Methodology and Design

Final Project

IT41028

AI-Powered Personalized Environmental Impact Tracker and Advisor

GROUP MEMBERS:

ITBIN-2211-0104

ITBIN-2211-0268

ITBIN-2211-0115

1.0 Analysis

1.1 Introduction

This chapter provides a detailed analysis of the AI-Powered Personalized Environmental Impact Tracker and Advisor. It examines the system's intended users, operational environment, and the underlying problem it aims to address. The analysis presents the feasibility of the proposed solution, explains the fact-finding techniques applied in understanding user needs, and outlines the functional and non-functional requirements that will guide the system's development. It also includes a profile of the expected user base, describing their behavior patterns, technical capabilities, and usage frequency. The chapter concludes with a description of the software development methodology selected to ensure that the system's objectives align with its technical implementation.

1.2. Feasibility Study.

The feasibility analysis carried out shows that the proposed system is feasible along operational, technical, economic, and schedule dimensions. A preliminary survey conducted during the requirement-gathering phase indicated that users are increasingly interested in understanding their personal carbon footprint and actively seek tools that help them assess their environmental impact. This again establishes clear operational feasibility since the system directly addresses an identified and validated user need. Technical feasibility is assured by the adoption of widely available, well-documented, and compatible machine learning and web development technologies like Python (Foundation, 2025), Flask (Projects, 2025), React.js (Meta, 2025), and MongoDB (Inc., 2025); these combined provide a stable and scalable basis for system development.

From the economic perspective, the system is very viable since it relies on open-source technologies primarily, thus negating major licensing fees. The main expenses are hosting and maintenance costs. The estimated cost of hosting on the cloud ranges between USD 5–15 per month, which translates into approximately Rs. 1,540–4,620. The costs for maintenance, Server monitoring, updates, and periodic model retraining, have been estimated to lie between USD 10–20 per month (Rs. 3,080–6,160). In cases where the system needs to scale up owing to high user activity, the projection of costs increases only to around

USD 20–30 per month, or approximately Rs. 6,160–9,240. This ensures the long-term affordability of the system.

Schedule feasibility is ensured through a structured development plan, which divides the project into sequential phases involving the gathering of requirements, data preparation, model development, interface design, and back-end integration along with testing. Such a phased timeline allows for better resource allocation and completion of all aspects of the project on time.

1.3. Fact-Finding Techniques.

Primary and secondary fact-finding techniques were combined to provide accurate and actionable information for the definition of the system requirements. The primary data was obtained through a structured online questionnaire, "Sustainable Lifestyle & Carbon Footprint Survey Form," conducted as a preliminary survey among 30 students from the targeted demographic. This yielded a 100% response rate, providing full participation and a very reliable representation of the students' sustainability behaviors, awareness level, and expectations. The survey consisted of multiple sections that captured detailed lifestyle attributes, such as weekly use of public, private, and non-motorized transport; frequency of consuming meat, dairy, and plant-based meals; daily electricity usage for study or work; level of familiarity with the concept of a carbon footprint; and preferred system features such as visual dashboards, personalized recommendations, notifications, and behavioral tracking.

Sample Questions extracted from the survey instrument included:

- “How familiar are you with the concept of a carbon footprint?”
- “How many days per week do you use public transportation, private vehicles, or non-motorized travel?”
- “What type of meals do you consume most frequently (meat-based, dairy-based, plant-based)?”
- “Which features would you prefer in a sustainability tracking system (graphs, progress indicators, personalized tips, reminders)?”

Analysis of the responses provided several key findings that informed the system requirements directly. About 80% of the respondents wanted to be able to monitor their carbon footprint, 70% preferred getting personalized recommendations, and 65% preferred visual dashboards with graphs

rather than text summaries. Other common issues mentioned included how to access correct emissions data, guidance on how to decrease personal carbon footprint, and tools that could track past behaviors. These findings provided necessary evidence in designing features of the system, usability requirements, and user support mechanisms.

Other main fact-finding techniques included semi-structured interviews with two experts in environmental studies; these provided very useful insights into how to choose scientifically valid emission factors and design recommendations that would meaningfully engender sustainable behavior. Observational studies were also conducted to assess how users interact with existing carbon footprint tools such as the WWF Footprint Calculator (WWF, 2021) and the My Planet App. These observations led to the identification of various limitations, including limited personalization, limited long-term behavioral tracking, and a lack of predictive or AI-driven insights-all observations that informed the design considerations of the proposed system.

Secondary data were collected by analyzing the IPCC's (IPCC, 2023), WWF (WWF, 2021), and carbonfootprint.com (Carbonfootprint.com, 2025) validated datasets on emission factors. These datasets represented scientifically sound numerical baselines for calculating emissions and training the machine-learning model that would form part of the proposed system. Comparison with various other sustainability applications identified several limitations, such as insufficient personalization, incomplete tracking, and limited recommendation systems. These limitations served to further guide the development of a system that would enable users to obtain accurate, personalized, actionable insights into sustainability through intuitive dashboards and enhanced data-tracking capabilities.

1.4. Requirement Analysis.

A preliminary survey conducted with 30 students and young professionals was used to understand the intended users' characteristics, expectations, and usage patterns. The responses indicate that most of the participants fall between the age brackets of 18-34 years, while there were limited responses from participants over 35 years. Most respondents were university students or early-career professionals, which is also representative of the typical demographic likely to adopt digital tools for sustainability. Technical proficiency among participants ranged from basic to high, where most users are comfortable with apps and other digital platforms. Familiarity with the concept of carbon footprint varied in a number of ways, although the majority reported being slightly to

moderately aware of the principles of sustainability and currently engaged in occasional behaviors of an environmentally responsible nature, including recycling, reducing energy consumption, or choosing mindful transportation options.

From these results, the target group of users can be characterized as eco-aware people in the age range of 18-45 years who have at least basic digital skills and are interested in learning more about their ecological footprint. The main needs are being able to fill in lifestyle data, obtaining a personalized carbon footprint estimate, having sustainability progress reflected over time, and being presented with recommendations for lowering one's carbon footprint in clear and actionable form. Users will access this system during free-after-work time-more specifically, evenings and weekends-when they can ponder how they lead their lives. According to the feedback data in the survey, this should happen approximately three or five times weekly, indicating consistent engagement rather than erratic interaction.

The projected number of system users is based directly on survey participation and the demographic characteristics of those responding. Given that 30 people participated in this preliminary study, adoption at launch is anticipated to be in the range of 60 to 80 users, as the tool is likely to be shared within peer groups, academic settings, and professional environments. Later-phase scaling to roughly 150 to 200 users is anticipated as awareness increases and the organic growth common to many sustainability applications takes hold. This projection is supported by strong interest in features such as graphical progress tracking, personalized insights, and trend analysis-common hallmarks of high-engagement sustainability tools. This implies that the intended users of the system require a simple, intuitive, informative, and visually supportive platform to gain insight into their environmental impact, track changes in their behavior, and acquire sustainable habits. The findings from the preliminary survey form a robust foundation for defining the system's functional and non-functional requirements, as well as ensuring the design meets actual user expectations and usage behavior.

1.4.1. Success Factors

In general, several preconditions must be achieved to run this AI-Powered Personalized Environmental Impact Tracker and Advisor successfully. Users need a stable internet connection and a modern web browser to interact with the system efficiently since the application is purely web-based. Since the accuracy of calculations of carbon footprint depends on complete, honest,

and consistent lifestyle data from users, good user input is crucial for the system's success. Scientifically validated emission factor datasets provided by reputed organizations, such as IPCC (IPCC, 2023), WWF (WWF, 2021), and carbonfootprint.com (Carbonfootprint.com, 2025), are necessary to guarantee the accuracy of the prediction output of the machine-learning model.

It needs to meet security and privacy preconditions: secure hosting of the application, encrypted transport of data, password hashing, and correct session management should guarantee that user data is not compromised. Continuous updates, monitoring, and maintenance must be performed for continued availability, integrity of the system, and reliability of performance. The system must be used regularly by its users-several times a week-to make sense of any trend analysis, personalized recommendations, and progress tracking over time.

Finally, the success of the system depends on sufficient backend resources and cloud infrastructure that can handle multiple parallel users, allow fast API responses and ensure reliable data storage, as well as uninterrupted dashboard visualization. These preconditions will ensure the system delivers accurate insights, is scalable, and reaches its main goal of promoting sustainable user behavior.

1.5. Functional Requirements.

Must-Have Requirements

1. **User Registration & Authentication**
    - The system needs to allow users to create accounts with a valid email and password.
    - Authentication requests should be processed within ≤ 3 seconds.
    - Any user input should be validated before form submission.
2. **Lifestyle Data Input**
    - It allows users to input their weekly transportation habits, dietary patterns, electricity usage, and waste management information.
    - All lifestyle information should be validated and stored securely in the system.
3. **Carbon Footprint Prediction**

- The model must generate predictions of carbon footprint with an ≥ 85% accuracy based on validation testing.
- Predictions should be returned within ≤ 4 seconds.

4. **Dashboard Visualization**
    - The system must provide a dashboard of the emissions through clear and understandable visualisations.
    - The dashboard elements should be loaded within 3 seconds for both desktop and mobile devices.
5. **Tracking Historical Data**
    - It must save previous predictions made.
    - Users should be allowed to see past data through chronological graphs showing how their environmental impact has changed.

Should-Have Requirements

6. **Personalized Recommendations**
    - It should make specific sustainability recommendations, tailored to user lifestyle patterns and emission outputs.
    - Recommendation logic should follow the user's preference indicated in the preliminary survey.
7. **Responsive Interface**
    - It should support interfaces that can adapt seamlessly to different screen sizes and resolutions for consistent usability across devices.

Could-Have Requirements

8. **Gamification Features**
    - Optional progress badges, milestone notifications, or weekly reminders could be implemented to enhance user engagement.
    - These features should only be developed if time and resources permit.

Won't-Have Requirements

9. **Smart Home Device Integration**

- The system will not integrate with smart home devices such as IoT sensors or automated energy trackers in this version.

10. Social Media Sharing

- Because of privacy concerns and less relevance to the surveyed users, sharing user results on social media will not be included.

1.6. Non-Functional Requirements.

1. Performance Requirements

- The system must return carbon footprint predictions within 2 seconds for typical requests.
- This threshold is justified because user satisfaction declines after 3 seconds, and surveyed students prefer fast, interruption-free interactions.
- The system must handle peak usage without noticeable delays during lifestyle data submission and dashboard loading.

2. Usability Requirements

- The interface must allow users to complete the lifestyle data-entry process within 3–5 minutes without external assistance.
- Navigation must be intuitive, with clear labels, tooltips, and structured input sections to support students with moderate sustainability knowledge.
- The design must comply with general usability principles, including consistency, clarity, and accessibility.

3. Reliability Requirements

- The system must maintain 99% uptime, appropriate for a non-critical but regularly accessed sustainability tool.
- Scheduled maintenance windows must be communicated in advance to ensure predictable availability.
- System components should support fault-tolerant behavior to reduce unexpected outages.

4. Security Requirements

- All passwords must be stored using secure cryptographic hashing such as bcrypt.
- All communication between client and server must use HTTPS encryption (Projects, 2025).
- Sensitive lifestyle data must follow secure data-handling procedures, including restricted access and proper session management.

5. Scalability Requirements

- The system must support at least 100 concurrent users without performance degradation.
- The architecture must allow smooth expansion for future institutional or public-level deployment.

6. Maintainability Requirements

- The system must be developed using modular, well-documented code to enable easy updates and bug fixes.
- Version control practices must be followed to support collaboration and long-term maintainability.
- System configuration must allow new features to be added with minimal disruption.

7. Compatibility Requirements

- The system must operate correctly on major browsers, including Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari**.**
- The interface must function reliably on both Android and iOS mobile devices.

8. Portability Requirements

- The system must run consistently across desktop, tablet, and mobile devices.

- No installation should be required; the system should run entirely within a browser environment.

9. Disaster Recovery Requirements

- Automated daily backups must be carried out and stored securely.
- Complete system restoration must be possible within 24 hours in the event of data loss, failure, or corruption.
- Backup procedures must be verified periodically to ensure reliability.

1.7. User Roles & Privileges

It provides two main categories of users: Standard Users and Administrators, with different privileges and responsibilities assigned to them. Standard Users represent the general public who will use the system to calculate their carbon footprint, visualize data, read personalized recommendations, and examine their progress over time. They will be able to edit their lifestyle inputs, manage their profile information, and see historical data, but they cannot view or change any setting within system-level configurations. The Administrators have special privileges and are responsible for managing datasets of emission factors, the performance of the system, the machine learning model, and the user accounts in the event of misuse or system error. Administrators can also access aggregated statistics of system usage to assess user engagement and environmental impact trends. These separate roles ensure security, data integrity, and efficient management of the application.

1.8. Test Strategy (Testability)

Testability was built into the system design from the outset, allowing for the effective validation of all components comprising the AI-powered carbon footprint tracking system. For example, a modular system architecture separates the front-end interface, API layer, machine learning model, and database as independent units, enabling testing of each module independently through unit and integration tests. The clear data flow structures, standardized API endpoints, and well-defined inputs and outputs are designed to facilitate black-box and white-box testing. Moreover, logging mechanisms built into the web application and machine learning module trace the occurrence of

errors during runtime, thus making debugging and regression testing more efficient. The system design includes rules for validation against all user inputs for ensuring that test scenarios simulate both valid and invalid data. This ensures a testability-focused approach whereby the final implementation can be thoroughly evaluated in terms of accuracy, reliability, and usability.

1.9. Methodology for System Development.

This project adopts the Agile Scrum methodology (Sutherland, 2020) because it provides an iterative, feedback-driven structure that aligns well with the evolving nature of machine-learning development and user-centered system design. Scrum is preferable to other Agile approaches such as Kanban or Extreme Programming (XP) because it offers time-boxed sprints, clearly defined roles, and structured ceremonies that support experimentation, regular validation, and continuous improvement. Unlike Kanban, which focuses on continuous flow without fixed iteration cycles, Scrum's sprint-based approach allows systematic retraining, evaluation, and refinement of machine-learning models. Similarly, XP emphasizes practices such as pair programming and test-driven development, which are less suited to data-driven experimentation where model performance often depends on iterative preprocessing, tuning, and error analysis rather than strict code-first design.

Development is organized into two-week sprints, each beginning with Sprint Planning, where tasks such as data preprocessing, feature engineering, model training, backend API development, frontend implementation, unit testing, and integration testing are selected and prioritized. Sprint Planning ensures that the team sets achievable goals while accounting for dependencies between ML components and user interface requirements. Daily Stand-ups are held to provide short updates on progress, identify obstacles, and coordinate task allocation. This continuous communication is especially valuable for machine-learning work, where challenges such as data imbalance, inconsistent model performance, or unexpected drops in accuracy must be identified and addressed quickly.

At the end of each sprint, a Sprint Review is conducted to demonstrate the completed increment, for example, improved prediction accuracy, a new dashboard module, or enhanced data-entry workflow-to supervisors or peers for feedback. This structured review cycle aligns with Scrum's emphasis on stakeholder involvement and enables frequent validation of the model's effectiveness and usability. Following the review, a Sprint Retrospective allows the team to reflect on the sprint,

evaluate which practices worked well, and identify workflow adjustments for the next cycle. Retrospectives are particularly useful for machine-learning projects where the team may need to adjust preprocessing strategies, tune hyperparameters differently, or reorganize data pipelines based on outcomes from the previous sprint.

In an academic context, Scrum roles are adapted to fit a student project structure. One team member acts as the Product Owner, responsible for maintaining and prioritizing the Product Backlog based on supervisor guidance, user survey results, and project milestones. Another team member assumes the role of Scrum Master, facilitating meetings, ensuring adherence to Scrum practices, and mitigating blockers such as unclear requirements or technical issues. The remaining team members function as the Development Team, collaboratively implementing features, training and validating models, building the user interface, and conducting testing activities.

By providing adaptive planning, structured evaluation cycles, and frequent opportunities for improvement (al., 2001), Scrum effectively supports the uncertainties and exploration nature of machine-learning development. Its iterative structure ensures that both the predictive model and user-facing components evolve logically over time, resulting in a more accurate, usable, and reliable sustainability-tracking system.

2.0 Design

2.1 Introduction

This chapter outlines the requirements identified in phase II of the analysis and provides structured design architecture for the AI-Powered Personalized Environmental Impact Tracker and Advisor. The work analyzes different design approaches, rationalizes the selected approach for implementation, and presents detailed models such as case diagrams, class diagrams, sequence diagrams, activity workflow maps, process workflow patterns, or database structure structures. Although the diagrams are not present in this version, some placeholders have been added to indicate their placement in the final report. It then proceeds with more detailed considerations of interface design and finally discusses the hardware requirements for the system.

2.2. System Design Process.

During the design phase, three potential options were evaluated. Initially, the development was tailored to use Python (Foundation, 2025) for machine learning, Flask (Projects, 2025)for backend logic, React.js (Meta, 2025) for frontend functionality, and MongoDB (Inc., 2025) for data storage. This led to its own unique approach. It provided the greatest degree of adaptability, complete control over system behavior, and was well-integrated into the predictive model. Another approach aimed at integrating with external carbon footprint APIs and prioritizing frontend visualization was proposed. The. The project's focus was on personalization, but this approach lacked the necessary features for AI-driven prediction and customization. However, it still fell short. Low-code platforms were the third option, allowing for more straightforward interface design but significant backend customization and incompatibility with machine learning components. This evaluation led to the decision of selecting the custom development approach as it was deemed appropriate for the system's long-term objectives, scalability needs, and technical requirements.

2.2.1 Use Case Diagram

The use case diagram shows the functional interactions between external actors and the AI-Powered Environmental Impact Tracker system. It lays out the main features of the system. This includes actions at the user level and the administrator level. It also covers relationships between core processes and optional ones. The system involves two primary actors. The User interacts with the application to track and manage personal environmental impact. The Administrator takes care of maintaining emission datasets. They also manage user accounts and monitor overall system performance. The rectangle labeled "AI-Powered Environmental Impact Tracker" marks the system boundary. It encloses all the functional use cases. Actors stay outside this boundary. They connect with the system through the defined functionalities.

Users get several key functions from the system. The Register/Login use case lets them create a new account or authenticate an existing one. It includes the Reset Password use case as a required sub-process for forgotten credentials. Through the Enter Lifestyle Data use case, users input details about energy consumption, travel, and diet. This case includes a Validate Data process to check accuracy and consistency in the input. The Generate Carbon Footprint use case then calculates the user's carbon footprint based on that information. It can optionally extend to Get Recommendations for actionable suggestions to cut emissions. Users view historical data via the View History use case. They access trends, analytics, and

recommendations on the Dashboard. The Get Recommendations use case can run on its own too. It draws from stored data to offer personalized advice.

Administrators handle specific tasks for maintenance and monitoring. The Manage Emission Dataset use case lets them update emission factors. This keeps carbon footprint calculations accurate. They manage user accounts with the Manage Users use case. That covers adding, modifying, or removing accounts. The Monitor System Performance use case helps them track overall operation. It allows for smooth running and quick issue detection.

The diagram uses <<include>> and <<extend>> relationships to show mandatory and optional flows. The Register/Login use case includes Reset Password as a must-do step. Enter Lifestyle Data includes Validate Data in the same way. These represent required parts of the processes. The Generate Carbon Footprint use case extends to Get Recommendations. This points to an optional path where users get extra advice for lowering environmental impact. The setup clearly divides essential functions from optional ones. It also illustrates how users and administrators engage with the system inside its boundaries.

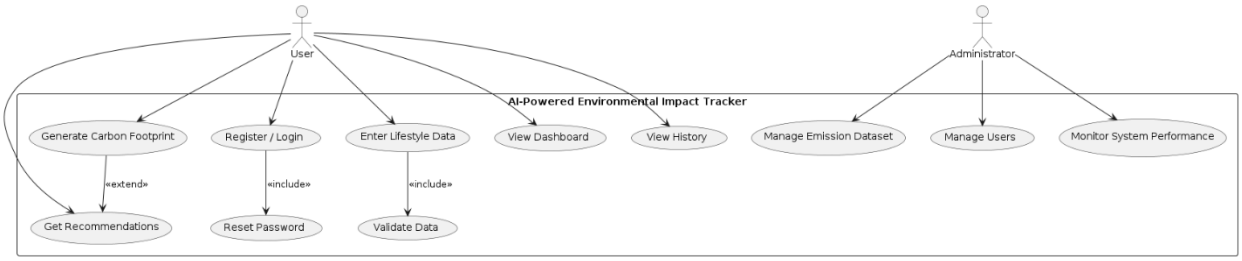

Figure 1:Use Case Diagram

2.2.2. Use Case Narratives.

The AI powered environmental impact tracker system mainly deals with two kinds of people. Users and administrators both have roles to play. Users start by registering or logging in to get access. The system even has a reset password option for when they forget their credentials. That keeps things secure. They input details about their daily lifestyle. The system checks that data to make sure it is accurate. Then it stores everything properly. From there, users can run calculations on their carbon footprint. The AI algorithms handle that part. Sometimes users get tips tailored just for them. Those suggestions help cut down on emissions if they want. They can check their info

on a dashboard too. Looking back at past records shows progress over time. Trends become clear that way.

Administrators take care of the bigger picture. They update the dataset on emissions regularly. They watch over all user accounts closely. System performance gets monitored to keep calculations right. Operations stay smooth without issues. Things like include and extend relationships show how processes connect. Some are required, like reset password tying into login. Others are extra, such as recommendations adding to footprint generation. In the end, the whole setup offers a solid way to track impacts. People can analyze their habits and make changes. It stays reliable thanks to admin oversight.

2.2.3 Class Diagram

The class diagram shows the main parts of the AI-Powered Environmental Impact Tracker. It covers their attributes, methods, and how they connect to each other. Some key classes stand out like User, LifestyleInput, CarbonModel, RecommendationEngine, Dashboard, DatabaseManager, and EmissionFactors. Users handle things like registering, logging in, and sending in various lifestyle records over time. Those records get checked and run through the CarbonModel, which figures out the carbon footprints accurately. Then the RecommendationEngine steps in with tips tailored just for that user. Meanwhile, the Dashboard pulls up trends and works directly with the DatabaseManager to store or grab data as needed. EmissionFactors provide all the essential info for those calculations to come out right. The links between these classes, along with their multiplicities, set up solid connections. That setup keeps the data moving smoothly, holds everything together functionally, and makes the whole system easier to maintain in the long run.

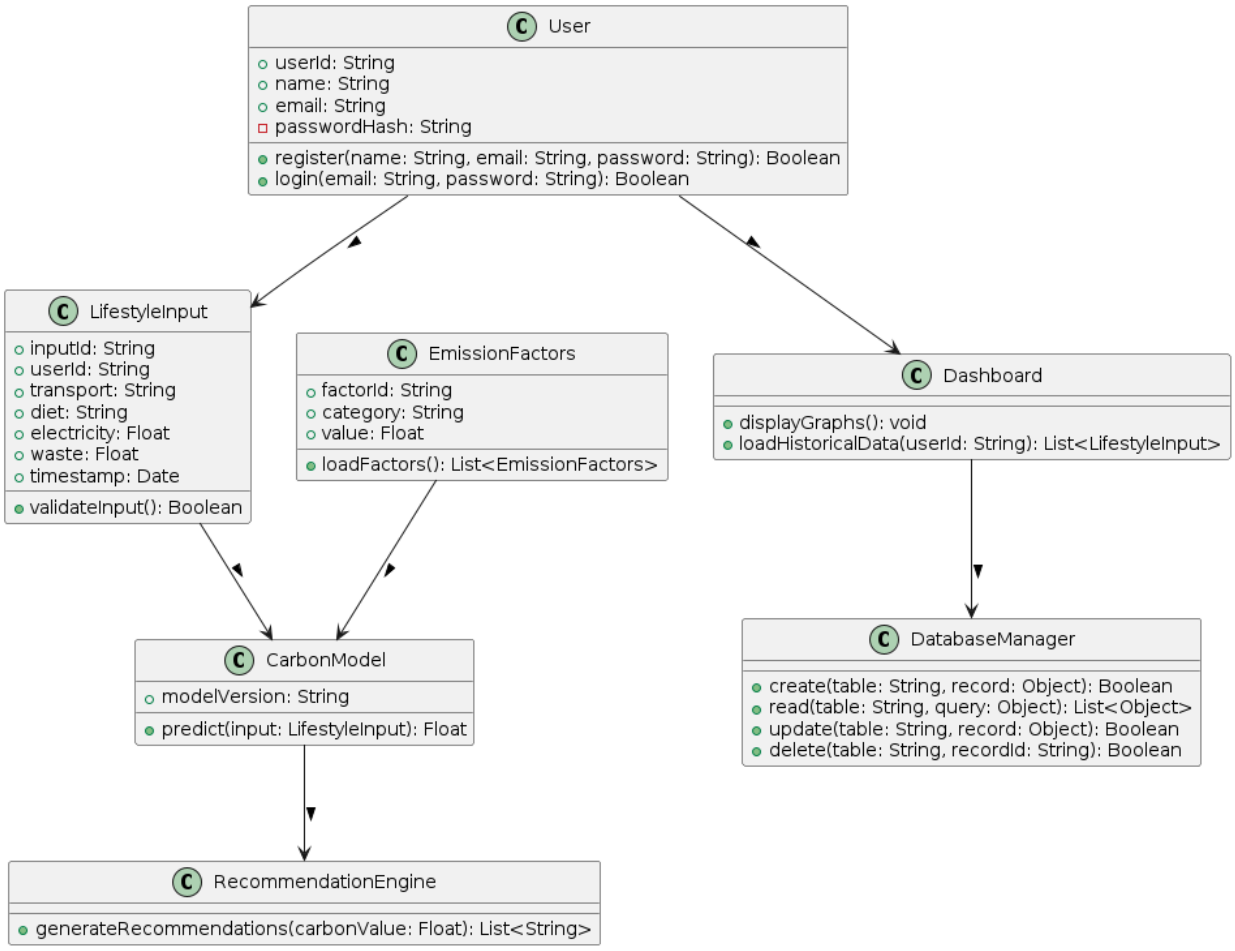

Figure 2:Class Diagram

2.2.4 Sequence Diagrams

The sequence diagram shows how actors and system components interact in two main scenarios: user login and generating carbon predictions, including how errors are managed. In the login process, the user types of their email and password into the Frontend UI. The request then goes to the Backend API, which checks the Database for user details. Here's where it gets interesting: if the password matches, the API confirms this and sends back a success message. The UI then takes you straight to your dashboard. On the flip side, an incorrect password leads to a failure response from the API, resulting in an error message on your screen. This setup guarantees not only successful logins but also secure handling of your data. Experiencing login failures can be frustrating. Here's how the carbon prediction process usually goes: The User enters their lifestyle information through the Frontend interface. This data travels to the Backend API and gets saved

in the Database. After that, the API asks the CarbonModel for a carbon footprint prediction. Now, there are two possible scenarios:

First, let's look at when everything works smoothly. In this case, the ML model provides a carbon value prediction. The API then stores this value in the Database and sends it along with personalized advice back to the UI, so the User can see it.

But things don't always go perfectly. If there's an issue with the ML model—like if it's down or taking too long-the API will log what went wrong and send a friendly message to the UI to let you know there's a temporary glitch.

This sequence diagram does a great job of illustrating how messages get passed around and who is involved at each step. It makes clear how different parts of the system interact under

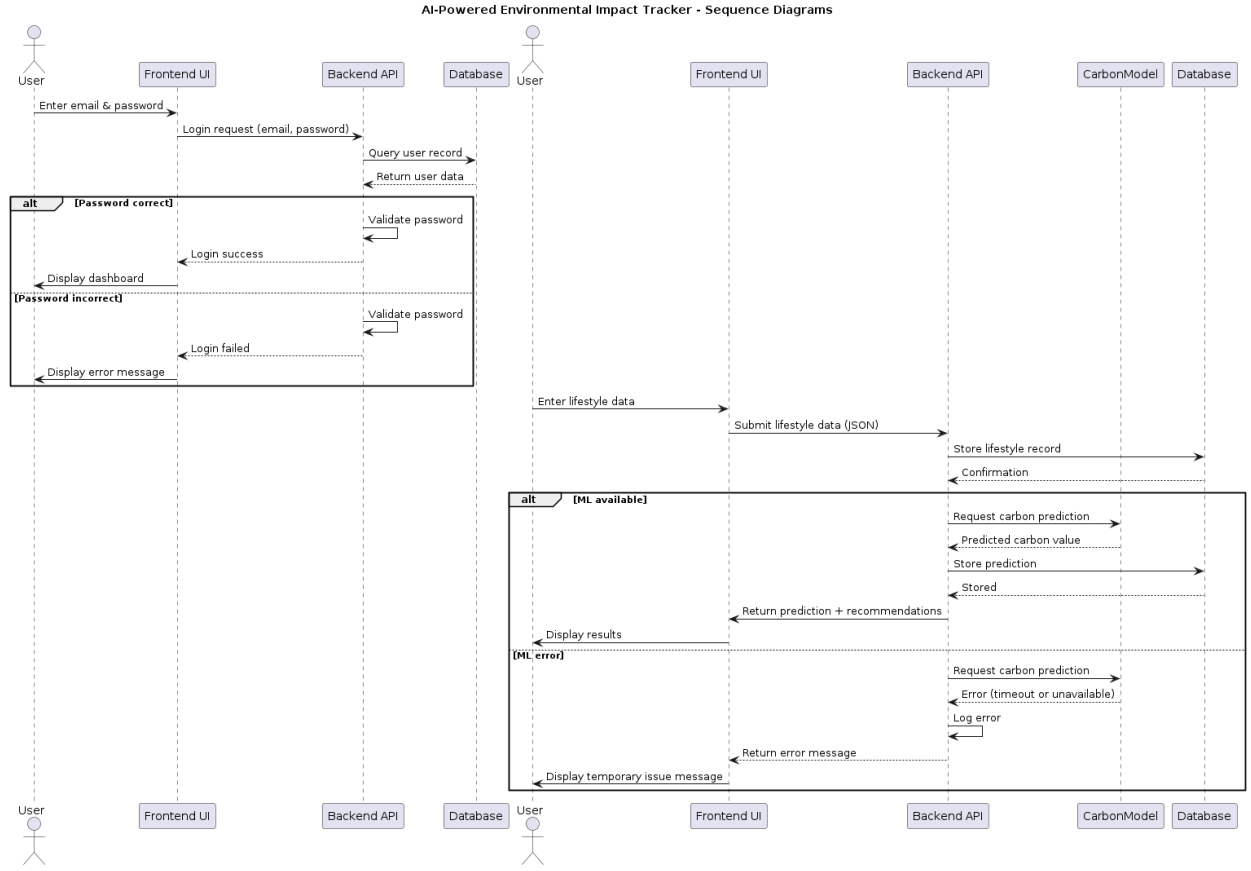

Figure 3:Sequence Diagram

2.2.5 Activity Diagrams

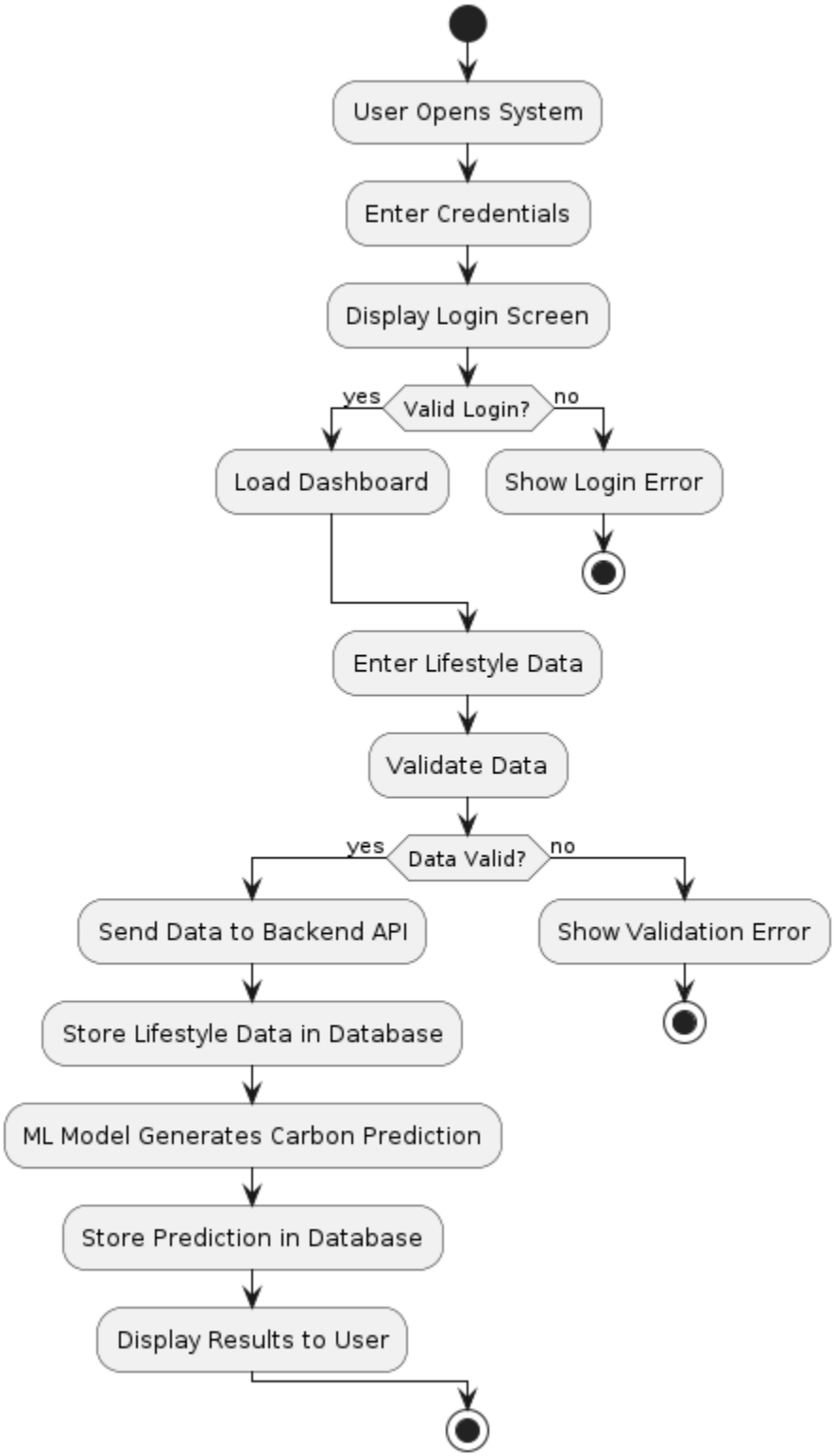

Figure 4:Activity Diagram

2.3 Interface Design

2.3.1 Design Principles

The interface design is built on principles of simplicity, visual consistency, accessibility, and responsiveness. A minimal and structured layout ensures ease of navigation, while uniform colour schemes and typography establish a cohesive visual identity. Accessibility is prioritized by ensuring adequate text contrast, clear font sizes, and support for screen adaptation on different devices. The system is designed to be responsive, allowing users to access it from desktops, tablets, and mobile devices without compromising usability or readability.

2.3.2 Actual Screens of the System

The system comes with a neat and easy to use interface, which is very helpful for the ease of navigation to all the different places in the app. The Login Screen has a clean design, where the input areas are clearly labeled and the error messages are easy to see. After a successful login, the user gets to the Dashboard Screen showing his/her carbon footprint insights through charts, graphs, and categories' breakdowns. The Lifestyle Input Screen lets the user provide information about

his/her transport, energy, and diet consumption, water use, and waste, by using well-organized fields and dropdown menus to avoid mistakes. After data submission, the Prediction Result Screen discloses total carbon footprint plus category-wise emissions through visualizations that are not hard to understand. The Recommendations Screen has suggestions from AI in card style, which is a way to encourage users to be green. The History Screen keeps the record of previous predictions and emission trends, so the user can observe his/her improvement through the time. Color themes that are the same, typography that is easy to read, and layouts that are responsive, guarantee that the system is powerful on all devices and at the same time is user-friendly for people with different levels of technical skills.

2.4. Security Architecture

The security architecture of the system was designed in a layered way to protect user data and provide secure interactions across all parts of the system. In the context of user authentication, the system implements a token-based workflow using JSON Web Tokens, where credentials are hashed before being stored using state-of-the-art algorithms. During login, the system checks user

identity and then generates a time-limited token that is required for accessing protected endpoints. All communication between the front-end, backend APIs, and the database is encrypted with HTTPS, preventing an interceptor from reading or tampering with data. Input validation and sanitization are used throughout the system to avoid injection attacks, while role-based access control ensures only administrators can perform sensitive operations regarding dataset updates or model management. Secure session management, periodic logouts, and inactivity timeouts are also used to reduce unauthorized access risks. Regular backups and monitoring logs are designed to detect suspicious activities.

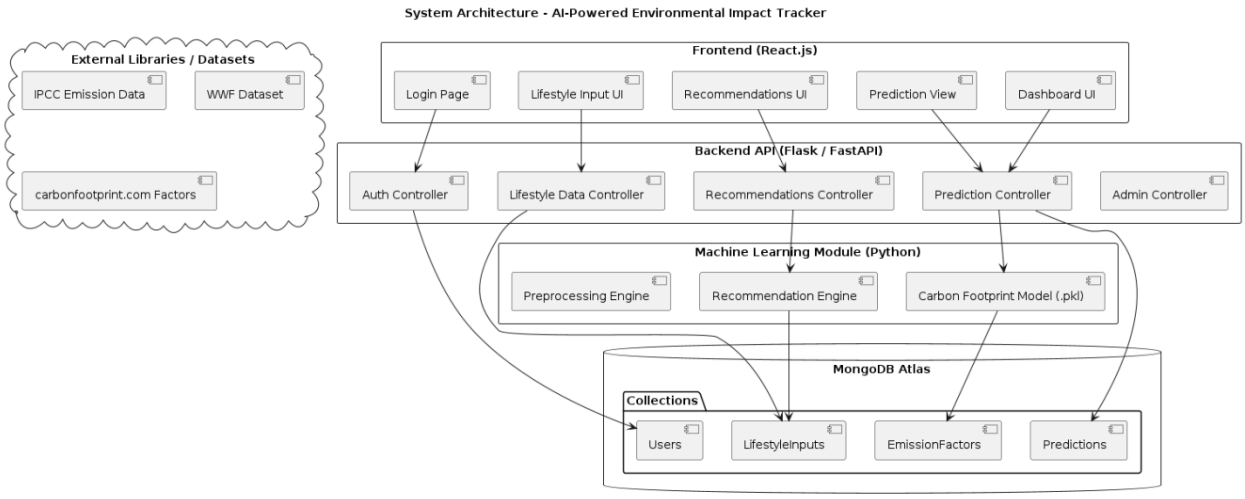

Figure 5:System Architecture

2.5 Deployment Architecture

While the detailed deployment architecture is presented in the implementation chapter, this design considers a cloud-based deployment environment for scalability, reliability, and performance. The system will be set up to run on a cloud infrastructure, such as AWS, Google Cloud, or Azure, using containerized deployment via Docker for portability. The backend API and machine learning model will be deployed in a load-balanced environment to cater to multiple requests without any interruption to the service. The database will be set up on a managed cloud database service with automatic backups, while read replicas are optional for performance optimization. Static front-end assets will be served over CDN to ensure minimum load times for users across the globe. This kind of deployment design ensures fault tolerance, efficient resource management, and the ability to scale with the number of users.

Deployment Architecture - AI-Powered Environmental Impact Tracker

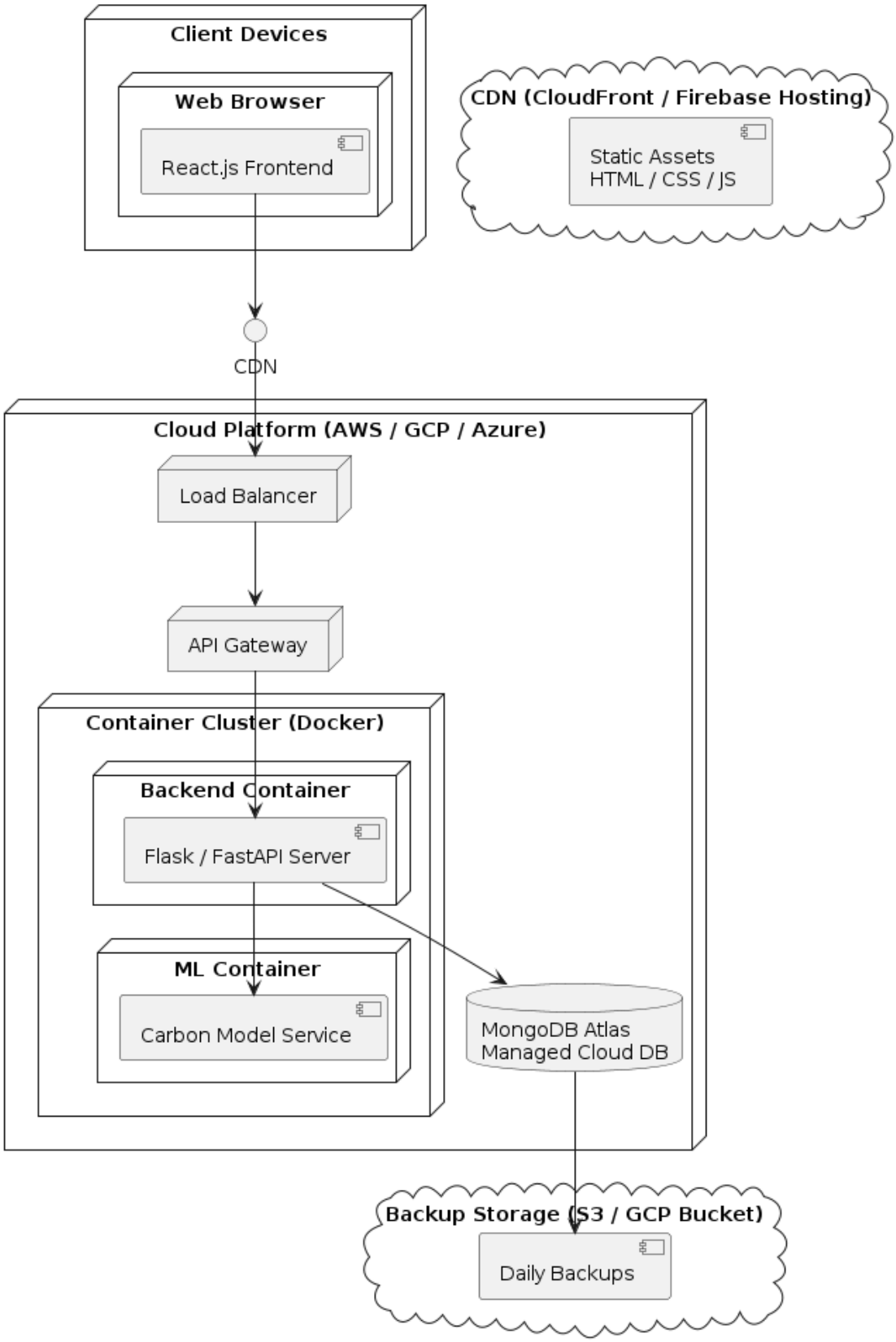

Figure 6:Deployment Architecture

2.6. Database Design

2.6.1 Entity Relationship Diagram

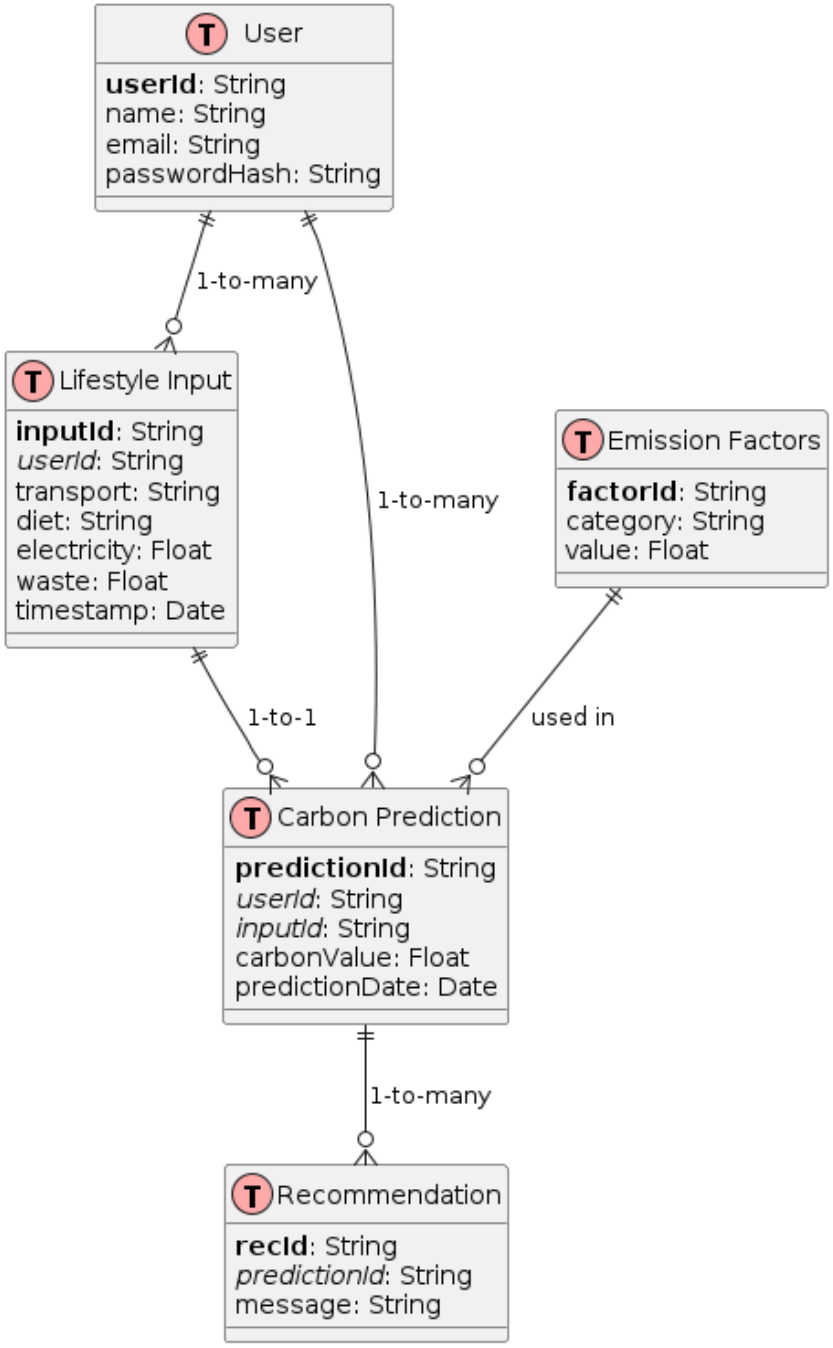

Figure 7:ER Diagram

The ERD outlines how data is structured and connected among the main parts of the AI-Powered Environmental Impact Tracker system. It offers a snapshot of the database design, illustrating entities, their characteristics, primary keys, foreign keys, and how they relate to each other. Think of the User entity as representing people who use the system; it includes details like user Id (PK), name, email, and password Hash. Users can log multiple lifestyle records, establishing a one-to-many link with the LifestyleInput entity. This captures specifics about transportation habits, diet choices, electricity consumption, waste production, and time stamps. Meanwhile, the CarbonPrediction entity keeps forecasts of users' carbon footprints. It connects back to both user Id and input Id using foreign keys. The system connects each prediction to a specific lifestyle record. This involves a many-to-one link with User and a one-to-one link with LifestyleInput. The Recommendation entity gives users personalized advice on cutting down their carbon footprint. A single CarbonPrediction can result in multiple recommendations, creating a one-to-many relationship. EmissionFactors contain coefficients for various categories like energy, transport, and diet. These factors play a role in calculating the carbon footprint and are linked to multiple carbon predictions, showing their importance. Crow’s foot notation clearly represents all these relationships, highlighting primary keys in bold and foreign keys in italics for clarity. This setup ensures accurate storage, retrieval, and calculation of personalized environmental impact data.

2.6.2. Normalization.

The database for the AI-Powered Environmental Impact Tracker follows standard normalization rules (DigitalOcean, 2025), (freeCodeCamp, 2022), (GeeksforGeeks, 2025), (Guru99, 2025). This helps avoid redundancy, ensures data remains consistent, and supports growth. The normalization starts from an Unnormalized Form (UNF) and moves to First Normal Form (1NF). Afterward, it advances to Second Normal Form (2NF) and eventually reaches Third Normal Form (3NF).

1. Unnormalized Form (UNF)

Initially, data was stored on a single table containing all user activities and related emissions. This table had repeating groups and redundant data:

Table 1

userId	name	email	transport	diet	electricity	waste	emission Category	emission Value
U001	Alice	alice@email.com	Car, Bus	Vegan	50	5	Transport	0.25
U001	Alice	alice@email.com	Car, Bus	Vegan	50	5	Diet	0.15
U002	Bob	bob@email.com	Bike	Non-Vegan	30	3	Transport	0.05
U002	Bob	bob@email.com	Bike	Non-Vegan	30	3	Diet	0.25

Issues in UNF:

- Repeating groups (e.g., multiple transport modes per user).
- Redundant data (user info repeated for every lifestyle record).
- Partial dependencies (emission category and value depend on multiple columns).

2. First Normal Form (1NF)

Goal: Eliminate repeating groups; ensure atomicity.

- Each repeating value is placed in a separate row.
- All columns contain atomic values.

LifestyleInput Table:

Table 2

Input Id	userId	transport	diet	electricity	waste	timestamp
L001	U001	Car	Vegan	50	5	2025-11-23

L002	U001	Bus	Vegan	50	5	2025-11-23
L003	U002	Bike	Non-Vegan	30	3	2025-11-23

Functional Dependencies after 1NF:

- Input Id → userId, transport, diet, electricity, waste, timestamp
- userId → name, email

3. Second Normal Form (2NF)

Goal: Remove partial dependencies; every non-key attribute must depend on the whole primary key.

- LifestyleInput is now keyed by input Id (single-column primary key).
- Attributes like name and email depend on userId rather than input Id.
- Separate User table:

User Table (2NF):

Table 3

userId	name	email
U001	Alice	alice@email.com
U002	Bob	bob@email.com

LifestyleInput Table (2NF):

Table 4

Input Id	userId	transport	diet	electricity	waste	timestamp
L001	U001	Car	Vegan	50	5	2025-11-23
L002	U001	Bus	Vegan	50	5	2025-11-23

L003	U002	Bike	Non-Vegan	30	3	2025-11-23

Functional Dependencies after 2NF:

- userId → name, email
- input Id → userId, transport, diet, electricity, waste, timestamp

4. Third Normal Form (3NF)

Goal: Remove transitive dependencies; all non-key attributes must depend only on the primary key.

- CarbonPrediction table separates computed predictions from lifestyle inputs.
- EmissionFactors table separates emission coefficients from user activities**.**

Tables in 3NF:

- **EmissionFactors Table:**

Table 5

Factor Id	category	value
E001	Transport	0.25
E002	Diet	0.15

- **CarbonPrediction Table**

Table 6

Prediction Id	Input Id	userId	carbon Value	prediction Date
C001	L001	U001	25.5	2025-11-23
C002	L003	U002	12.5	2025-11-23

Functional Dependencies after 3NF:

- userId → name, email
- input Id → userId, transport, diet, electricity, waste, timestamp
- prediction Id → input Id, userId, carbon Value, prediction Date
- factor Id → category, value

2.6.3. Relational Schema.

The final relational schema features four main entities: User, LifestyleInput, CarbonPrediction, and EmissionFactors. These entities are each normalized and structurally independent. This setup maintains data integrity, allows for scalability, and provides efficient data access. The complete schema below is presented in standard relational notation. Primary keys are underlined, foreign keys are in italic, and data types are specified.

1. **User Table**
   - Stores the authentication and identity details of each system user.

User(
userId*:* VARCHAR (50),
name: VARCHAR(100),
email: VARCHAR(100) UNIQUE,
passwordHash: VARCHAR (255)
)

2. **LifestyleInput Table**

   - Captures user-entered lifestyle behavior data.

LifestyleInput(
inputId: VARCHAR(50),
userId: VARCHAR(50),
transport: VARCHAR(50),
diet: VARCHAR(50),
electricity: FLOAT,
waste: FLOAT,

timestamp: DATETIME

)

- Foreign Key: *userId* REFERENCES User(userId)

3. CarbonPrediction Table

- Stores the carbon footprint calculation generated for each lifestyle entry.

CarbonPrediction(

predictionId: VARCHAR(50),

userId: VARCHAR(50),

inputId: VARCHAR(50),

carbonValue: FLOAT,

predictionDate: DATE

)

- Foreign Keys:
  - *userId* REFERENCES User(userId)
  - *inputId* REFERENCES LifestyleInput(inputId)

4. EmissionFactors Table

- Contains verified emission coefficient values used by the AI model.

EmissionFactors(

factorId: VARCHAR(50),

category: VARCHAR(50),

value: FLOAT

)

5. Recommendation Table

- Stores generated recommendations linked to each prediction.

Recommendation(

recId: VARCHAR(50),

predictionId: VARCHAR(50),

message: TEXT

)

- **Foreign Key:** *predictionId* REFERENCES CarbonPrediction(predictionId)

Table 7

Table	Primary Key	Foreign Keys
User	userId	—
LifestyleInput	inputId	userId → User(userId)
CarbonPrediction	predictionId	userId, inputId → User & LifestyleInput
EmissionFactors	factorId	—
Recommendation	recId	predictionId → CarbonPrediction

2.7. Hardware Component.

The system operates without the need for specialized hardware. This is a fully web-based application that can be read on any modern browser on devices like laptops, mobile phones, or tablets. It is possible to use standard personal computers to build and test the system as a developer.

References

al., K. B. (2001). Retrieved from Manifesto for Agile Software Development: https://agilemanifesto.org/

Carbonfootprint.com. (2025). Retrieved from Industry-standard carbon calculation platform: https://www.carbonfootprint.com/

DigitalOcean. (2025). Retrieved from Database Normalization: 1NF, 2NF, 3NF & BCNF Examples: https://www.digitalocean.com/community/tutorials/database-normalization

Foundation, P. S. (2025). Retrieved from Python Official Documentation: https://docs.python.org/3/

freeCodeCamp. (2022). Retrieved from Database Normalization – Normal Forms 1nf 2nf 3nf Table Examples: https://www.freecodecamp.org/news/database-normalization-1nf-2nf-3nf-table-examples/

GeeksforGeeks. (2025). Retrieved from Normal Forms in DBMS: https://www.geeksforgeeks.org/dbms/normal-forms-in-dbms/

Guru99. (2025). Retrieved from DBMS Normalization: 1NF, 2NF, 3NF Database Example: https://www.guru99.com/database-normalization.html

Inc., M. (2025). Retrieved from MongoDB Documentation: https://www.mongodb.com/docs/

IPCC. (2023). (I. P. Change, Producer) Retrieved from Emissions Factor Database: https://www.ipcc-nggip.iges.or.jp/EFDB/main.php

Kurzgesagt. (n.d.). What is a Carbon Footprint? Retrieved from https://youtube.com/watch?v=8q7_aV8eLUE

Meta. (2025). Retrieved from React.js Documentation: https://react.dev/

Projects, P. (2025). Retrieved from Flask Documentation: https://flask.palletsprojects.com/

Sutherland, K. S. (2020). Retrieved from The Scrum Guide: https://scrumguides.org/

WWF. (2021). Retrieved from Carbon Footprint Calculator Methodology Document: https://www.wwf.org.uk/

YOUTUBE VIDEO REFERENCES:

- https://youtube.com/watch?v=8q7_aV8eLUE
- NASA Climate Change Channel: https://youtube.com/c/NASAClimateChange
- WWF International: https://youtube.com/user/wwf