ChatGPT

AI-Driven LCA Platform for Metals (SIH 2025)

1. LCA and Circularity Concepts

Life Cycle Assessment (LCA) quantifies a product's environmental impacts across **every phase of its life** – from raw material extraction through manufacturing, use, and end-of-life disposal or recycling [1]. In practice, LCA requires data on inputs (materials, energy, transport) and outputs (emissions, waste) for each stage. **Circularity** (the circular economy) is the idea that materials should circulate through reuse, remanufacturing, and recycling rather than being wasted. In a circular economy, "materials never become waste" and products are kept in use as long as possible. Metals like aluminum and copper are especially suited to circularity because they can be recycled many times with little quality loss, drastically cutting energy and CO_2 (for example, recycled aluminum uses only ~5% of the energy of primary production).

Figure: The "butterfly" diagram of the circular economy, showing continuous material flows through use, recycling, and regeneration. In a circular model, products/materials are kept in circulation and waste is minimized.

LCA and circularity complement each other: modern LCA frameworks explicitly model recycling and reuse loops. For metals, a circularity-aware LCA will track not just emissions, but also how much material is reused or recycled and thus avoid virgin mining. As one review notes, LCA is "emerging not just as a tool for measuring environmental impact, but as a key strategy for advancing circularity" in industries like metallurgy. By comparing different scenarios (e.g. all-new aluminum vs. partially recycled aluminum), LCA helps decision-makers design products and processes that **minimize waste** and keep resources in use longer.

2. Problem Statement Requirements

The hackathon problem calls for an **intuitive AI-powered LCA software** for metals (especially Al and Cu) that emphasizes circularity. Key requirements include:

- **Flexible Input Interface:** Users should be able to input or select all relevant process details, including choice of raw vs. recycled material routes, energy sources, transport modes, and end-of-life options. For example, a user might select "primary aluminum smelting" vs "secondary (recycling) aluminum", or choose different transportation distances.
- **AI/ML Estimation:** The tool must **fill data gaps** automatically. Machine learning models or heuristics should estimate missing LCA parameters (e.g. if the user doesn't know an exact emission factor) and **predict circularity indicators**. This might include calculating recycled content, resource efficiency, or potential extension of product life based on the inputs. In other words, even with sparse inputs, the tool uses AI to compute likely environmental and circular metrics.
- **Flow Visualization:** The platform should graphically visualize material and energy flows across the value chain, highlighting circular loops vs. losses. This means generating **Sankey diagrams** or flow charts that show how much metal is reused or recycled at each stage and how much goes to waste. These visuals help users grasp where impacts occur.

- **Scenario Comparison:** Users must be able to compare a **conventional (linear)** pathway against a **circular** scenario side-by-side. For example, compare a process using 100% virgin copper versus one with 50% recycled copper, and immediately see differences in emissions or resource use. The platform should allow toggling scenarios or running "what-if" analyses.
- **Automated Reporting:** After analysis, the tool should generate clear, actionable reports. These would summarize key metrics (carbon footprint, energy use, recycled content) and provide recommendations for improving sustainability (e.g. "increase recycled input by X% to save Y kg CO_2"). Reports should be understandable even by users with limited LCA expertise. The emphasis is on **accessible outputs** – charts, dashboards and written summaries – so non-experts can make data-driven decisions.
- **Ease-of-Use:** Overall, the platform must cater to users with *minimal technical background*. This implies a clean GUI, guided input (e.g. dropdowns or wizards), and sensible defaults. Complex algorithms should run in the background (AI inferring missing data) without requiring the user to be an LCA specialist. The system should even work in a quasi-"real-time" fashion: as users tweak parameters, the impacts and circularity scores update quickly, giving immediate feedback on the consequences of different choices.

3. Proposed Modules and Features

A practical prototype can be divided into functional modules and features:

- **User Input Module (UI):** A React-based front end with forms or selectors for process parameters. For example, users pick metal (Al/Cu), specify raw vs. recycled fraction, input energy consumption (or select grid mix vs renewables), choose transport modes and distances, and set end-of-life options (landfill vs. recycling). The UI can include tooltips or presets (e.g. "default emission factor for coal power"). This module gathers all data needed for the LCA.
- **AI/ML Layer:** A backend service (e.g. Python FastAPI) that runs machine-learning models to **estimate missing data**. For instance, if a user omits a value, the AI might use regression or nearest-neighbor on historical LCA data to fill it. It would also **predict circularity metrics**: e.g. calculate expected recycled content or suggest improvements. In a hackathon prototype, these models can be simplistic (like linear fits from sample data) or even stubbed (returning fixed values) just to show functionality. TensorFlow or scikit-learn could be used to train quick models on sample LCA data, but for a 2-day demo, hard-coded or rule-based estimates (based on documented averages) are acceptable. The key is to simulate the "smart" behavior.
- **Flow Visualization (Sankey Diagrams):** A visualization module that takes the calculated material/energy flows and draws them. Sankey diagrams are ideal for this purpose. For example, one diagram might show tons of copper coming from primary mining vs scrap, flowing through smelting, usage, and then splitting into recycled feed vs waste. See below for an example:

Example Sankey diagram of copper flows through extraction, use, and recycling. Sankey diagrams clearly highlight how much material follows a circular loop versus how much is lost. In the prototype, libraries like D3.js (with the d3-sankey plugin) or React-based charting libraries can render these flows. Next to the diagram, a KPI dashboard can display key figures (total CO_2, percentage recycled, energy use).
- **Scenario Engine:** This feature allows creating "what-if" scenarios. Under the hood, it might be simple: run the LCA twice with different inputs (e.g. "100% virgin" vs "50% recycled") and show the differences. The UI could have buttons or toggles ("Conventional vs. Circular"), and the platform highlights the delta (e.g. "Circular route saves 30% CO_2"). Technically, this means structuring the app so users can clone or parametrize an analysis. For a prototype, the scenarios can be limited (e.g. just two presets), but the UI should allow the comparison charts to appear side-by-side.
- **Report Generator:** A component that compiles all results into a user-friendly report. This might be a PDF or on-screen summary with charts and text. It should include explanations of the results (e.g.

“Using more recycled content reduces the carbon footprint by X%”) and recommendations (e.g. “Consider switching energy source to reduce impact”). Automating report generation can use Python libraries like ReportLab or simply export HTML from React. The goal is that when a user finishes inputting data, the report is a one-click download. Even if the report content is largely static (not fully data-driven) in a quick prototype, the structure should be there.
- **Integrations (Optional Stretch Features):** Time permitting, the platform could mock up advanced integrations. For example, **IoT connectivity** might mean auto-fetching real plant data (e.g. a CSV of electricity use from a factory sensor) and feeding it into the LCA. **Blockchain traceability** could mean demonstrating how material provenance (e.g. certified recycled content) is verified via an immutable ledger. **Digital Twin simulation** might involve using a physics engine or a simplified simulation to predict process changes. In a 2-day demo, these would be placeholders: e.g. a button labeled “Fetch live data” that just loads sample data, or a “Blockchain proof” PDF icon. Mentioning them shows awareness of future scope, but they do not need full implementation.

4. Technology & Implementation Guidance

To build this prototype quickly, use familiar web and data tools with minimal setup:

- **Frontend (React):** Create a single-page app using React (with Create React App or Next.js) for the UI. Use component libraries (e.g. Material-UI) for form elements and layout. For charts, libraries like D3.js or Recharts (which has a Sankey component) can draw the Sankey diagram and bar/line charts for KPIs. Keep the UI simple: e.g. forms on one side, a “Generate” button, and results (diagram + charts) displayed below or to the side. Even a bare-bones UI (no styling) is acceptable for a hackathon.

- **Backend (FastAPI + Python):** Implement REST endpoints using FastAPI. For example, POST `/run_lca` can accept JSON input (user choices) and return computed results (impact metrics). Inside, use Python libraries (pandas, numpy) to process data. The AI estimation step can call a dummy function or a lightweight ML model (e.g. a pre-trained TensorFlow/Keras model saved as an `.h5` file, or even a scikit-learn model pickled). Given the short time, you can **mock** the AI logic: for instance, if a value is missing, just look up a default in a JSON file or compute a simple formula. Focus on the architecture rather than accuracy.

- **Database (PostgreSQL or SQLite):** Use PostgreSQL (with SQLAlchemy in Python) to store any example inventory data or user projects. However, for a demo you might skip a full DB and just use in-memory structures or JSON files. If persistence is needed, SQLite (no setup) is quick. You could store, for example, saved user inputs or retrieved emission factors. But it’s optional: your FastAPI can simply use hard-coded lookup tables for emissions (e.g. `{ "electricity_coal": 0.9, "electricity_hydro": 0.1 }` kg CO_2/kWh, etc.).

- **Containerization (Docker/Kubernetes):** Containerize the app with Docker to ensure portability. Write a `Dockerfile` for the backend and one for the frontend, or a multi-stage build. In a 2-day build, a simple Docker Compose file can spin up both services together. Kubernetes is probably overkill for a prototype; you can mention it as an option for deployment scaling. The key is that using Docker helps run the entire stack (React, FastAPI, DB) easily.

- **Simplifications:** To save time, some modules can be stubbed or simplified. For example, rather than a fully dynamic Sankey, you could pre-generate a Sankey image (like the one above) and just swap it in. The AI model can be a trivial function (“if user didn’t specify recycled %, assume

0.3"). The report generator can output a static PDF template with a few variables filled in. Prioritize making the main flow work end-to-end: user inputs → calculations → charts → report.

- **Collaboration Tips:** Divide work: one team member on frontend forms and charts, another on the API and logic. Use Git for code sync. Keep modules loosely coupled (e.g. agree on JSON schema between frontend/backend). Test with sample data. Because the demo is short, don't get bogged down in polished UI—focus on a proof-of-concept that **works** and shows each feature.

5. Data Sources for Prototyping

For a demo, you can use a mixture of open datasets and synthetic data:

- **Life Cycle Inventory Databases:** Open LCA datasets can supply process emissions. For example, the European Reference Life Cycle Database (ELCD) provides basic LCI data for many materials (free to download). The International Aluminium Institute (IAI) publishes detailed LCI data for aluminum production; their **2019 LCI dataset** can serve as a realistic reference. (Similarly, the International Copper Association has data on copper recycling.) While these full databases may not be free, you can extract a few key numbers (e.g. CO_2 per tonne of primary Al, per tonne of recycled Cu) and hard-code them.

- **Emission Factor Libraries and APIs:** Use emission factor collections for transport and energy. Public APIs like Climatiq offer carbon intensity factors for fuels and activities. The **Open Grid Emissions** initiative provides free, validated hourly CO_2 intensity for the U.S. power grid. (India-specific data might be limited, so you can use generic figures or convert energy mixes.) For transport, governments often publish average emissions: for example, Defra (UK) or the IPCC supply factors like "0.2 kg CO_2 per tonne-km for truck transport." In a demo, you might simply store a JSON of representative factors (e.g. `{ "truck": 0.1, "rail": 0.02 }`) and use them.

- **Industry Data and Reports:** Use high-level statistics for illustrative numbers. For instance, the Aluminum Association notes that recycling aluminum saves **95% of the energy** compared to primary production; you could use this to set your recycling benefit. The Copper Alliance states that copper is infinitely recyclable and recycled copper requires much less energy. These facts can seed your AI's predictions. Also, datasets from the global steel or mining industry (e.g. World Steel's sustainability report) could give context.

- **Sample/Product Data:** If you need example products or processes, simple cases suffice. E.g. consider an "aluminum ingot production" process with fixed inputs (from literature or open LCA studies). The International Aluminium Institute factsheets or **openLCA Nexus** (OzLCI2019) have typical values. For demonstration, even a made-up mini-database of a few processes (in an Excel or JSON) can populate the system.

- **Mock JSON APIs:** For ease, you can simulate "external data" by serving static JSON. For example, set up a FastAPI endpoint `/emission_factors` that returns a JSON of factors. Or just include a JavaScript file with hardcoded data. As long as the flow is realistic (data is fetched, parsed, used in calc), it illustrates the point.

The goal is not perfect accuracy but plausibility. Cite known values or sources where possible (as above) to justify your choices. Using these data, the prototype can calculate sample impacts (e.g. "1 tonne of recycled Al saves ~9 tonnes of CO_2") to make the demo insightful.

6. Expected Impact for Industry

A successful platform will **empower metal industries** to make more sustainable, circularity-driven decisions. By automating LCA and presenting results clearly, even non-experts (engineers, managers) can see trade-offs between, say, using scrap vs. virgin metal. The tool translates complex data into *actionable insights*: companies can identify "hotspots" (e.g. transport or energy use) and test circular strategies. As the problem statement notes, such a tool "empowers the metals sector to make practical, data-driven choices that foster environmental sustainability while advancing circular, resource-efficient systems".

In real-world terms, this means: designers might choose a product design that is easier to disassemble and recycle; smelters might adjust energy sources when they see the emissions savings; supply chain teams could source higher recycled-content alloys to cut carbon footprints. Over time, the platform could help Indian metal firms meet regulatory targets (e.g. lower GHG emissions), qualify for green certifications, and reduce costs by improving resource efficiency. By providing **real-time feedback** on decisions, the tool turns sustainability from an abstract goal into a concrete, measurable outcome. In short, the AI-powered LCA platform guides industries toward a circular economy – reducing waste, cutting emissions, and keeping valuable metals in use longer.

Sources: Hackathon problem statement; LCA and circular economy literature [1]; industry data (Aluminum/Copper associations); sustainability reports.

---

[1] Life Cycle Assessment (LCA) – Everything you need to know | Ecochain
https://ecochain.com/blog/life-cycle-assessment-lca-guide/