*CSC 490 2025 Spring Final Report*

EcoImpact

Project Team: Ivan Ramos Candelero, Marwen Alova, Benjamin Woods

Table of Contents

**1. Project Definition (300-500 words)**

1. Why: Although a vast amount of climate change data is publicly available online, it is often presented in ways that are overly technical, dense, or abstract. As a result, the average person struggles to grasp the real-world significance of this information. This communication gap between climate science and the general public can lead to disengagement, misinformation, or apathy toward one of the most urgent issues of our time. Many people care about the environment but feel overwhelmed by data or confused by conflicting interpretations. We believe that by translating complex data into more intuitive formats, we can foster greater awareness, concern, and ultimately meaningful action.
2. What: Our solution is a user-friendly web application that simplifies and clarifies essential—but frequently misunderstood—climate change data. The goal of the app is to make these insights accessible to the everyday user through compelling visuals and interactivity. Instead of scrolling through charts or tables, users are guided through visual stories that make abstract concepts tangible. The app focuses on three primary dimensions of climate impact: water, air, and ground. Each category contains multiple subsections exploring specific issues such as sea level rise, CO₂ concentration of companies compared to individuals, the impact of littering, or ocean plastic accumulation. These sections are designed not just to inform but to evoke a deeper emotional and cognitive response from users.
3. How: To achieve this, the web app will rely heavily on visualizations—animated infographics, dynamic charts, interactive maps, and illustrated timelines—that convey climate data in an instantly understandable and emotionally resonant way. The interface will be clean, intuitive, and segmented so users can easily navigate between different environmental categories. Each section will aim to answer not just what the data says, but why it matters and what it looks like when translated into real-world consequences. The design philosophy is simple: make climate science feel immediate, human, and actionable to anyone, regardless of their background.

**2. Project Requirements Analysis**

1. Functional Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Functional Requirement** | **Category** | **Notes** | **Priority** |
| 1 | The system ingests and stores corporate CO₂ emissions data and exposes a /companies endpoint returning the  top-5 investor-owned companies by annual emissions. | Data Integration | Implemented in  /companies/ (returns yearly, monthly, weekly, daily, hourly). | High |
| 2 | The system offers three “super-category” entry points—Air, Water, Ground—on the home page, each linking to  its own section. | Navigation/UI | Implemented in src/App.js  (Home → /air, /water, /ground). | High |
| 3 | (Not yet implemented) Store and serve global temperature history via an API. | Data Integration | No temperature endpoints  or UI exist today. | Low |
| 4 | Allow users to enter their personal CO₂ drivers (weekly driving miles, monthly electricity, annual flights),  calculate lifetime emissions, then compare vs. a company. | Personalized Comparison | Implemented in  CarbonComparison.js + /companies/ → circles animation. | Medium |
| 5 | Provide a world-map visualization of country CO₂ emissions; support selecting/deselecting countries to see  how total changes and display a 5-year history graph. | Interactive Visualization | Implemented in MapPage.js + /air\_super/…/past\_five\_years. Users can click countries, toggle “map/country” view, and reset selections. | High |
| 6 | Ice-sheet slider: when the user drags a slider (0–10 ticks), show stacked ice-sheet images, global sea-level  rise, and descriptive impacts. | Interactive Visualization | Implemented in  IceSheets.js → slider 0–10, maps to cm/in and narrative cards. | High |
| 7 | Ocean plastic projections: animate cumulative plastic coverage over time; let users step forward (“Litter  some plastic”) and reset; show country-level coastal risk. | Interactive Visualization | Implemented in PlasticOcean.js + endpoints /ocean-projections, /countries, /country-impact/{code}. | High |
| 8 | Ground-degradation: show how long different littered items take to break down; include a timeline marker and (for recyclable items) a looping recycle animation. | Interactive Visualization | Implemented in  StickAround.js → biodegradable, plastic, metal cup with timeline + video. | Medium |
| 9 | Farm emiss­ions showdown: visualize and compare CO₂ emissions from different agricultural activities. | Interactive Visualization | Implemented in FarmEmissions.js (charts or widgets comparing farm sectors). | Medium |
| 10 | (Not yet implemented) Tree-loss/gain map: visualize forest-cover change vs. CO₂ impact on a world map. | Planned Visualization | No “forest” or “ground  map” feature exists. | Low |

1. Usability Requirements
   1. User interface
      1. The homepage features three clear “cards” (Air, Water, Ground), each color-coded and labeled with icons.
      2. A consistent TopBar and Footer are present on every page to provide quick context and easy navigation back to the homepage.
      3. Section landing pages for Air, Water, and Ground each display two to three “info-cards” that link to deeper visualizations.
      4. Sub-pages include back buttons to return to their parent categories.
      5. Form controls in the Carbon Comparison tool and the country selector in the Plastic Ocean section use native HTML elements like <input> and <select> to ensure familiarity.
      6. All visualizations include alt-text for accessibility, and buttons and sliders are labeled for screen readers.
      7. The layout is responsive, using flex-wrap for card containers and designed to degrade gracefully on narrower viewports; mobile support is functional but not fully polished.
   2. Performance
      1. API response times for simple lookups (such as /air\_super, /companies, and /ocean-projections) are under 200 milliseconds on a local Postgres database.
      2. Map data, consisting of approximately 200 country records, loads in under 500 milliseconds. On the frontend, the initial bundle load (React and D3) takes approximately 2–3 seconds on a typical broadband connection, while subsequent route changes and visual updates occur in under 100 milliseconds.
      3. Interactive animations, such as sliders and circle scaling, perform at 50 milliseconds or better per frame.
      4. The database uses primary keys and indexes on fields such as year, country\_code, and number\_code to ensure fast lookups and efficient execution of aggregated queries like top-5 company listings.
      5. For scalability, the backend is designed to support containerization and the addition of sharding or read-only database replicas as traffic increases.
2. System Requirements
   1. Hardware
      1. For development or small-scale hosting, a machine with 2 CPU cores, 4 GB RAM, and 20 GB of disk space is sufficient.
      2. For production or medium-scale deployment, it is recommended to use a machine with 4 or more CPU cores, 8 or more GB of RAM, and SSD storage.
   2. Software
      1. The backend requires Python 3.8 or newer and uses FastAPI, SQLAlchemy, and Psycopg2.
      2. The frontend requires Node.js version 14 or newer and is built with React 17 or newer, React-Router, D3.js, and Tailwind or CSS modules.
      3. The system can run on Linux (Ubuntu 20.04 or later), macOS, or Windows with WSL. Build tools include npm or yarn for the frontend, and uvicorn or gunicorn for serving the API.
   3. Database
      1. The system uses PostgreSQL version 10 or newer, with testing confirmed on versions 12 and 13.
      2. Database schemas are managed through SQLAlchemy models and initialized using a SQL dump file located in the datasets/Setup directory.
      3. Connection strings and credentials are injected using environment variables.
3. Security Requirements
   1. CORS is currently configured to allow all origins during development, but it must be restricted to known frontends in production.
   2. Input validation and SQL injection protection are handled through parameterized queries using SQLAlchemy and Pydantic, eliminating injection risks.
   3. In development, HTTP is used; however, HTTPS is required in production, with TLS termination handled at the proxy or load balancer level.
   4. Database credentials must not be checked into source control and should be managed using .env files or a secret manager.
   5. Authentication and authorization are not currently implemented, and all endpoints are open.
   6. In the future, JWT or OAuth2 protections should be added around any user-specific or write-capable routes, such as posting new carbon records.
   7. To maintain security, dependencies in both Python and Node should be kept up to date, and vulnerability scans (e.g., using npm audit or pip-audit) should be run periodically.

**3. Project Specification**

1. Focus / Domain / Area
   * 1. EcoImpact is an educational web application devoted to helping users explore and understand humanity’s impact on the environment. Its primary focus spans three interconnected domains—air, water, and ground—each illuminated through real-world data on carbon emissions, ocean plastics, melting ice sheets, and material degradation. By bringing together corporate and personal carbon‐footprint metrics, sea-level projections, coastal waste risks, and degradation timelines for common items, the project aims to foster deeper engagement with climate science and inspire more sustainable choices.
2. Libraries / Frameworks / Development Environment
   * 1. The backend of EcoImpact is built in Python 3.8+ using FastAPI for RESTful endpoints, with SQLAlchemy managing data models and PostgreSQL as the persistent store. In development, Uvicorn serves the API locally, and database schemas are versioned via SQL dumps and SQLAlchemy migrations in the datasets/ folder. The frontend leverages React 17+ (bootstrapped with Create React App), React Router for client-side navigation, D3.js for custom charts and maps, and standard CSS modules (with some Tailwind utilities) for styling. Node.js 14+ and npm scripts orchestrate local builds, tests, and the development server, while Visual Studio Code or any modern IDE may be used for coding.
3. Platform
   * 1. EcoImpact is delivered as a responsive web application that runs in any modern browser on desktop or mobile devices. It has been tested primarily on Chromium-based and WebKit-based browsers (Chrome, Edge, Safari) and requires no native installation, making deployment as a containerized service straightforward.
4. Genre
   * 1. Rather than a game, EcoImpact is an interactive data‐visualization application with educational intent. It combines narrative elements with hands‐on sliders, maps, charts, and animated comparisons to turn abstract environmental metrics into visceral experiences.
5. Dataset / Data Resources
   * 1. The project ingests a variety of open environmental datasets. Historical country‐level CO₂ emissions are drawn from public repositories (e.g., Our World in Data), corporate emissions are aggregated from investor‐owned companies, ocean‐plastic projections come from peer‐reviewed marine‐pollution studies, and coastal‐waste statistics derive from national waste reports. Ground‐degradation timelines for everyday materials are curated from academic and industry sources. All raw data lives in the datasets/ directory as CSVs or SQL dumps, is loaded into PostgreSQL via migration scripts, and then exposed through FastAPI endpoints.

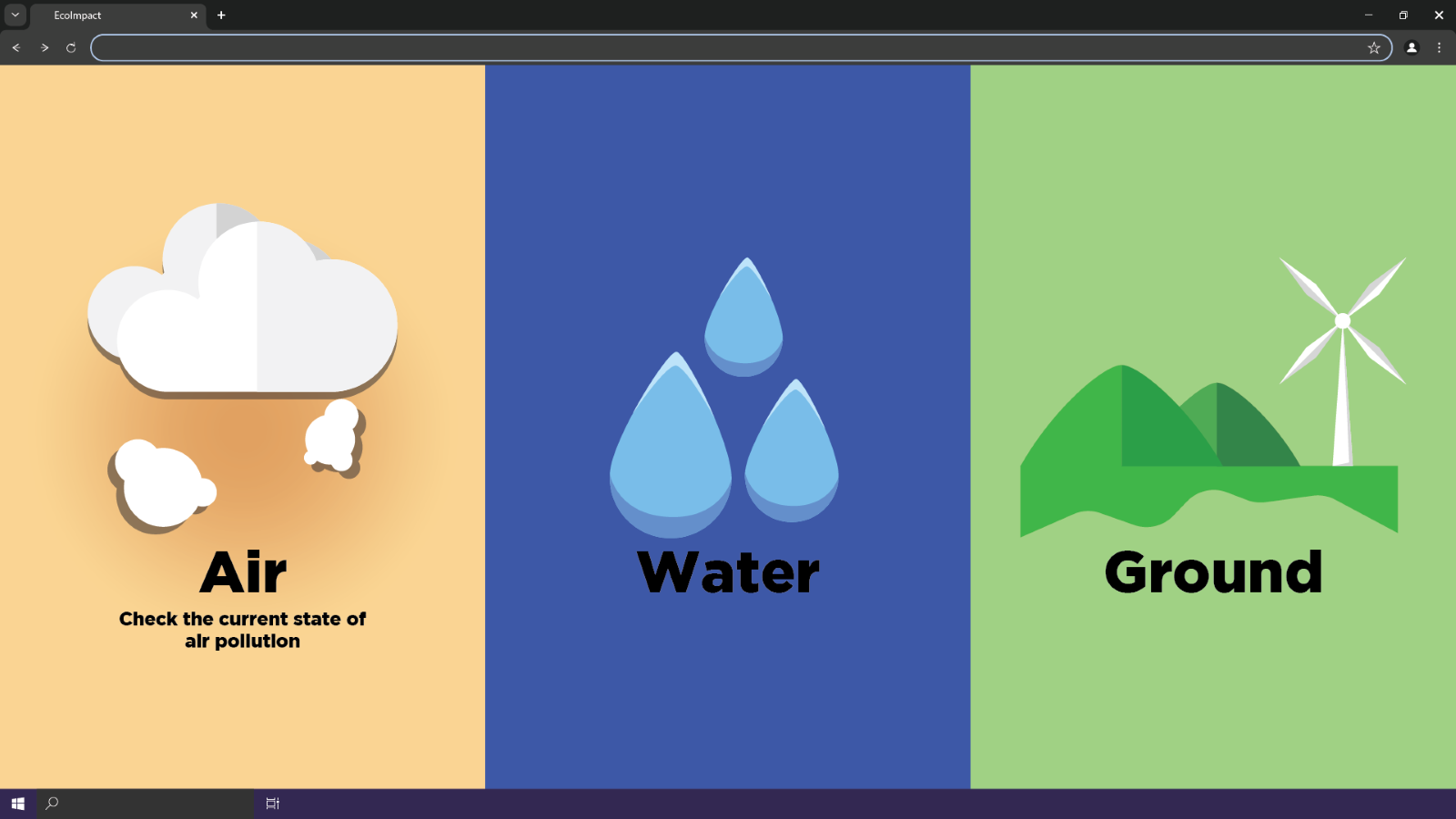
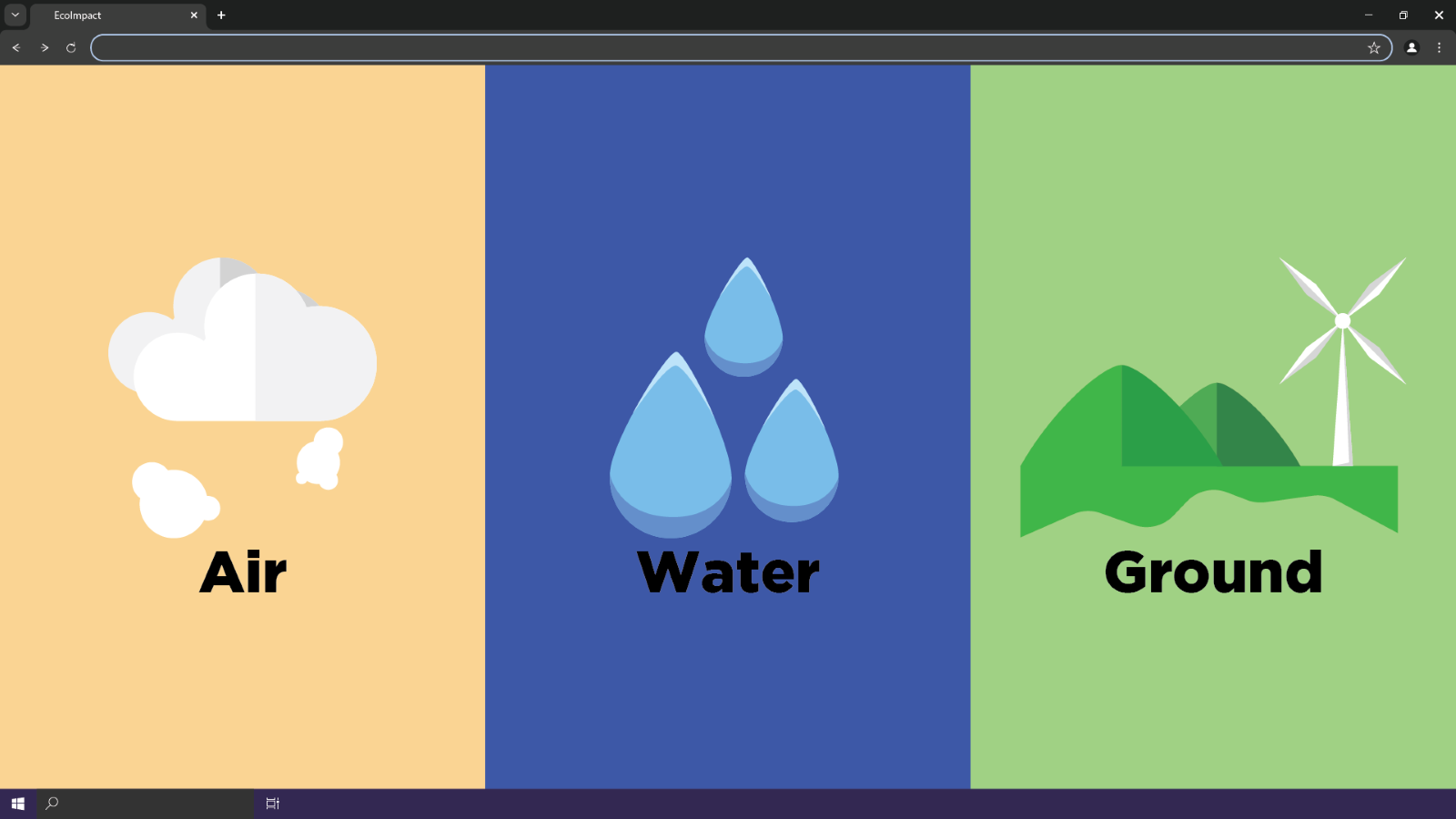
**4. System – Design Perspective**

1. Identify subsystems – design point of view
   * 1. From a design standpoint, EcoImpact is organized into two primary subsystems that together realize a classic three-tier web architecture. The first subsystem is the Backend API and Data Management layer. Its responsibility is to ingest raw CSV and SQL‐dump datasets into a normalized PostgreSQL schema, to run any necessary transformations, and then to expose that information in a consistent, secure, and performant way through RESTful endpoints built on FastAPI. The second subsystem is the Frontend Visualization and Interaction layer, which is a React single‐page application augmented with D3.js for custom charts and SVG maps. It fetches data from the API, orchestrates client‐side routing, and drives all of the user’s interactive experiences (sliders, animated circle comparisons, map selection, and stepwise projections).
2. Overall System Architecture
   * 1. EcoImpact follows a three‐tier model. On the bottom tier sits PostgreSQL, holding tables for country‐level emissions (“air\_super”), water and plastic‐waste statistics, ocean‐plastic projections, corporate emissions, and material‐degradation timelines. In the middle tier, a FastAPI service running under Uvicorn acts as the gateway: it opens database sessions, executes parameterized SQL via SQLAlchemy, and marshals data into JSON for the client. The top tier is a React application served as static assets; it uses React Router to move the user through the Air, Water, and Ground sections and relies on D3.js for all heavy‐lifting visualizations.
3. Use-Case model
   * 1. A typical user scenario begins when someone navigates to the root URL. The homepage displays three cards (“Air,” “Water,” “Ground”). When they click Air, the app fetches country‐list data and corporate emission figures. They may choose “Company V. You,” enter driving, electricity, and flight data, and see a lifetime CO₂ comparison versus a selected company. Alternatively, they can open the world‐map view, click a country to toggle it on or off, observe how the global total shifts in the progress bar, and drill down to see the most recent five years of CO₂ history plotted on a line chart. Similar flows exist under Water (sea-level slider and plastic ocean steps) and Ground (material‐degradation timelines).
4. Relational Schema
   * 1. Under the hood, the database schema is straightforward and denormalized for read performance. The air\_super table keys on country\_code and year and tracks total, coal, oil, gas, cement, flaring, other, per\_capita, and a numeric country code. Water and waste data live in a plastic\_waste\_data table keyed by country\_code, with columns for total waste, recycling rate, per\_capita\_waste\_kg, and coastal\_waste\_risk. Ocean projections are in a plastic\_projections table with year, coverage (fraction of ocean), and narrative impact. The companies table captures parent\_entity, parent\_type, and total\_emissions; corporate emissions are aggregated on query. Finally, object degradation timelines live in an object\_degradation table keyed by object\_id, with object\_name, object\_type, and time\_period in years. All lookups are accelerated by primary‐key indexes on country\_code, year, and object\_id.
5. System Design
   1. Subsystem1: Backend API and Data Management
      * 1. This subsystem is built in Python 3.8+ with FastAPI and SQLAlchemy on top of PostgreSQL. Database connections are pooled by SQLAlchemy’s sessionmaker, and all SQL is parameterized to prevent injection. Models are defined declaratively, and tables are created at startup if they do not exist. Each route corresponds to a specific data slice—air\_super CRUD, top‐5 companies, ocean projections, country impact, and so on. CORS is enabled during development, with the expectation of locking origins in production. The API scales horizontally by adding more Uvicorn workers behind a load balancer.
   2. Subsystem2: Frontend Visualization and Interaction
      * 1. This subsystem is a React 17+ application scaffolded with Create React App. Routing is handled by React Router, and each major page (MapPage, CarbonComparison, IceSheets, PlasticOcean, StickAround, FarmEmissions) is a self-contained component that fetches from the API, renders controls (forms, sliders, buttons), and drives D3 or CSS animations to bring data to life. State is managed with React hooks, and side effects (data fetching, map redraws) are isolated in useEffect. All assets (SVG icons, PNG sequences, videos) live under public/visuals or src/assets, making it straightforward to swap in new imagery or localized text. The design emphasizes accessibility with alt text, labelled controls, and responsive layouts that adapt from desktop to mobile screens.

**5. System – Analysis Perspective**

1. Identify subsystems – analysis point of view
   * 1. The Data Ingestion Layer reads raw CSV and SQL-dump files and populates PostgreSQL tables. The API Layer exposes that data via REST endpoints built on FastAPI and SQLAlchemy. The Client Visualization Layer is a React/D3 application that fetches JSON from those endpoints and renders charts, maps, sliders, and animations.
2. Use-Case Descriptions
   1. Use Case 1 – Select a Super-Category of Pollution

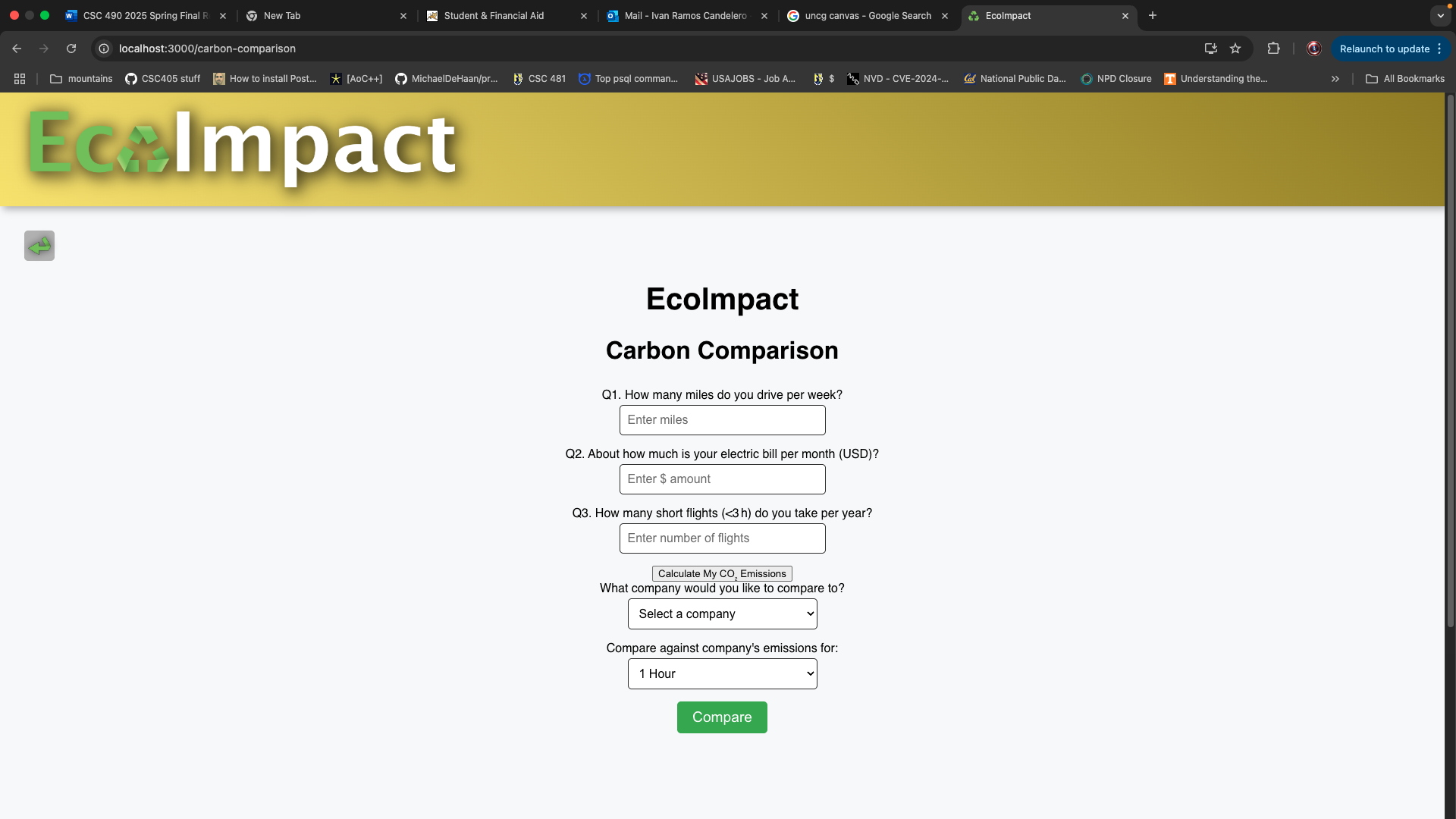
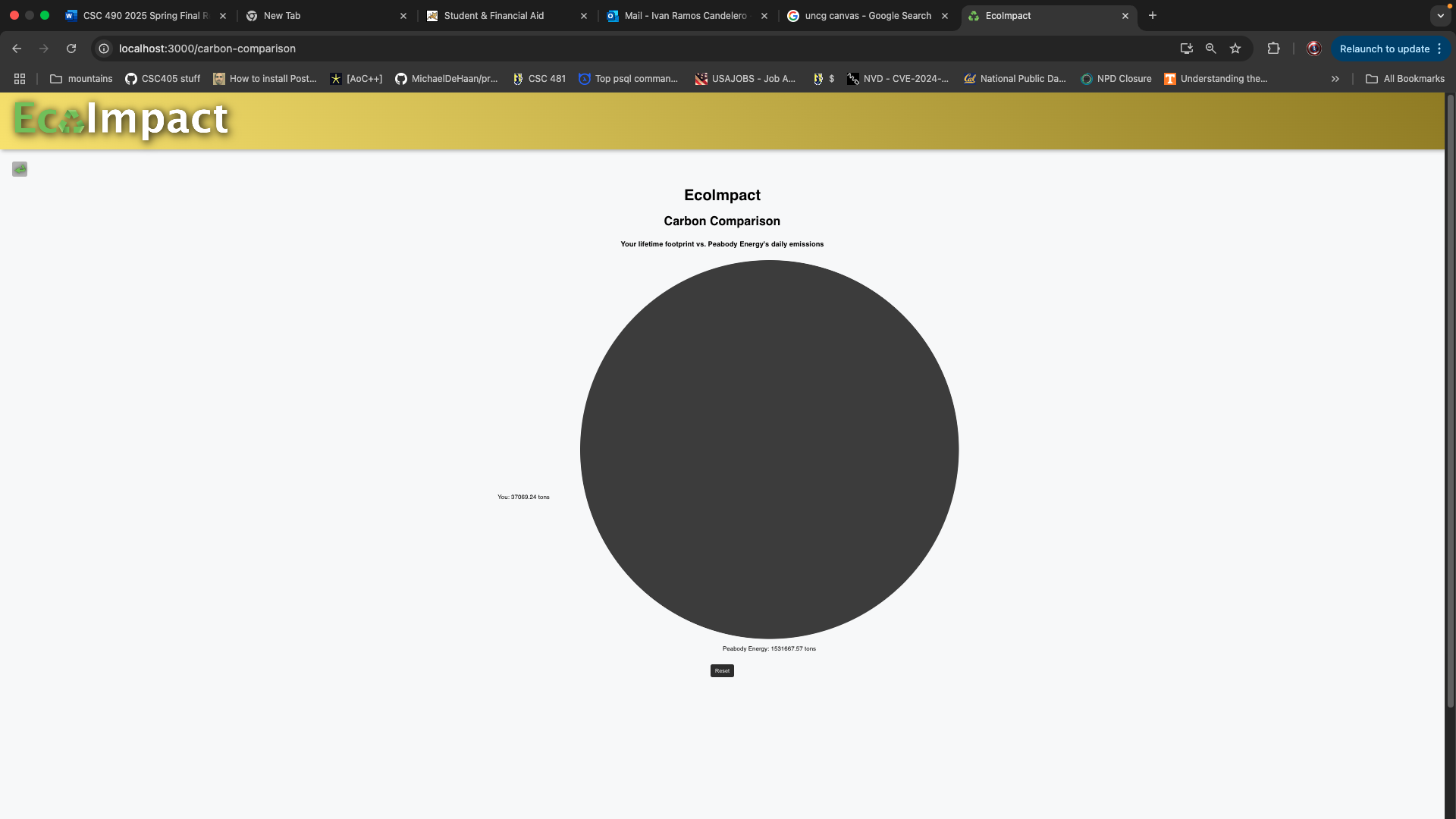
|  |  |
| --- | --- |
| Use Case | Select a super category of pollution |
| Summary | A visitor to EcoImpact wants to begin exploring data by choosing one of the three major environmental domains: Air, Water, or Ground. |
| Actors | The end user and the system. |
| Preconditions | User has opened a browser and is currently on the EcoImpact home screen. |
| Basic Sequence | 1. User clicks on the tab “Air” or “Water” or “Ground”. 2. User is taken to the respective “Air” or “Water” or “Ground” webpage. |
| Exceptions | None. |
| Post Conditions | The requested section page (Air, Water, or Ground) is rendered. |

* + 1. Mock Layout
    2. 
    3. Data Objects

|  |  |
| --- | --- |
| Data | Description |
| CategoryID | an internal identifier for each super-category (e.g. 1=Air, 2=Water, 3=Ground) |
| CategoryName | the human-readable name (“Air,” “Water,” “Ground”) |
| Url | the route path ("/air", "/water", "/ground") |

* Use Case 2 – Enter Information to Compare Carbon Footprint
  + 1. Use Case Description

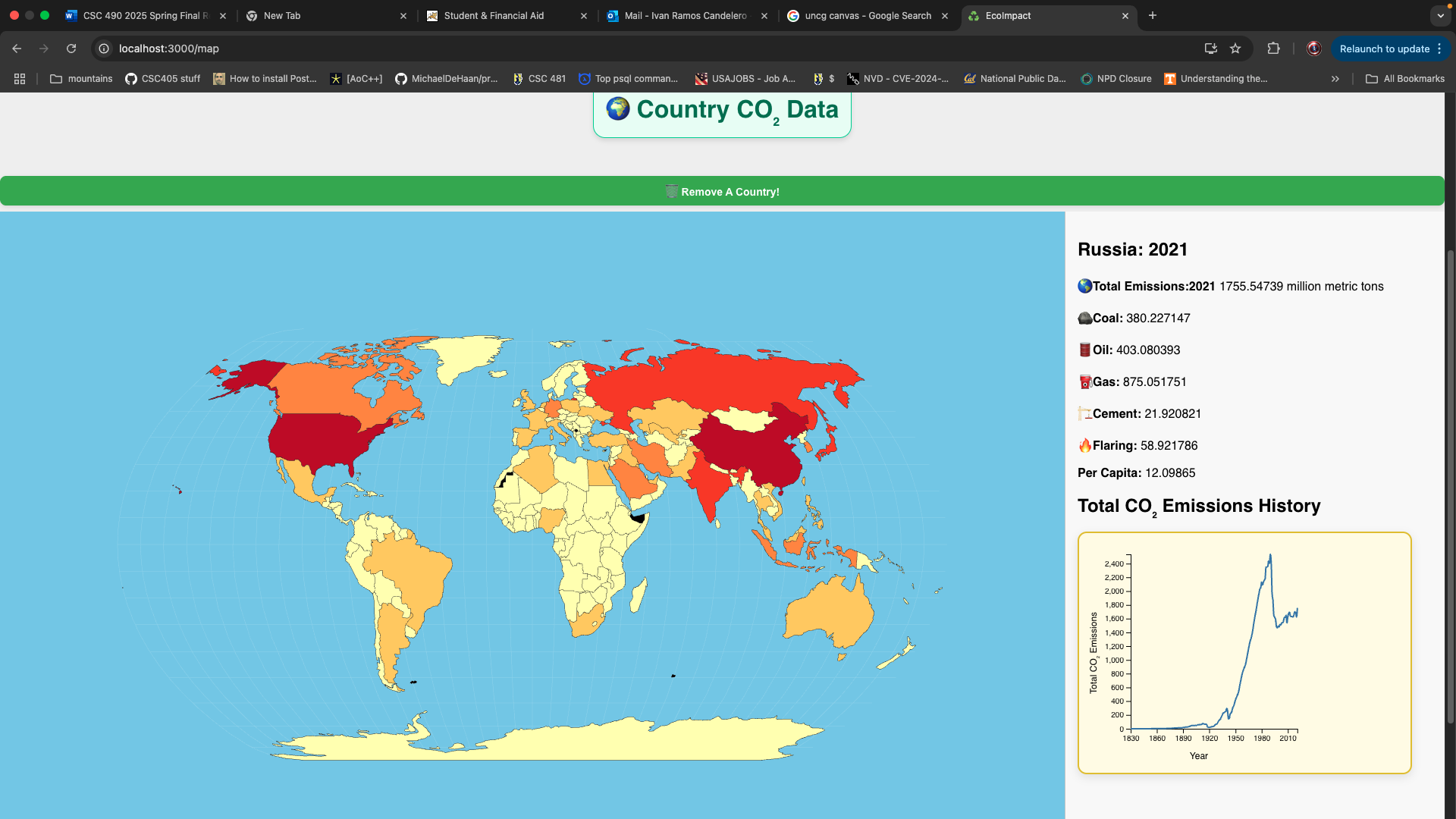
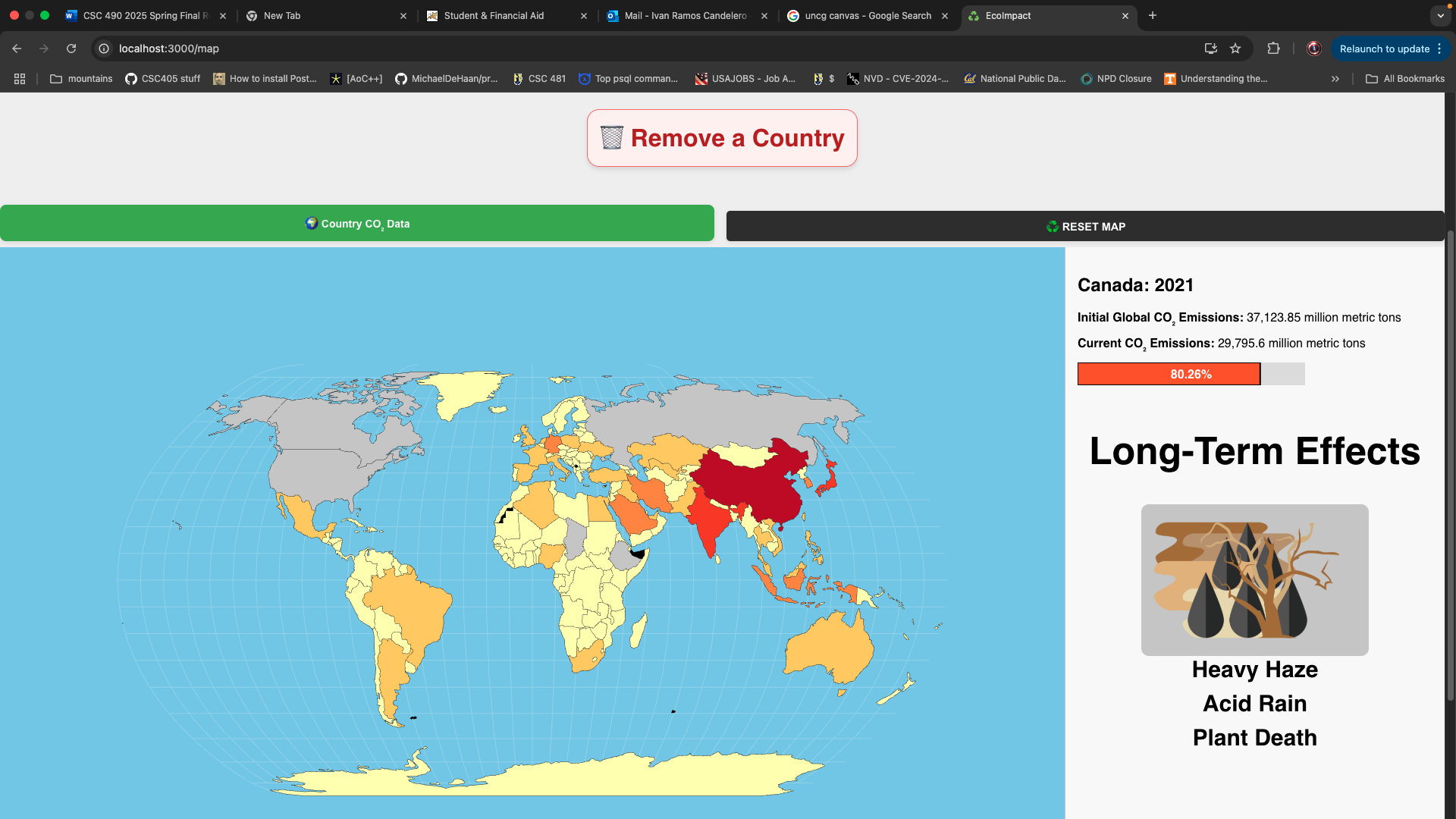
|  |  |
| --- | --- |
| Use Case | Enter information to compare carbon footprint |
| Summary | A user in the Air section wants to calculate their lifetime CO₂ emissions from driving, electricity, and flights, then compare that total to a large company’s emissions. |
| Actors | The end user and the system |
| Preconditions | The user has navigated to “Air” and clicked through to the “Carbon Comparison” page. The system has already fetched the top-5 companies’ emissions data. |
| Basic Sequence | 1. The user enters weekly driving miles, monthly electricity bill, and number of short flights per year. 2. The user clicks “Calculate My CO₂ Emissions.” The system computes lifetime emissions and displays the result. 3. The user selects a company from a dropdown list. 4. The user chooses a comparison basis (hourly or daily). 5. The user clicks “Compare,” and the system animates two circles: one sized to the user’s lifetime total, the other to the company’s rate. |
| Exceptions | If any input is missing or invalid, the system shows an error message and halts the calculation. |
| Post Conditions | The comparison animation completes and both emissions figures remain visible. |

* + 1. Mock Layout
    2. 
    3. Data Objects

|  |  |
| --- | --- |
| Data | Description |
| weeklyMiles | number input by the user (miles/week) |
| monthlyBill | number input by the user (USD/month) |
| flightsPerYear | number input by the user (short flights/year) |
| userFootprint | computed lifetime CO₂ (tons) |
| companies | array of { name, daily\_emissions, hourly\_emissions } fetched from the backend |
| comparisonType | “hourly” or “daily” |

1. Use Case 3 – See the Impact of Different Nations’ CO₂ Visually
   * 1. Use Case Description

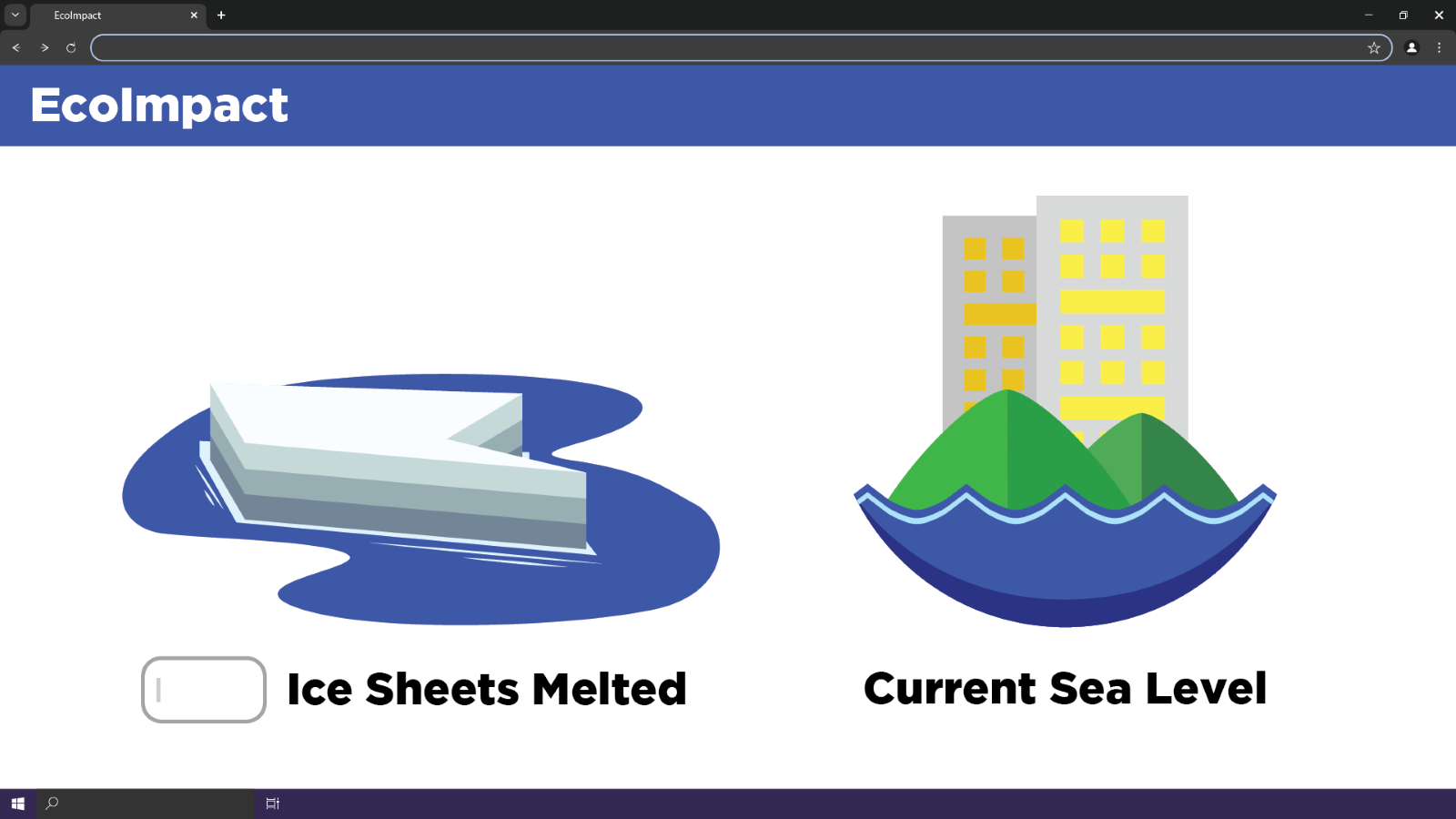
|  |  |
| --- | --- |
| Use Case | See the Impact of Different Nations’ CO₂ Visually |
| Summary | A user wants to explore how selecting or deselecting countries alters the global CO₂ total and drill down into a country’s recent emissions history. |
| Actors | The end user and the system. |
| Preconditions | The user has navigated to the “Air” section and clicked “Map the Emissions.” The world-map topology and country emissions data have loaded. |
| Basic Sequence | 1. In “map” view, the user clicks on a country. If it was included, it becomes excluded (grayed out) and the global total in the progress bar decreases; if excluded, it is re-included and the total increases. 2. The user toggles into “country” view mode and clicks a specific country. The system highlights it, fetches that country’s past five years of emissions, and renders a line chart. 3. The user may click “Reset” to clear selections and restore the original global total. |
| Exceptions | None. |
| Post Conditions | The map and progress bar accurately reflect the new total, and the line graph shows the country’s historical data when in country view. |

* + 1. Mock Layout
    2. 
    3. Data Objects

|  |  |
| --- | --- |
| Data | Description |
| data | Map of the major countries along with their C02 outputs |
| EnvironmentCondition | Stores the different visuals of the potential environment as a result of the total C02 levels |
| C02Levels | Has the total C02 levels based on the countries selected. |

1. User: Visualize the effect of ice sheets melting
   * 1. Use Case Description

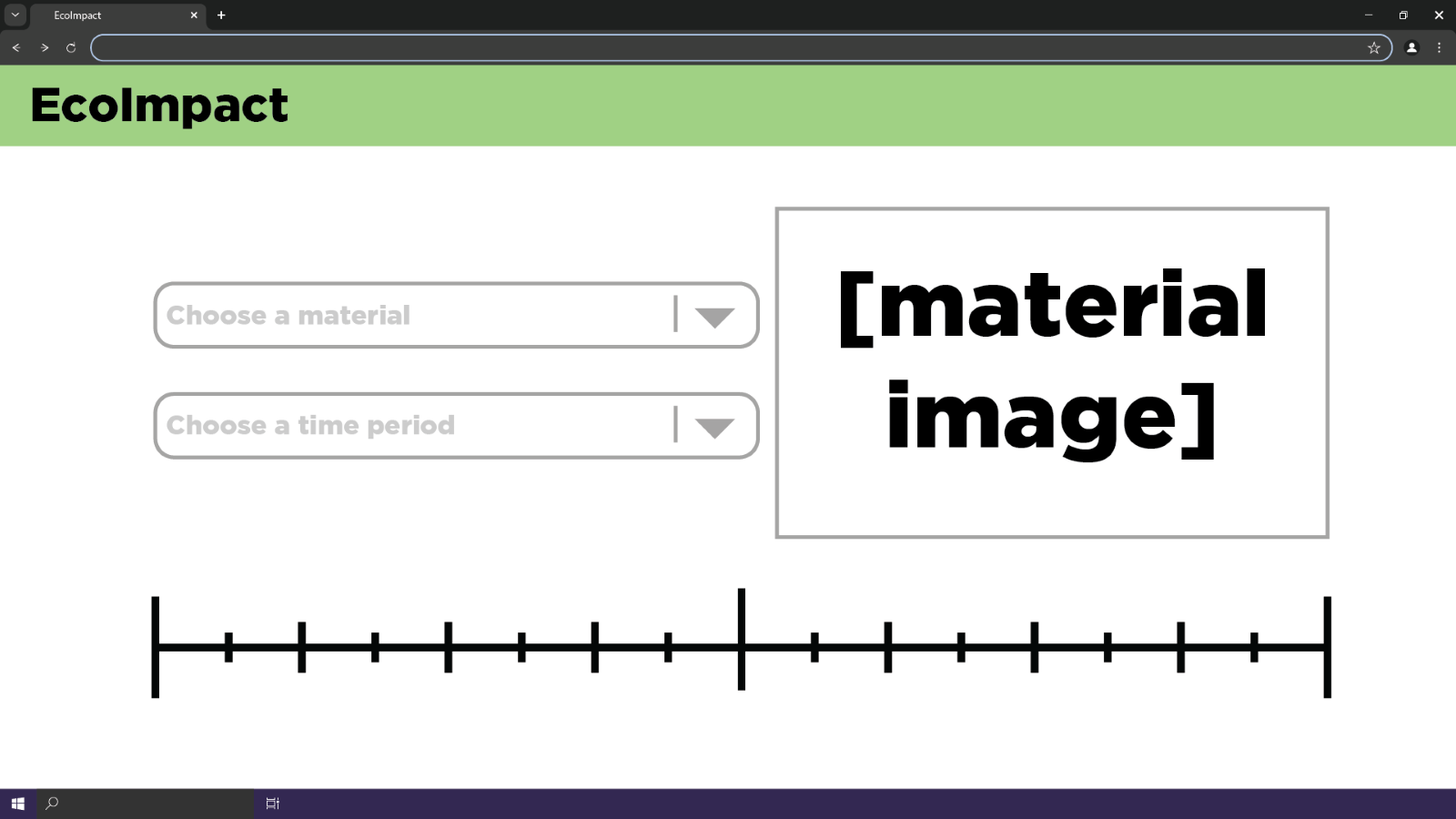
|  |  |
| --- | --- |
| Use Case | array of country records { number\_code, country, total } from /air\_super |
| Summary | A user in the Water section wants to see how melting Greenland’s ice sheets raises sea levels and learn about the impacts at each step. |
| Actors | The end user and the system. |
| Preconditions | The user has navigated to “Water” and selected “Ice? Sheet!” The slider and all ice-sheet and sea-level images are available. |
| Basic Sequence | 1. The user moves the slider between 0 and 10 ticks. 2. For each tick, the system updates the ice-sheet image, calculates sea-level rise in cm/in, computes cumulative gigatonnes melted, and displays a descriptive impact card. |
| Exceptions | None. |
| Post Conditions | The ice-sheet, sea-level visualization, and narrative impact text all correspond to the slider’s value. |

* + 1. Mock Layout
    2. 
    3. Data Objects

|  |  |
| --- | --- |
| Data | Description |
| meltedSheets | integer 0–10 from the slider |
| cmRise | meltedSheets × 10 |
| inchRise | meltedSheets × 36,000 (Gt of ice) |
| greenlandPct | percentage of the Greenland ice sheet melted |

1. Use Case 5 – See How Long Daily Items Last Before Degrading
   * 1. Use Case Description

|  |  |
| --- | --- |
| Use Case | See How Long Daily Items Last Before Degrading |
| Summary | A user in the Ground section wants to find out how long everyday items (metal, plastic, biodegradable cups) take to break down in the environment. |
| Actors | The end user and the system. |
| Preconditions | The user has navigated to “Ground” and selected “Stick Around?” The degradation metadata and visuals are loaded. |
| Basic Sequence | 1. The user clicks one of the three material buttons. 2. The system displays the matching item image, explanatory text about its decomposition timeline (or recycling), and animates a timeline marker to the final year. 3. If the metal cup is “recycled,” a looping recycle animation appears. |
| Exceptions | If the user tries to select more than one material at once, the system ignores additional clicks until the first selection is cleared. |
| Post Conditions | The degradation visualization, timeline marker, and narrative text accurately reflect the chosen material’s lifecycle. |

* + 1. Mock Layout
    2. 
    3. Data Objects

|  |  |
| --- | --- |
| Data | Description |
| selectedMaterial | string (“metal-cup”, “plastic-cup”, or “biodegradable-cup”) |
| degradationData | mapping of material → { years, degradesIn } |
| recycled | boolean indicating if the metal cup is recycled immediately |

1. Use Case 6 – Animate Ocean Plastic Accumulation and Show Country Impact
   * 1. Use Case Description

|  |  |
| --- | --- |
| Use Case | Animate ocean plastic accumulation and show country impact. |
| Summary | A visitor in the Water section wants to see how plastic builds up in the ocean over time and learn about a particular country’s coastal‐waste risk and recycling rate. |
| Actors | The end user and the system. |
| Preconditions | The user has navigated to the “Water” page and clicked “A Plastic Ocean.” The app has already fetched the ocean‐projections array from /ocean-projections and the country list from /countries. |
| Basic Sequence | 1. The system displays the first projection year’s coverage, narrative text, and a static ocean with fish icons. 2. The user clicks the “Litter some plastic” button. 3. The system advances to the next projection: it raises the blue overlay height, plays the fish‐flee animation, updates the year/coverage text, and shows the new impact blurb. 4. At any time the user may change the country selector; the system fetches /country-impact/{code} and displays that country’s name, coastal\_waste\_risk, recycling\_rate, and per\_capita\_waste\_kg in the info panel. 5. The user clicks “Reset” to return to year 0 (coverage 0%) and restore the original narrative. |
| Exceptions | None. |
| Post Conditions | The ocean visualization, fish animation, impact text, and country‐impact panel all reflect the current projection step and selected country. |

* + 1. Mock Layout

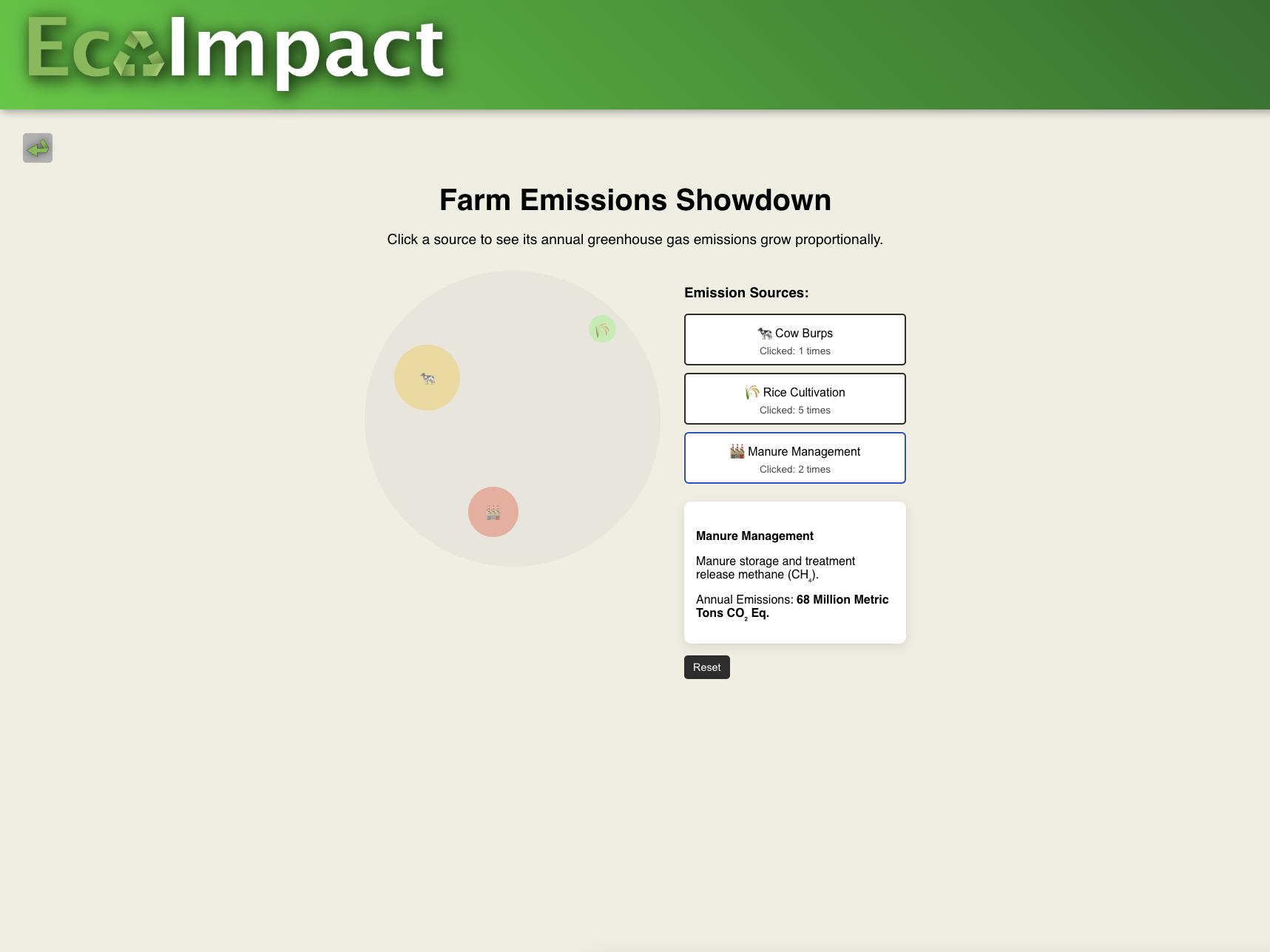


* + 1. Data Objects

|  |  |
| --- | --- |
| Data | Description |
| projections | list of { year, coverage, impact } |
| step | integer index in the projections list |
| coveragePercent | coverage × 100 |
| coverageAreaKm2 | coverage × 361,900,000 |
| overlayHeight | coverage × 2000 (capped at 100%) |
| countryList | list of { name, code } |
| selectedCountry | ISO code string |
| countryImpact | { country, per\_capita\_waste\_kg, recycling\_rate, coastal\_waste\_risk } |

1. Use Case 7 – Farm Emissions Showdown
   * 1. Use Case Description

|  |  |
| --- | --- |
| Use Case | Farm Emissions Showdown |
| Summary | A visitor in the Ground section wants to compare greenhouse‐gas sources from agriculture by clicking different icons and watching their relative “gas clouds” grow. |
| Actors | The end user and the system. |
| Preconditions | The user has navigated to the “Ground” page and clicked “Farm Emissions Showdown.” The client has the static emissionsData (cow, rice, industrial) loaded. |
| Basic Sequence | 1. The system renders three buttons (Cow Burps, Rice Cultivation, Manure Management) and three empty “gas‐cloud” divs. 2. The user clicks one of the source buttons (e.g. Cow Burps). 3. The system increments that source’s click count, increases its gas‐cloud div’s width/height by (annualEmissions × SCALE\_FACTOR), and displays the source’s name, description, and annualEmissions. 4. The user may repeat clicks to see the cloud grow further. 5. At any time the user clicks “Reset” to zero out all click counts, gas‐cloud sizes, and hide the info panel. |
| Exceptions | None. |
| Post Conditions | Each gas‐cloud’s size and click counter correctly reflect the cumulative clicks, and the information panel shows details for the last‐clicked source. |

* + 1. Mock Layout
    2. 
    3. Data Objects

|  |  |
| --- | --- |
| Data | Description |
| emissionsData | mapping of source → { name, description, annualEmissions } |
| clickCounts | { cow, rice, industrial } |
| gasSizes | { cow, rice, industrial } |
| selectedSource | string key of the last‐clicked source |

1. Tables and Descriptions

* EcoImpact persists all of its structured data in a single PostgreSQL database. The schema evolved directly from the CSV and SQL‐dump files in the datasets/ folder. Below is a concise catalog of the tables exposed by our API, together with a plain-English description of what each one holds.
* air\_super

This table tracks annual country-level CO₂ emissions broken down by source. Every row records one country’s emissions in a given year, summing coal, oil, gas, cement, flaring, and “other” into a total, and calculating per-capita output.

* water\_super

This table holds national plastic-waste statistics for “water” impact pages. Each row lists a country’s total waste generation, its main waste sources, recycling rate, per-capita waste (kg), and a coastal- waste-risk category.

* plastic\_projections

This table contains a time series of ocean-plastic projections. Each record has a year, a fractional coverage of the world’s oceans, and a short narrative describing global impact at that level.

* companies

This table aggregates corporate greenhouse-gas outputs. It stores the parent entity’s name and type (e.g. Investor-owned Company), plus its annual total emissions in million metric tonnes (MMT CO₂e).

* object\_degradation

This table enumerates everyday materials and how long they persist in the environment. Each entry names an object, classifies its type, and gives a time period (in years) until full degradation.

* 1. Storage
* All tabular data lives in PostgreSQL tables created at application startup from the SQLAlchemy models and/or loaded via our initial SQL dump (datasets/Setup/ecoimpact\_dump.sql). The digital assets that drive the UI—React bundles, CSS, PNG sequences, SVG icons, and short looping videos—reside in the frontend’s public/visuals directory and are served as static files by whichever web server hosts the React app. No client-side or in-memory caching beyond the browser’s normal HTTP cache is employed.
  1. Data Dictionary

air\_super

• country (VARCHAR): full country name • country\_code (CHAR): ISO‐ style short code (e.g. “USA”, “CHN”)

• year (INTEGER): calendar year of the data • total (FLOAT): total CO₂ emissions in million metric tonnes (MMT CO₂)

• coal, oil, gas, cement, flaring, other (FLOAT): sub-source emissions (MMT CO₂) • per\_capita (FLOAT): per-person emissions in tonnes

• number\_code (INTEGER): numeric ISO country code

water\_super  
• Country (VARCHAR): full country name  
• Total\_Waste (FLOAT): total plastic/municipal waste in million tonnes  
• Main\_Sources (VARCHAR): dominant waste-generation sectors/industries  
• Recycling\_Rate (FLOAT): national recycling percentage (0–100)  
• Per\_Capita\_Waste\_KG (FLOAT): kilograms of waste per person per year  
• Coastal\_Waste\_Risk (VARCHAR): qualitative risk category (“Low,” “Medium,” etc.)  
• Country\_Code (CHAR[3]): ISO code

plastic\_projections  
• year (INTEGER): projection year  
• coverage (FLOAT): fraction of global ocean surface covered by plastic (0–1)  
• impact (TEXT): narrative description of environmental consequences

companies  
• id (SERIAL INTEGER): internal primary key  
• parent\_entity (VARCHAR): company or group name  
• parent\_type (VARCHAR): ownership classification (e.g. “Investor-owned Company”)  
• total\_emissions (FLOAT): peak reported annual CO₂ emissions in MMT CO₂e

object\_degradation  
• object\_id (INTEGER): unique identifier  
• object\_name (VARCHAR): human-readable item name (e.g. “Plastic Cup”)  
• object\_type (VARCHAR): material category (e.g. “plastic”, “metal”)  
• time\_period (INTEGER): years until full environmental degradation or recycle cycle completion

1. Algorithm Analysis
   1. Big - O analysis of overall System and Sub-Systems
      1. Below is a walk-through of the major data-flows and interactive routines in EcoImpact, together with their time-complexity in Big-O terms. In every case, n refers to the size of the underlying data set (e.g. number of country records), m to the number of company records, p to the number of ocean-projection steps, f to the number of map features (~200), and y to the number of years plotted in a history graph (<10).

1. Data Ingestion Subsystem

• Reading raw CSV or SQL-dump files of D rows into PostgreSQL is linear time, O(D).

• Memory use is constant extra space beyond input streaming; final storage is O(D) in the database.

2. Backend API Layer  
 – Fetch all air\_super records (GET /air\_super): O(n) to scan n

rows.  
 – Filter by country\_code or year (WHERE … = indexed): O(log n + r) where r is number of matching rows.  
 – Fetch past five years by number\_code: same indexed lookup, O(log n + 5).  
 – Top-5 companies (GROUP BY + ORDER BY + LIMIT): O(m + m

log m) to group m rows then sort; in practice m is small and the LIMIT short-circuits.  
 – Ocean projections (SELECT \* FROM plastic\_projections):

O(p).

– Country impact (SELECT … FROM water\_super WHERE country\_code = …): O(log n).

3. Client Visualization Layer  
 – Map drawing (TopoJSON with f features): iterating and binding each SVG path is O(f), and each redraw on resize is O(f).  
 – Country toggle click: updating a Set is O(1); recomputing the

new global total by summing up to f – k included countries is O(f).

– Line graph rendering for y points: building scales, axes, and a single polyline is O(y).

– Legend draw and static elements (stops, rect, axis): constant bound independent of n, O(1).

– Carbon-Comparison input → calculation: all arithmetic is O(1). Animating two circles over T frames (~100) is O(T).

– Companies dropdown populate (fetch m records): rendering m options is O(m).

– Ice-Sheet slider update (0–10 ticks): image and text update in O(1) per slider event.

– Plastic-Ocean “litter” step (p steps): updating overlay height, fish icons (constant count ~5), and text is O(1) per step.

– Stick-Around material select: image swap, narrative text and timeline marker animate in O(1).

– Farm Emissions button click: updating one of three “gas clouds” and a counter in O(1).  
 4. Overall System Complexity

– Any single API call or visualization update runs in time proportional to the size of the data slice it touches (typically O(n), O(f), or O(m)), with small constant factors.

– Worst-case end-to-end browser response to a map click is O(f) for DOM updates plus O(f) for summing totals, i.e. O(f).

– Memory usage in the browser scales with the largest data set currently loaded—typically O(n) records in a JavaScript array, which for n≈200–300 is negligible.  
  
In practice all interactive paths stay in the low hundreds of items, so even “linear”  
behaviors complete instantly on modern hardware.

**6. Project Scrum Report**

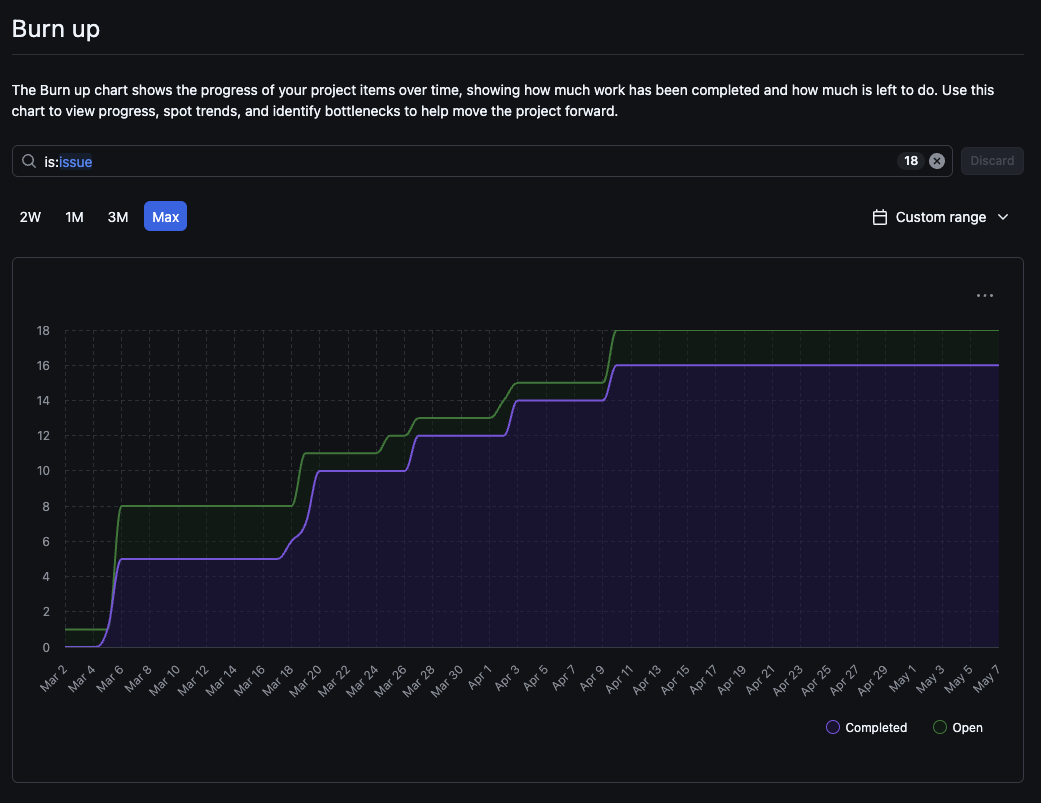
1. Implementation Plan – Sprint Timeline

|  |  |
| --- | --- |
| **Week** | **Ivan R.C: Backend and Data Functionally** |
| Week 1  02/20 - 02/27 | Creating the Database and getting the data from API or public datasets. Study the data to cleanse the data and organize the data. Helping setting up the backend framework. |
| Week 2  02/28 – 03/06 | Week 1: Data Manipulation and Cleaning, apply data to backend and database. |
| Week 3  03/07 – 03/20 | Week 2: Data Manipulation and Cleaning, apply data to backend and database.  -Apply new data to the visualizations when ready.  - Apply CRUD calls to the data and API to Super pages. |
| Week 4  03/21 – 03/27 | Week 1: Finishing up with Data Analysis: Apply the newly formed data to complete super visualizations. First round of testing with the data reaction to visualization. |
| Week 5  03/28 – 04/03 | Week 2: Applying the formed data to the visualizations, helping and adjusting backend and frontend if need to apply the the data to the visualizations. Second Round of testing. |
| Week 6  04/04 – 04/10 | Final adjustments and testing with application and data. |

|  |  |
| --- | --- |
| **Week** | **Benjamin W: Full stack** |
| Week 1  02/20 - 02/27 | * Create the initial frontend React “Air”, “Water” and “Ground” super pages in a “dummy” format. * Collect at least half of the hard data needed for the initial use cases (data apart from the Apis) |
| Week 2  02/28 – 03/06 | * Create the initial action pages in a “dummy” format. * E.g. in the “National Impact” section:   + Give a placeholder for the interactive map and other interactive elements |
| Week 3  03/07 – 03/20 | * Make sure the super category pages are connectable to the home page and the action pages. * Begin feeding Api data into at least half of the action pages. * Make the action pages partially actionable (without the animations).   + E.g. User can select the ice sheets they want to melt but won’t see the animation that goes along with it. |
| Week 4  03/21 – 03/27 | * Make at least half of the action pages fully actionable with the Api and hard data provided from the backend.   + Work with Marwen A. to fill in the placeholder sections with animations provided by Marwen A. |
| Week 5  03/28 – 04/03 | * Make the other half of the action pages fully actionable with the Api and hard data provided from the backend.   + Work with Marwen A. to fill in the placeholder sections with animations provided by Marwen A. |
| Week 6  04/04 – 04/10 | * Test the flow of the super category pages and the action pages. Ensure all test cases pass. * Tie up any loose ends that are exposed during testing. |

|  |  |
| --- | --- |
| **Week** | **Marwen A: Frontend Presentation** |
| Week 1  02/20 - 02/27 | * Plan the animations needed for each scenario   + Home Page   + Air: Air pollution country simulator   + Water: water level animations   + Ground: Decomposition timeline * Make the base of the Air page’s 1st interactive segment   + Carbon Comparison   + Status Images for each CO2 level   + Winning animation for each side |
| Week 2  02/28 – 03/06 | * Finish the Air page’s animations and implement them * Make the base of the Water page’s interactive segment   + Ice sheet levels   + Water levels   + Selectable |
| Week 3  03/07 – 03/20 | * Finish the Water page’s animations and implement them * Make the base of the Ground page’s interactive segment   + Compostable types   + Timeline   + Compostable Status |
| Week 4  03/21 – 03/27 | * Finish the Water page’s animations and implement them * Make the base of the Air page’s 2nd interactive segment   + CO2 meter level by country   + Map with selectable countries   + Graphics of consequences |
| Week 5  03/28 – 04/03 | * Finish the Carbon Comparison page’s animations and implement them * Finish the animations for the home page |
| Week 6  04/04 – 04/10 | * Implement the animations for the home page * Check for any bugged animations and fixes |

1. GitHub Product Backlog
   * 1. https://github.com/users/littleivan12/projects/1/views/1
2. Burn Down/Up chart
   * 1. https://github.com/users/littleivan12/projects/1/insights?period=max



**7. Discussion**

1. Limitations and future work
   * 1. EcoImpact succeeds in weaving together corporate emissions, personal‐footprint calculators, sea‐level projections, plastic accumulation, and material-degradation timelines into a single cohesive interface. Yet there are gaps and places for improvement. The database is seeded from static CSVs and SQL dumps; we currently lack an automated ETL pipeline to pull in new data releases or real-time feeds. The air, water, and ground sections rely on fixed datasets—there is no support yet for live updates or granular drill-down beyond the existing endpoints. Our world map handles a few hundred country records fluidly, but if we added subnational regions or detailed industry-sector breakdowns we would need to rethink performance and perhaps introduce client-side spatial indexing or server-side tiling.
     2. On the front end, the visualizations are deliberately playful and lightweight, but they do not yet meet full accessibility standards: keyboard navigation, high-contrast modes, and ARIA labels require more attention. The layout, though responsive, would benefit from a dedicated mobile design rather than simple flex wrapping. We also currently do not have user accounts, history tracking, or social sharing, so personalization and community features remain on the roadmap.
     3. Looking ahead, a natural next step is to automate data ingestion from APIs such as Our World in Data or NASA’s climate services, triggering nightly database refreshes and alerting the UI to new records. We could integrate tree-cover change maps, temperature anomaly graphs, or interactive carbon budgets keyed to user goals. Adding authentication and user profiles would allow people to save scenarios, compare progress, and participate in challenges. Finally, packaging the entire stack into a Docker compose—or deploying behind a Kubernetes cluster with autoscaling—would simplify production operations and make EcoImpact ready for broader public release.
2. Other thoughts
   * 1. Building EcoImpact has underscored the power of combining narrative, interactivity, and real data to drive environmental literacy. By converting abstract metrics into sliders, animated maps, and circle-sized comparisons, we invite users to “feel” the scale of climate impacts rather than just read numbers. Keeping the code modular—with a clear separation between the FastAPI data layer and the React/D3 front end—gives us the flexibility to swap in new datasets or visualization libraries without a full rewrite.
     2. As we refine the UI, prioritize accessibility, and automate our data pipelines, EcoImpact can serve both as an educational tool for classrooms and a public dashboard for communities. The open-source nature of the project means other developers can contribute new pages—perhaps centered on soil health, renewable energy adoption, or regional air-quality alerts—and help us build a richer, more inclusive climate-action platform. With each new feature, we hope to keep the experience inviting and intuitive, so that understanding our world’s environmental challenges leads naturally to informed and empowered choices.

**8. Complete System**

1. GitHub Project Link: https://github.com/littleivan12/EcoImpact
2. Walkthrough Video Link: https://youtu.be/NU14Ic7x9K8
3. Other thoughts
   * 1. We hope EcoImpact will better help you to understand the various climate change data and inspire you to take meaningful action on one of the greatest challenges of our time.