◇ **Secure & Optimized Data Access:** Firestore security rules ensure indexed data remains protected while being easily accessible to authenticated users.



4. Real-Time Features in Eco Footprint Tracker

Overview:

Provide real-time tracking and user engagement through real-time notifications on carbon footprint changes and sustainability tips.

Implementation:

Live Carbon Footprint Tracking

Real-time updates when users log activities like transportation, energy use, or diet choices.

Notifications for Sustainability Insights

Push notifications for personalized tips on sustainability, weekly updates on progress, and carbon challenges.

Dynamic Data Updates

Sync real-time carbon footprint calculations and recommendations across multiple devices.

Outcome:

- **Real-Time Feedback** – Users get immediate feedback on their actions.
- **Higher Engagement** – Engages people to take action to reduce carbon emissions.
- **Real-Time** – Data gets synchronized without any delay between devices.

---

6. Testing and Quality Assurance

Overview:

Conduct extensive testing to ensure the **Eco Footprint Tracker** meets performance, security, and functionality benchmarks.

Implementation:

Unit & Integration Testing:

- Use Jest and React Testing Library to test individual React components.
- Ensure the correctness of calculations for carbon footprint estimations.
- Validate API endpoints to check data retrieval and storage integrity.

Load Testing:

- Use JMeter or K6 to simulate multiple users accessing the platform simultaneously.
- Ensure that the system handles large datasets, such as multiple user inputs and footprint calculations, without performance drops.

Security Testing:

- Identify vulnerabilities in user authentication and authorization.
- Ensure secure data transmission with HTTPS and encryption.

Outcome:

- A stable and secure platform capable of handling multiple users efficiently.
- Accurate carbon footprint calculations with real-time feedback.
- Secure user authentication and protected data storage.

---

7. Deployment

Overview:

Deploy the Eco Footprint Tracker to a production environment with robust monitoring and security tools.

Implementation:

CI/CD Pipelines:

- Automate deployment using GitHub Actions.
- Deploy to AWS, Google Cloud, or Heroku for seamless updates.

Hosting Services:

- Use Firebase Hosting or Netlify for front-end deployment.
- Utilize MongoDB Atlas for a cloud-hosted, scalable database.

Monitoring & Maintenance:

- Integrate New Relic or Prometheus to monitor system performance.
- Use Sentry for real-time error tracking and bug reporting.

Outcome:

- A fully deployed and accessible carbon footprint tracking application.
- Smooth user experience with minimal downtime.
- Scalable infrastructure to support an increasing user base.

Challenges and Solutions in the Eco Footprint Tracker Project

1. Data Accuracy and Consistency

Challenge:

- Assurance that carbon footprint calculations are robust and based on verified emission factors
- Handle inconsistent user inputs (e.g. incomplete or incorrect data entries)

Solution:

- Emission factors should be based on well-respected data sources (e.g. governmental or environmental organizations).
- Implement data validation checks to prevent incorrect entries.
- Pre-populate values and use auto-suggestions to help users enter valid data

2. Real-Time Updates and User Experience

Challenge:

- Making sure carbon footprint calculations update in real time as users type in information.

Solution:

- Use React state management (like Redux or Context API) to handle live updates efficiently.
- Optimize API responses using caching and indexed queries in MongoDB.
- Implement asynchronous data fetching to avoid UI blocking issues.

3. Scalability and Performance Optimization

Challenge:

- Handling increased user traffic as more users adopt the application.
- Ensuring that queries run efficiently without slowing down the platform.

Solution:

- Optimize database indexing for faster searches and queries.
- Lazy load charts and analytics to reduce initial load time.
- Implement pagination for displaying historical carbon footprint data efficiently.

4. Security and Privacy Concerns

Challenge:

- Protecting user data and ensuring privacy in carbon footprint tracking.
- Preventing unauthorized access and potential data breaches.

Solution:

- Implement JWT-based authentication for secure user logins.
- Use encryption (TLS/SSL) to protect data during transmission.
- Role-based access controls (RBAC) — Configure access restrictions on sensitive data.

5. User Engagement and Retention

Challenge:

- Encouraging users to consistently track their carbon footprint over time.
- Providing meaningful recommendations to help users lower their footprint.

Solution:

- Implement gamification features (e.g., badges, progress tracking, and rewards).
- Send personalized notifications with sustainability tips.

Outcomes of Phase 4

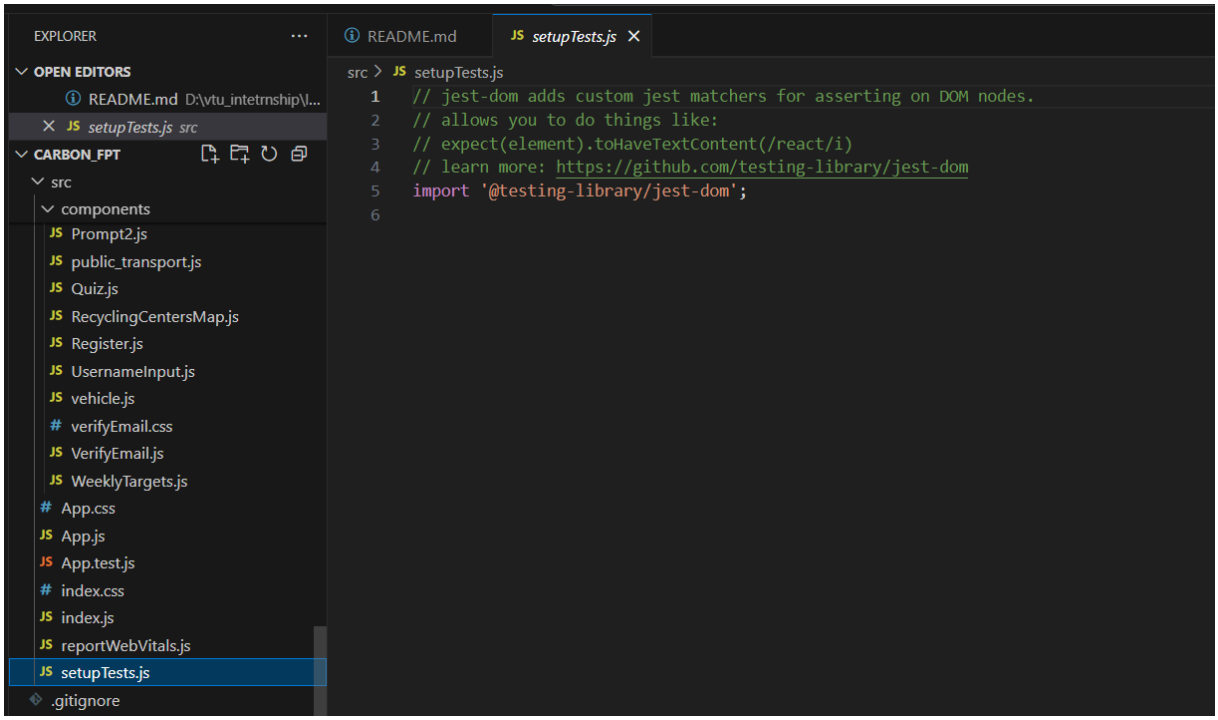
- A fully functional and user-friendly carbon footprint tracking platform.
- Secure user authentication and data storage for privacy and protection.
- Optimized architecture for seamless user experience and scalability.

Next Steps

1. Enhance Carbon Footprint Calculation Accuracy
2. Improve Personalization and Recommendations
3. Mobile App Development
4. Integrate External Data Sources

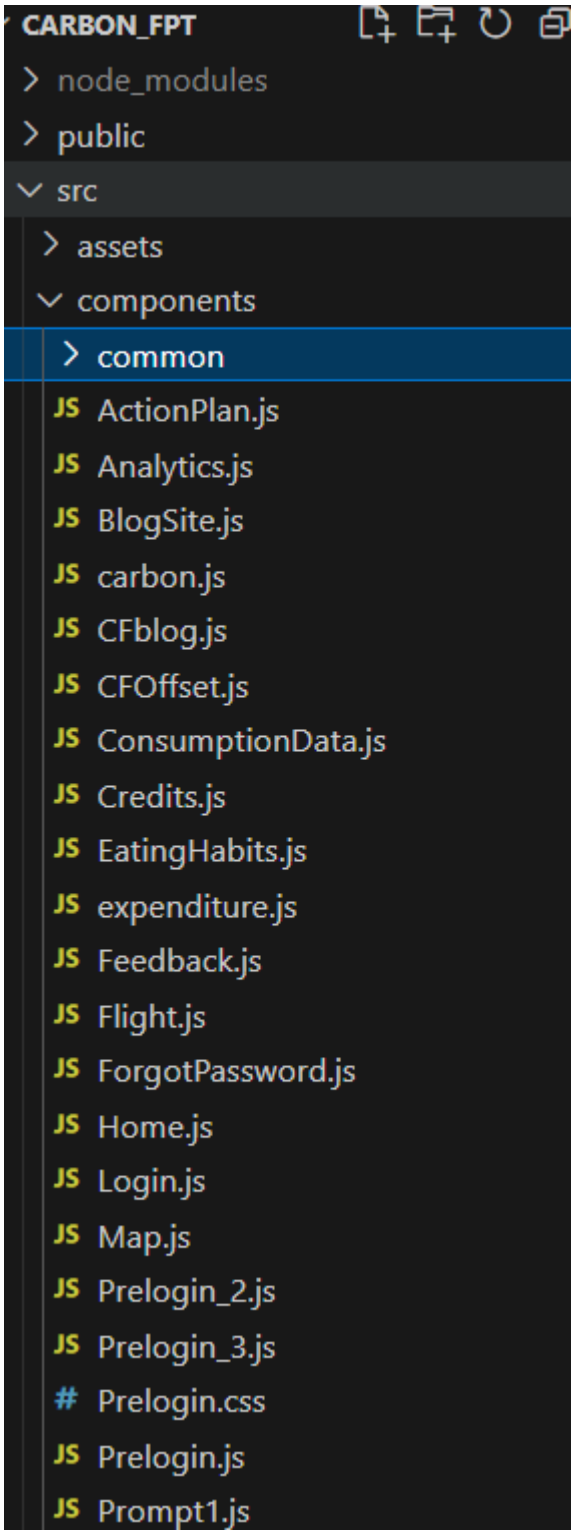
5. Monitor and Scale System
6. Conduct User Feedback Sessions

Screenshots & Code Snippets



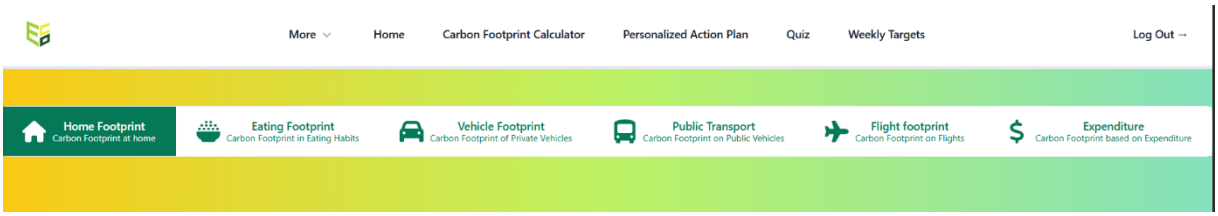

```
src > App.js > ...
1  import './App.css';
2  import Home from './components/Home';
3  import UsernameInput from './components/UsernameInput';
4  import EatingHabits from './components/EatingHabits';
5  import CarbonFootprintCalculator from './components/carbon';
6  import CarbonFootprintCalculatorVehicle from './components/vehicle';
7  import CarbonFootprintCalculatorPublicVehicle from './components/public_transport';
8  import CarbonFootprintCalculatorExpenditure from './components/expenditure';
9  import Flight from './components/Flight';
10 import VerifyEmail from './components/VerifyEmail';
11 import { getFirestore,collection,query,where,getDocs } from 'firebase/firestore';
12
13 import {
14   Routes,
15   Route,
16   useNavigate
17 } from "react-router-dom";
18 import { useState } from 'react';
19 import { signInWithEmailAndPassword, createUserWithEmailAndPassword, getAuth ,sendEmailVerification,fetchSignInMethodsForEmail,deleteUser
20 import { app } from './firebaseconfig';
21 import Prelogin from './components/Prelogin';
22 import Login from './components/Login'
23 import Register from './components/Register';
24 import ForgotPassword from './components/ForgotPassword';
25 import ConsumptionData from './components/ConsumptionData';
26 import ActionPlan from './components/ActionPlan';
27 import Quiz from './components/Quiz.js';
28 import Feedback from './components/Feedback';
29 import Analytics from './components/Analytics';
30 import Map from './components/Map.js';
31 import Credits from './components/Credits';
```

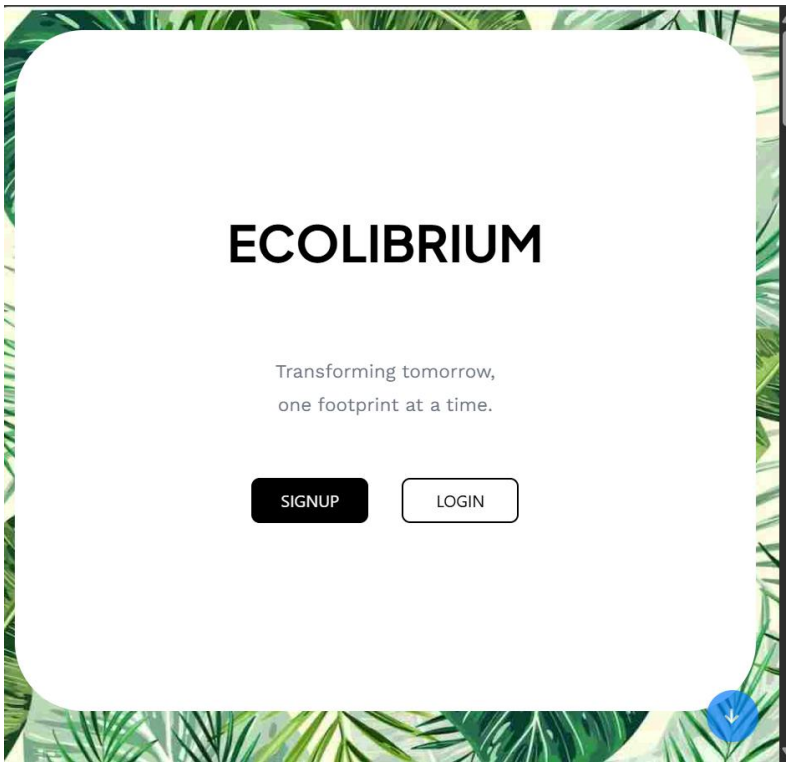
```jsx
import React, { useEffect, useState, useRef } from 'react';
import { getFirestore, doc, getDoc } from 'firebase/firestore';
import html2canvas from 'html2canvas';
import './Tailwind.css';
import Header from './Header';
import LoadingSymbol from "./LoadingSymbol";

const UserDataCard = () => {
  const [userData, setUserData] = useState(null);
  const [username,setUsername] = useState(null);
  const cardRef = useRef(null);

  useEffect(() => {
    const fetchUserData = async () => {
      try {
        // Get the username from session storage
        const username = sessionStorage.getItem('Username');
        setUsername(username);

        // Fetch carbon score and global rank from Firestore
        const db = getFirestore();
        const currentMonthYear = new Intl.DateTimeFormat('en-US', { month:

        // Fetch carbon score from users/{username}/Total/{currentMonthYea
        const userDocRef = doc(db, 'users', username, 'Total', currentMont
        const userDocSnapshot = await getDoc(userDocRef);
        const carbonScore = userDocSnapshot.exists() ? userDocSnapshot.dat

        // Fetch global rank from Average/{currentMonthYear}/Leaderboard/{
        const leaderboardDocRef = doc(db, 'Average', currentMonthYear, 'Le
        const leaderboardDocSnapshot = await getDoc(leaderboardDocRef);
        const rankCarbonFootprint = leaderboardDocSnapshot.exists() ? lead
        const totalUsers = leaderboardDocSnapshot.exists() ? leaderboardDo
        const globalRank = rankCarbonFootprint !== 'Unranked'? `${rankCarb
```



src > components > JS Map.js > ...

```jsx
import React from "react";
import "./common/Tailwind.css";
import RecyclingCentersMap from './RecyclingCentersMap';
import Header from "./common/Header";

export default function Map(){
    return(
        <>
            <Header />
            <div className='flex flex-col justify-center items-center h-[90vh] w-full gap-x-6 px-2 sm:px-[10%] items-stretch '>
                <div className=' mx-auto flex mt-[3%] pb-4 w-fit bg-gradient-to-r from-green-300 via-blue-500 to-purple-600 bg-clip-text
                Recycling Centers near you !
                </div>
                <p className="mt-[2%] mx-auto">Note: You may have to zoom out</p>
                <div className="flex justify-center items-center w-full h-full px-2 sm:px-5 mt-[1%] mb-[5%]">
                    <RecyclingCentersMap />
                </div>
            </div>
        </>

    )
}
```





Key Features

Carbon Footprint Calculator
Detailed montly calculation based on lifestyle factors like transportation, energy use, and diet.

Personalized Sustainability Plan
Tailored action plan to reduce carbon emissions, empowering users with achievable steps.

Interactive Map for Recycling Locations
Locate nearby recycling facilities for convenient waste management.

Competitive Leaderboards
Engage in friendly competition, track progress, and motivate peers through challenges and leaderboards.

Private Vehicle Carbon Footprint

Note: Please fill in the details once a month

Mileage: In km/L

Select Fuel Type: Petrol

Car: Distance in km

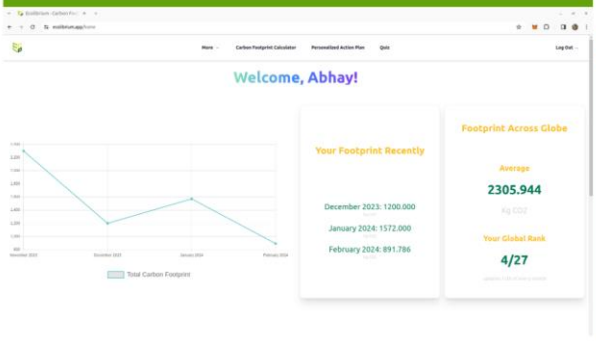
Motorcycle type: Average Motorcycle

Motorcycle: Distance in km

Calculate

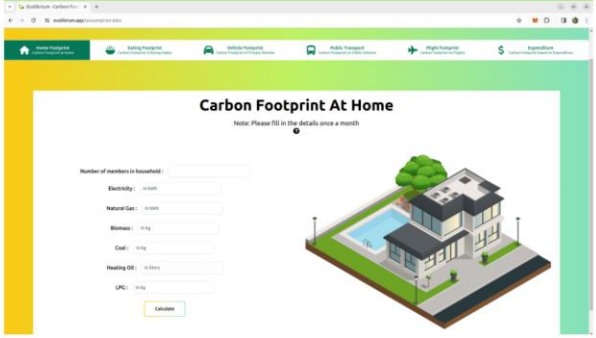
User-Friendly Interface

Our user-friendly interface ensures easy navigation and accessibility, making sustainability engaging.



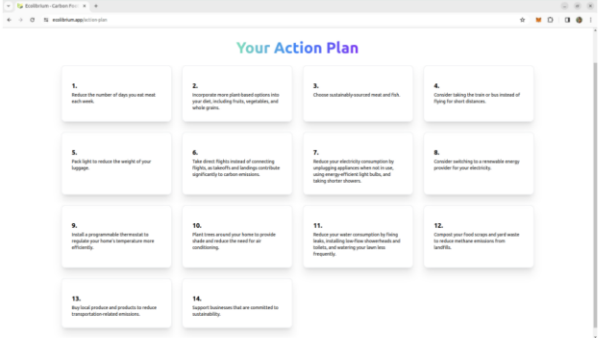
Carbon Footprint Calculator

Detailed montly calculation based on lifestyle factors like transportation, energy use, and diet.



Personalized Sustainability Plan

Tailored action plan to reduce carbon emissions, empowering users with achievable steps.



Competitive Leaderboards

Engage in friendly competition, track progress, and motivate peers through challenges and leaderboards.