GRADO EN INGENIERÍA MATEMÁTICA E INTELIGENCIA ARTIFICIAL

TRABAJO FIN DE GRADO

Plataforma de Modelado Financiero para Promover la Adopción de Tecnologías de Cocinado Limpio

Financial Modelling Platform to Promote the Adoption of Clean Cooking Technologies

Autora: Almudena Garrido Garcia-Pita

Director: Pablo Duenas Martinez

Co-Director: Santos Díaz-Pastor

Madrid
Mayo de 2026

Declaration of Originality

I declare under my responsibility that the Project presented with the title **Financial Modelling Platform to Promote the Adoption of Clean Cooking Technologies** at the ICAI School of Engineering of the Comillas Pontifical University in the academic year 2026 is of my authorship and has not been presented previously for other purposes. The Project is not plagiarised from any other, either totally or partially, and the information that has been taken from other documents is duly referenced.

Use of Artificial Intelligence[1]

I declare under my responsibility that (select the correct option):

☐ I have not used Artificial Intelligence in the preparation of this document.

☒ I have used Artificial Intelligence in the preparation of this document and/or Annex B under the conditions allowed by Comillas Pontifical University, i.e. applying Level 2 of the Perkins et al. (2024) Assessment Scale: *"AI can be used for pre-task activities such as brainstorming, description and initial research. This level focuses on the use of AI for planning, synthesising and generating ideas, but assessments should emphasise the ability to develop and refine these ideas independently"*. Specifically, Artificial Intelligence has been used to:

AI tools were used under the supervision of the TFG advisor to assist with manuscript structuring and editing, support exploratory methodological discussions, improve coding efficiency, and refine language and translation.

GARRIDO GARCIA-PITA ALMUDENA - 54210620B (sign here) Firmado digitalmente por GARRIDO GARCIA-PITA ALMUDENA - 54210620B Fecha: 2026.05.25 22:01:23 +02'00'
Signature: Almudena Garrido Garcia-Pita
Date: 25/05/26

[1]This declaration refers to the use of generative Artificial Intelligence to carry out the Project documents. It does not apply to Projects where, by their nature, artificial intelligence must be used as part of them (application of machine learning techniques, neural networks, data analysis...).

Authorisation for Project Delivery

Thesis Supervisor	Thesis Co-Supervisor
Pablo Duenas Martinez Digitally signed by Pablo Duenas Martinez Date: 2026.05.25 21:37:16 +02'00' (sign here)	DIAZ PASTOR SANTOS-JOSE - 53749713R Firmado digitalmente por DIAZ PASTOR SANTOS-JOSE - 53749713R Fecha: 2026.05.25 21:34:22 +02'00' (sign here)
Signature: Pablo Duenas Martinez	Signature: Santos Díaz-Pastor
Date: 25/05/26	Date: 25/05/26

GRADO EN INGENIERÍA MATEMÁTICA E INTELIGENCIA ARTIFICIAL

TRABAJO FIN DE GRADO

Plataforma de Modelado Financiero para Promover la Adopción de Tecnologías de Cocinado Limpio

Financial Modelling Platform to Promote the Adoption of Clean Cooking Technologies

Autora: Almudena Garrido Garcia-Pita

Director: Pablo Duenas Martinez
Co-Director: Santos Díaz-Pastor

Madrid
Mayo de 2026

Acknowledgements

This project would not have been possible without the education I received at ICAI, which gave me the technical grounding and analytical rigour needed to tackle a problem as complex as the one this platform addresses. I am also deeply grateful to IIT for giving me the opportunity to work on this: being able to contribute, however modestly, to something with this kind of real-world impact is not something I take for granted.

I want to thank Pablo Dueñas in particular, my supervisor, for his dedication, his availability, and for trusting me throughout the entire process. And Santos Díaz-Pastor, whose work on the NICCP reference model was the foundation on which this platform was built and whose collaboration was essential during development.

To SEforALL, for believing in this tool and committing to use it. Knowing that the platform responded to a real need, and that the team would put it to genuine use, was a constant source of motivation throughout development. Their feedback as users was also instrumental in making CC-WBT a truly functional tool, and I am grateful for their trust in giving me the chance to contribute, however small, to such a complex and important challenge.

Plataforma de Modelado Financiero para Promover la Adopción de Tecnologías de Cocinado Limpio

Autora: Almudena Garrido Garcia-Pita
Director: Pablo Duenas Martinez
Co-Director: Santos Díaz-Pastor
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

Introducción

Más de dos mil millones de personas siguen cocinando hoy en día con biomasa sólida, fogones abiertos o estufas rudimentarias. La gran mayoría viven en África Subsahariana, Asia Central y Meridional o el Sudeste Asiático, y es un problema que continúa estando muy lejos de estar resuelto: el Banco Mundial estima que en 2030 el 60 % de quienes aún carezcan de acceso a la cocina limpia seguirán viviendo en África Subsahariana, donde el crecimiento demográfico supera el 2,5 % anual y avanza más rápido que el progreso. La contaminación del aire interior por combustión de biomasa provoca alrededor de 3,7 millones de muertes prematuras al año según la OMS, y son las mujeres y los niños quienes más sufren las consecuencias, tanto por el tiempo que pasan junto al fuego como por las horas dedicadas a recoger leña.

Por lo tanto, la pregunta no es si esta transición tiene que ocurrir, sino cómo financiarla a escala nacional. Los modelos tecno-económicos son capaces de proyectar demanda y comparar estrategias de despliegue, pero lo que no proporcionan es un cuadro financiero completo: estados de ingresos y gastos, balance, flujo de caja, brecha de viabilidad. La experiencia de IIT Comillas construyendo ese modelo en Excel para el Plan Nacional Integrado de Cocina Limpia de Ruanda (NICCP), en colaboración con SEforALL, dejó claro que hacía falta una herramienta más transparente y de código abierto que cualquier equipo pudiera usar sin empezar desde cero.

Objetivos

El objetivo central de este TFG es construir una plataforma de modelado financiero de código abierto para transiciones de cocina limpia a escala nacional: una herramienta que tome los resultados de análisis tecno-económicos como punto de partida y los

convierta en evaluaciones financieras integradas, de modo que un planificador pueda evaluar no solo cuánto cuesta una estrategia de despliegue, sino si es financieramente viable y qué se necesitaría para que lo sea.

Para ello se tienen que gestionar múltiples escenarios en paralelo, calcular los tres estados financieros como un sistema encadenado que cuadra en cada periodo, y mantener la lógica financiera en ficheros de configuración auditables sin tocar el motor de cálculo. La automatización tiene que ser suficiente para que los planificadores dediquen su tiempo a interpretar resultados, y cada escenario debe poder exportarse como un paquete reproducible.

Descripción de la plataforma

CC-WBT (Clean Cooking Web-Based Tool) es una plataforma web desarrollada en Python con un frontend en Streamlit y un backend en FastAPI. El frontend guía al usuario a través de un flujo estructurado de entrada de datos, con validación en tiempo real que impide que un error se propague silenciosamente por los estados financieros. El backend gestiona toda la computación, la persistencia de datos y los ficheros Excel de cada escenario.

Su núcleo es el *ExcelFormulaProcessor*, un motor declarativo impulsado por *formulas_map.json*. En ese fichero se escribe cada fórmula financiera de forma genérica, con marcadores de posición (`{country}`, `{model}`, `{fuel}`) en lugar de referencias a escenarios concretos. En tiempo de ejecución, el motor aplica una expansión cartesiana que instancia cada fórmula para cada combinación activa, de modo que una sola entrada puede expandirse en decenas de fórmulas sin intervención manual.

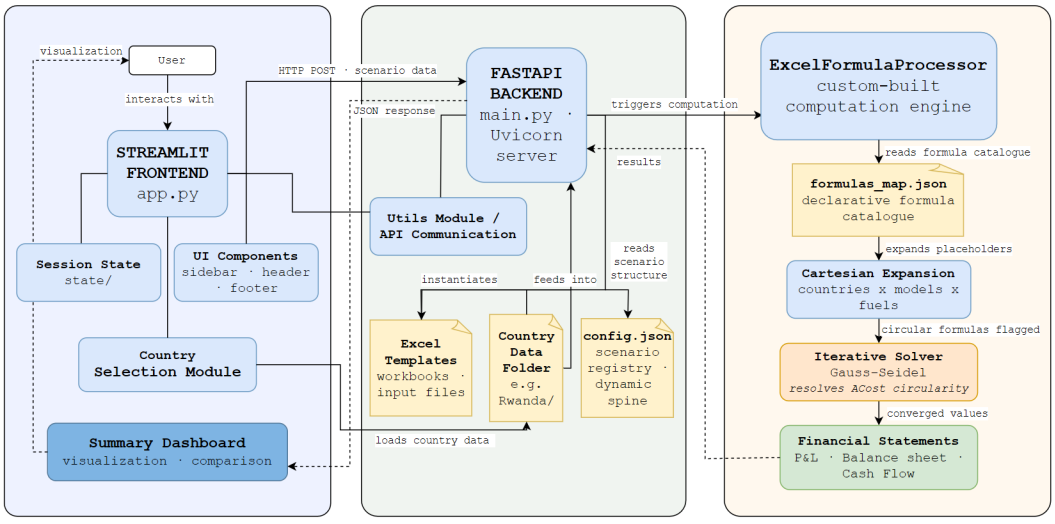


Figure 1: Flujo de información a través de las tres capas de CC-WBT: frontend (Streamlit), backend (FastAPI) y motor de cálculo (ExcelFormulaProcessor).

El modelo financiero se construye sobre un marco de coste de servicio. El *Regulatory Asset Base* (RAB) registra el valor neto de los activos en servicio; el *Annual Cost*

of Service (ACoSt) cuantifica lo que cuesta prestar el servicio cada año; y la diferencia entre ACoSt e ingresos tarifarios determina los *Long-Term Subsidies* (LTS). Estas magnitudes tienen una definición circular que se resuelve mediante un solver iterativo de Gauss-Seidel.

Resultados

La plataforma se validó con el caso de estudio de Ruanda, comparando sus resultados con el modelo de referencia del NICCP variable a variable, año a año y para cada mercado de combustible, sobre tres escenarios (Baseline, CleanStep y Aligned) en un horizonte 2023–2034.

La validación confirmó que el balance cuadra en todos los periodos con discrepancias del orden de 10^{-10} M$, y que el solver converge en dos o tres iteraciones con residuos de orden 10^{-13}, varios órdenes de magnitud por debajo de la tolerancia de parada. Más allá de los checks numéricos, la plataforma ha sido puesta a disposición de SEforALL para su uso en la planificación financiera de transiciones de cocina limpia, lo que supone también una validación práctica de su funcionamiento.

Conclusiones

CC-WBT nace para cubrir una brecha concreta: no existía una herramienta integrada, auditable y de código abierto para la planificación financiera de transiciones de cocina limpia a escala nacional. Esta plataforma automatiza todo el proceso manteniendo la lógica financiera en ficheros de configuración que cualquiera con conocimiento del dominio puede leer y verificar.

El proyecto conecta con los ODS 7, 3, 5 y 13, y se dirige a tres tipos de usuario: el ministerio que necesita la capa financiera que los modelos tecno-económicos no cubren, el banco de desarrollo que necesita un cuadro auditable antes de comprometer capital, y el equipo de investigación que busca una base extensible sin partir de cero. La plataforma está disponible públicamente en `https://github.com/SEforALL-IEAP/CC-WBT`.

Palabras clave: cocina limpia, modelización tecno-económica, planificación financiera, Python, Streamlit, FastAPI, análisis de escenarios, acceso a la energía, coste de servicio, base de activos regulatorios, código abierto.

Referencias

[1] F. de Cuadra, P. Dueñas, E. Sánchez-Jacob *et al.*, "Models and Tools for Integrated Clean Cooking Planning: Case Example of Rwanda," IIT–MIT Working Document, 2024.

[2] S. J. Díaz-Pastor and I. J. Pérez-Arriaga, "An integrated regulatory–financial proposal for universal electrification in Uganda," *Energy Economics*, Elsevier, 2025.

Financial Modelling Platform to Promote the Adoption of Clean Cooking Technologies

Author: Almudena Garrido Garcia-Pita
Supervisor: Pablo Duenas Martinez
Co-Supervisor: Santos Díaz-Pastor
Collaborating Entity: ICAI – Universidad Pontificia Comillas

PROJECT SUMMARY

Introduction

More than two billion people still cook every day over open fires or with rudimentary solid-fuel stoves. The vast majority live in Sub-Saharan Africa, Central and Southern Asia, or South-East Asia, and this is a problem that remains very far from resolved: the World Bank estimates that by 2030, 60 % of those still lacking clean cooking access will live in Sub-Saharan Africa, where population growth exceeds 2.5 % per year and is outpacing progress. Household air pollution from solid-fuel combustion causes around 3.7 million premature deaths a year according to the WHO, and it is women and children who suffer the consequences most, both from the time they spend near the fire and from the hours devoted to collecting firewood.

The question, then, is not whether this transition needs to happen, but how to finance it at country scale. Techno-economic models can project demand and compare deployment strategies, but what they do not provide is a complete financial picture: income statements, balance sheets, cash flows, the viability gap. IIT Comillas' experience building that model in Excel for Rwanda's National Integrated Clean Cooking Plan (NICCP), in collaboration with SEforALL, made it clear that a more transparent, open-source tool was needed — one that any team could use without starting from scratch every time.

Objectives

The core objective of this thesis is to build an open-source financial modelling platform for clean cooking transitions at country scale: a tool that takes the outputs of techno-economic analyses as its starting point and turns them into integrated financial

assessments, so that planners can evaluate not just what a deployment strategy costs, but whether it is financially viable and what it would take to make it so.

To achieve this, the platform must manage multiple scenarios in parallel, compute the three financial statements as a chained system that balances every period, and keep all financial logic in auditable configuration files without touching the computation engine. The level of automation must be high enough that planners spend their time interpreting results, and every scenario must be exportable as a reproducible bundle.

Platform Description

CC-WBT (Clean Cooking Web-Based Tool) is a web platform developed in Python with a Streamlit frontend and a FastAPI backend. The frontend guides the user through a structured data-entry flow, with real-time validation that prevents errors from propagating silently through the financial statements. The backend handles all computation, data persistence, and the Excel files for each scenario.

Its core is the *ExcelFormulaProcessor*, a declarative engine driven by *formulas_map.json*. In that file, each financial formula is written generically, with placeholders (`{country}`, `{model}`, `{fuel}`) instead of references to specific scenarios. At runtime, the engine applies a Cartesian expansion that instantiates each formula for every active combination, so a single entry can expand into dozens of formulas without manual intervention.

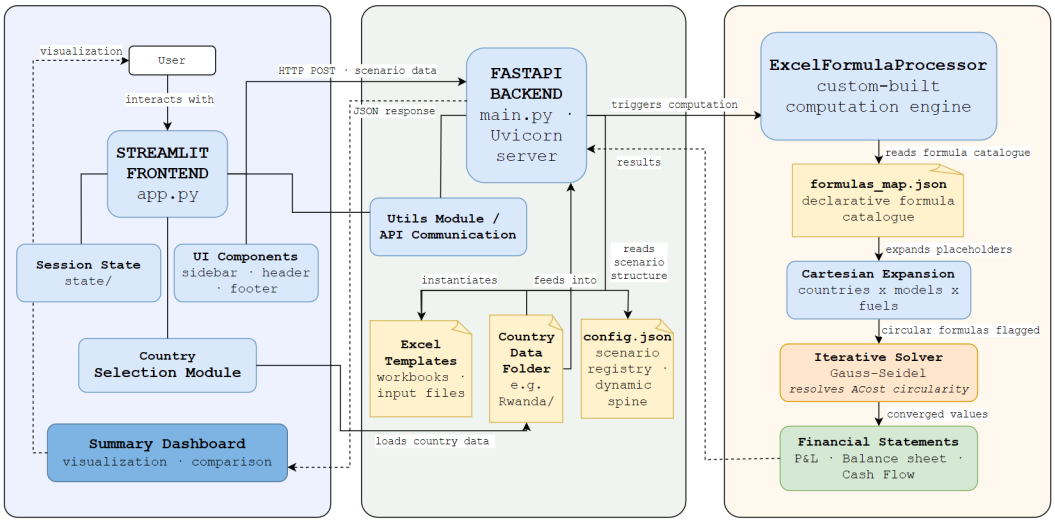


Figure 2: Information flow through the three layers of CC-WBT: frontend (Streamlit), backend (FastAPI) and computation engine (ExcelFormulaProcessor).

The financial model is built on a cost-of-service framework. The *Regulatory Asset Base* (RAB) tracks the net value of assets in service; the *Annual Cost of Service* (ACoSt) quantifies what it costs to provide the service each year; and the gap between ACoSt and tariff revenues determines the *Long-Term Subsidies* (LTS). These quantities have a circular definition that is resolved by a Gauss-Seidel iterative solver.

Results

The platform was validated against the Rwanda case study, comparing its outputs with the NICCP reference model variable by variable, year by year, and for each fuel market, across three scenarios (Baseline, CleanStep and Aligned) over a 2023–2034 horizon.

Validation confirmed that the balance sheet closes in every period with discrepancies of order 10^{-10} M$, and that the solver converges in two or three iterations with residuals of order 10^{-13}, several orders of magnitude below the stopping tolerance. Beyond the numerical checks, the platform has been made available to SEforALL for use in clean cooking financial planning, which also constitutes a practical validation of its performance.

Conclusions

CC-WBT was built to fill a concrete gap: there was no integrated, auditable, open-source tool for the financial planning of clean cooking transitions at country scale. This platform automates the entire process while keeping the financial logic in configuration files that anyone with domain knowledge can read and verify.

The project connects directly with SDGs 7, 3, 5 and 13, and addresses three types of user: the ministry that needs the financial layer that techno-economic tools do not cover, the development bank that needs an auditable picture before committing capital, and the research team looking for an extensible base to build on without starting from zero. The platform is openly available at `https://github.com/SEforALL-IEAP/CC-WBT`.

Keywords: clean cooking, techno-economic modelling, financial planning, Python, Streamlit, FastAPI, scenario analysis, energy access, cost of service, regulatory asset base, open source.

References

[1] F. de Cuadra, P. Dueñas, E. Sánchez-Jacob *et al.*, "Models and Tools for Integrated Clean Cooking Planning: Case Example of Rwanda," IIT–MIT Working Document, 2024.

[2] S. J. Díaz-Pastor and I. J. Pérez-Arriaga, "An integrated regulatory–financial proposal for universal electrification in Uganda," *Energy Economics*, Elsevier, 2025.

[3] I. J. Pérez-Arriaga, *Regulation of the Power Sector*. London: Springer, 2014.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Understanding CC-WBT starts with understanding the gap it was built to fill. This chapter covers what motivated the project, what the platform was built to do, how it connects to the broader development agenda, and how the rest of the document is structured.

1.1 Context and Motivation

Despite decades of international efforts, more than two billion people still cook every day over open fires or with rudimentary solid-fuel stoves. Most of them live in Sub-Saharan Africa, Central and Southern Asia, or South-East Asia. This is not a residual or fading problem. The World Bank projects that by 2030, roughly 60 per cent of those still without clean cooking access will live in Sub-Saharan Africa, where population growth above 2.5 per cent a year is outpacing progress. Left unaddressed, the gap will keep widening.

The costs are visible and well documented. Household air pollution from solid-fuel combustion is responsible for around 3.7 million premature deaths a year, according to the World Health Organization (WHO). Women and children bear the brunt of it, both because they spend the most time near the fire and because collecting fuelwood, which can take several hours a day, falls on them. Beyond the health toll, burning biomass at this scale accelerates deforestation and produces greenhouse gas (GHG) emissions that no climate plan can afford to ignore.

The question, then, is not really whether the transition needs to happen. It is how to finance it at country scale. And that is exactly where the tools available at the start of this project fell short. Techno-economic models like the Reference Electrification Model (REM), developed at IIT Comillas, or the planning frameworks behind national programmes such as Rwanda's National Integrated Clean Cooking Plan (NICCP), are well suited to projecting demand, sizing infrastructure and comparing deployment strategies. What they do not produce, at least not in any integrated or systematic way, is the financial picture: the profit and loss account, the balance sheet, the cash flow statement, the viability gap, the capital structure. The numbers that tell a planner whether a given strategy is actually financeable, and if not, how much public support it would take to make it so.

In practice, that financial layer has been handled through Excel models built from scratch for each study. Useful for a first pilot, hard to audit, and almost impossible to hand over to another team without losing half the logic along the way. IIT, working with Sustainable Energy for All (SEforALL), had already built one such model for

Rwanda, and that experience made the problem concrete. A more transparent, open-source tool was needed, one that any team could use, inspect and adapt without having to reassemble it from zero each time.

That is what this project set out to build. *CC-WBT* (Clean Cooking Web-Based Tool) is an open-source financial modelling platform for clean cooking developed in Python at IIT Comillas in collaboration with SEforALL. It combines a FastAPI back-end with a Streamlit frontend and keeps all financial logic in declarative configuration files that any technically informed user can read, check or adjust directly, without ever having to touch the computation engine itself. Throughout development, Rwanda's existing financial model served as the reference case against which the platform's outputs were validated.

1.2 Objectives

The core objective is an open-source financial modelling platform for clean cooking transitions at country scale: one that takes the outputs of techno-economic analyses as its starting point and turns them into integrated financial assessments, so that planners can evaluate not just what a deployment strategy costs, but whether it is financially viable and what it would take to make it so.

That top-level goal shapes a set of more specific constraints. The platform has to handle multiple country scenarios in parallel, each with several techno-economic models and fuel markets, running independent and coherent financial computations for every combination. The three financial statements (profit and loss, balance sheet and cash flow) need to be computed as a chained system, not independently, so that the balance sheet actually balances every period. Every financial formula needs to live in a configuration file that an informed user can read, check or modify without touching the computation engine itself.

Usability was treated as a constraint, not an afterthought. Input processing and scenario simulation should be automated enough that planners spend their time interpreting results rather than feeding data in by hand. Outputs should come through an interactive dashboard and structured numerical exports. And the whole platform needs to stay extensible without developer involvement, with every scenario exportable as a self-contained, reproducible bundle that any team can open and get exactly the same numbers.

1.3 Alignment with the Sustainable Development Goals

The project connects directly to four of the United Nations Sustainable Development Goals (SDGs). The most obvious one is SDG 7, Affordable and Clean Energy for All. Supporting the financial planning of clean cooking transitions at national scale is, by definition, working towards universal access to modern energy services.

SDG 3, Good Health and Well-Being, follows from the nature of the transition itself. Replacing solid-fuel combustion with cleaner alternatives reduces exposure to

household air pollution, which is among the leading environmental causes of respiratory disease and premature mortality, particularly for women and children in low-income settings.

SDG 5, Gender Equality, matters here because the burden of traditional cooking is not evenly distributed. Women and girls absorb most of it, in health terms and in time. Every hour saved from collecting firewood is an hour available for education or economic activity, and that is not a small thing.

The link to SDG 13, Climate Action, comes through the carbon module built into the platform. Replacing biomass combustion reduces GHG emissions and eases deforestation pressure, and the platform quantifies that directly, computing avoided emissions against a baseline and estimating what those reductions could be worth in carbon markets.

1.4 Thesis Structure

The rest of the document is organised into five chapters. Chapter 2 reviews the existing landscape of techno-economic and financial planning tools for clean cooking and electrification, introduces the regulatory and financial concepts behind the model, and identifies the gap this project addresses. Chapter 3 formulates the problem and describes the high-level solution: the system architecture, the data structure, the financial model, and the key design decisions that shaped the platform. Chapter 4 covers the full implementation: the dynamic computation engine and its declarative formula catalogue, the Cartesian expansion mechanism, the iterative solver that resolves circular dependencies, and the dashboard visualisation layer. Chapter 5 presents the validation against the Rwanda reference model, the balance sheet consistency checks, and the convergence tests on the iterative solver. Chapter 6 draws the main conclusions, reflects on the limitations of the work and outlines possible directions for future development.

Chapter 2

State of the Art

Understanding what CC-WBT does, and why it was worth building, requires knowing what was already out there. This chapter surveys the main tools and frameworks in the space where the project sits: techno-economic planning for energy access, financial and regulatory modelling for the power sector, and the specific instruments built to support clean cooking transitions. The aim is not to catalogue everything. It is to show what the field has achieved, where it still runs out of answers, and how those gaps made a project like this one necessary.

2.1 Computer-aided techno-economic planning

Bringing modern energy services to populations that lack them turns out to be, in essence, an optimisation problem. And a very large one. Millions of households scattered across territories of varying geography, income and infrastructure density each need to be connected via whichever technology is cheapest for their specific situation, taking into account what is happening in every surrounding community at the same time. No planner working with spreadsheets can handle that at national scale. Over the past two decades, the research community has built tools that can.

2.1.1 The Reference Electrification Model

The Reference Electrification Model, or REM, is one of the most complete examples of what these tools can do [2]. Developed at the joint MIT–Comillas Universal Energy Access Lab and described in detail in the 2018 IIT working paper by Amatya et al., REM is a computer-based optimisation tool designed to determine the least-cost way to electrify any given area. It works at the level of individual buildings. Given satellite imagery, population data, load profiles, and a catalogue of technology options, it finds for each household whether grid extension, a mini-grid, or a standalone solar system is the most cost-effective solution, and then produces the full network design to make that happen.

REM stands apart from earlier planning tools through its combination of scale and granularity. It can be applied to a single district or to an entire country, and it does not average out the population in the way simpler models do. Each consumer is modelled individually, with its own location, demand profile, and distance to the nearest existing infrastructure. The optimisation runs simultaneously over all of them, which means the solution is truly least-cost rather than an aggregation of locally optimal decisions that may be globally inefficient.

REM has been applied to real planning exercises in several countries in sub-Saharan Africa and South Asia. It produces detailed electrification roadmaps, identifies which villages should be connected via grid extension versus off-grid technologies, and estimates the capital and operating expenditure associated with each solution. These outputs are exactly what a national planning authority needs to set priorities and draft an investment programme.

2.1.2 Integrated clean cooking planning tools

Clean cooking has its own version of this. The Integrated Clean Cooking Planning Tool, or ICCP, was built as part of the collaboration between IIT Comillas, MIT, and the Sustainable Energy for All initiative (SEforALL), and applied in detail to the case of Rwanda, documented in the December 2024 working paper by de Cuadra et al. [1].

The ICCP extends the logic of techno-economic optimisation to a problem that is structurally different from electrification. In electricity planning, the key decision is which technology connects which household. In clean cooking, the decision also involves consumer behaviour, fuel markets, and adoption dynamics. Households do not switch fuels because a planner decides they should. They switch when the alternative becomes affordable, accessible, and culturally acceptable. A model that ignores this produces plans that look optimal on paper but that no one adopts in practice.

The approach used in the ICCP addresses this through what de Cuadra et al. describe as a multi-attribute optimisation. Rather than minimising a single cost metric, the tool balances several dimensions simultaneously: techno-economic efficiency, adoption rates, health outcomes, and environmental impact. The planner interacts with the tool iteratively, exploring different combinations of policies and infrastructure investments and observing their projected effects across all attributes. The Rwanda application modelled the entire national territory, covering urban and rural consumers, multiple fuel markets (biomass, LPG, electricity, and improved cookstoves), and a 20-year planning horizon.

The outputs are rich. The tool produces technology adoption trajectories, key performance indicators at district level, annual cash-flow estimates, resource and market projections, and a set of carefully post-processed results that can feed into further analysis. For the Rwanda exercise, it was the foundation on which the entire National Integrated Clean Cooking Plan was built.

2.1.3 What techno-economic models cannot do

The tools described above are genuinely impressive. The level of technical detail they handle, the scale at which they operate, and the quality of their planning outputs represent years of research and real-world application. But they were designed to answer technical and strategic questions, and there is a set of questions they were not designed to answer at all.

Díaz-Pastor and Pérez-Arriaga put their finger on this problem directly [3]. Their observation, developed in the context of electrification but equally valid for clean cooking, is that existing techno-economic models typically apply uniform or idealised financing assumptions. They use a single discount rate, or they assume that concessional

finance is available for all technologies on equal terms, or they simply take investment needs as given without asking where the money comes from and what conditions it carries. As a result, their outputs are disconnected from financial implementation. A plan that looks cost-optimal under these assumptions may look entirely different once realistic financing costs, debt service obligations, tax liabilities, and working capital constraints are introduced.

More concretely, what these models do not produce is a set of financial statements. There is no income statement showing revenues, operating costs, interest payments, and net profit year by year. There is no balance sheet showing assets, liabilities, and equity at each period. There is no cash flow statement reconciling operating income with financing needs. And without these, a planner cannot answer the most basic question that any investor or ministry of finance will ask: is this plan financially viable, and if not, how much support does it need and in what form?

This is not a minor omission. The connection matters enormously. Incorporating realistic cost-of-capital differences between technologies, for instance, can shift which solution is optimal for hundreds of millions of people in a model. An assumption that concessional finance is available for all technologies may make mini-grids appear cheaper than solar home systems, while a realistic financing structure would reverse that conclusion entirely. The technical plan and the financial plan cannot be designed independently.

2.2 Financial and regulatory frameworks for energy access

Running parallel to the techno-economic literature, and for the most part disconnected from it, there is a separate tradition of work on how energy access is financed and regulated. It asks different questions. Not which technology to deploy where, but how a utility covers its costs, how tariffs get set, and what institutional arrangements make long-term investment viable at all. The two strands have rarely spoken to each other directly, and that gap, as this section shows, matters more than it might seem.

2.2.1 Cost-of-service regulation and the regulatory asset base

Cost-of-service regulation is the core principle behind how network utilities are governed, laid out in detail by Pérez-Arriaga [4]. The idea is compact: the revenues a utility is allowed to collect from consumers are set to cover what it actually costs to deliver the service, plus a fair return on the capital invested. That return is calculated on the *Regulatory Asset Base* (RAB), which tracks the net book value of assets in service. New investment adds to it; depreciation drains it over time:

$$\mathrm{RAB}(t) = \mathrm{RAB}(t-1) + \mathrm{CAPEX}(t) - \mathrm{D\&A}(t) \tag{2.1}$$

The allowed return on that base is the weighted average cost of capital (WACC), which blends equity and after-tax debt costs in proportion to the capital structure of

the regulated entity:

$$\text{WACC} = \frac{E}{E+D}\, r_E + \frac{D}{E+D}\, r_D\,(1-\tau) \tag{2.2}$$

where E and D are the equity and debt amounts, r_E and r_D their respective costs, and τ the corporate tax rate that creates the debt tax shield.

In theory this creates a stable investment environment: the utility knows it will recover its costs and earn a reasonable return, which should make projects bankable. In practice, the framework runs into serious difficulties in developing-country contexts. Tariff increases large enough to fund rapid infrastructure expansion face political resistance and affordability constraints simultaneously. The result, as Díaz-Pastor and Pérez-Arriaga document for Uganda [3], is a chronic asynchrony: capital expenditure happens now, but cost recovery through tariffs trickles in slowly over the life of the asset. That gap has to be financed somehow, and deciding how is not a question that techno-economic models are built to answer.

The same structural problem applies to clean cooking. A utility rolling out LPG infrastructure or electric cooking connections faces large upfront costs and a population that, in the early years, does not generate enough tariff revenue to cover them. Without a financial model that captures this timing mismatch explicitly, there is no way to know how large the financing gap is, how it evolves over time, or what combination of grants, concessional debt, commercial debt and equity is needed to close it.

2.2.2 An integrated financial modelling approach

The work that comes closest to what this project does, and the one that shaped it most directly, is the integrated regulatory–financial framework developed by Díaz-Pastor and Pérez-Arriaga and applied to Uganda [3]. It is, to the best of the author's knowledge, the first serious attempt to bridge the gap between techno-economic electrification planning and rigorous financial modelling in a way that is both computationally tractable and fully replicable.

The framework structures the problem around two distinct entities. *Service-Co* is the operator of the distribution network: it delivers the physical service, collects tariff revenues, and is subject to cost-of-service regulation. Its allowed revenues are therefore tied to its RAB and its WACC; it can only collect what the regulator permits. *Finance-Co* is a dedicated financing vehicle that pools capital from multiple sources (grants, concessional loans, commercial debt and equity) and deploys it across grid and off-grid investments according to the needs of the electrification plan. The two entities are coupled: Finance-Co funds the assets that Service-Co operates, and Service-Co's regulated revenues determine what returns Finance-Co can pass back to its investors. Modelling them separately, but consistently, is what makes the framework analytically honest.

The allowed revenue that Service-Co may collect in any period is not a free parameter. It is determined by the cost-of-service calculation:

$$\text{AllowedRev}(t) = \text{WACC} \cdot \text{RAB}(t) + \text{OPEX}(t) + \text{D\&A}(t) \tag{2.3}$$

When tariff revenues fall short of this, as they typically do in developing countries in the early years of deployment, the gap must be covered by some form of external support. Making that gap explicit and quantifying it period by period is one of the framework's most important contributions.

The computational model generates complete financial statements for each actor, period by period. Income statement, balance sheet and cash flow statement are computed as a chained system, not independently. The balance sheet must balance at every period. The cash flow statement must be consistent with both the income statement and the changes in balance sheet positions. That coherence requirement is not trivial to satisfy when circular dependencies are present, and they are present whenever allowed revenues depend on the RAB and the RAB depends on investment decisions that are themselves influenced by financial projections.

The results from the Uganda application go well beyond that specific country context. The framework shows that universal access is achievable without permanent subsidies, provided the rollout is planned with a sufficiently long horizon. Extending the target year from 2030 to 2040, for instance, reduces annual investment needs by roughly two-thirds and substantially improves returns to investors. These are not intuitive findings. They only emerge when the financial model and the technical plan are genuinely integrated, and they cannot be obtained from a techno-economic model alone.

What the Uganda paper provides is therefore more than a case study. It is a proof of concept and a methodology: that it is possible to build a rigorous, transparent and replicable financial model for energy access that starts from a techno-economic plan and translates it into a complete picture of financial viability. CC-WBT is, in its core logic, an implementation of that methodology: extended to the specific context of clean cooking, made open-source, and built to scale across multiple country scenarios at once.

2.2.3 The governance dimension

There is also a governance dimension worth acknowledging. Gonzalez-Garcia, Diaz-Pastor, and Moreno-Romero bring it into focus: their work on universal energy access governance, grounded in more than a decade of field research across 25 countries, argues that the missing ingredient in most electrification efforts is not technical expertise but the alignment of a complex ecosystem of actors: governments, utilities, development finance institutions, NGOs, academic institutions, and the communities themselves [8].

Two points from that work bear directly on what CC-WBT is trying to do. When plans cannot be translated into financial projections that investors and governments can actually evaluate, they remain documents rather than programmes. The translation rate from national plan to physical infrastructure is, as the evidence shows, disappointingly low. Beyond that, tools need to reach beyond specialist modellers. Transparency and auditability matter as much as technical sophistication, and sometimes more.

Both of these points shaped the design of CC-WBT. The platform was built to produce financial outputs that ministries and development banks can interrogate, not just outputs that the person who built the model can interpret. And it was built on a declarative, open configuration structure precisely so that the financial logic can be

read and verified by anyone with relevant technical background, without needing to dig into the underlying engine.

2.3 Existing financial tools for clean cooking

After all that background, it would be reasonable to expect financial modelling tools for clean cooking to be well established by now. They are not. Most of the financial planning done in this space has relied on bespoke Excel models built for individual country studies, and the clean cooking sector has not had an integrated, open-source financial platform until now.

2.3.1 The Rwanda Financial Planner

The most directly relevant existing tool is the financial model built as part of the Rwanda project that served as the basis for the National Integrated Clean Cooking Plan, documented in the de Cuadra et al. working paper [1]. The Financial Planner, as it is described there, was an Excel-based model designed to take the outputs of the techno-economic planning tool and translate them into a financial picture.

It took the CAPEX and OPEX projections from the techno-economic analysis and combined them with macroeconomic inputs, tariff assumptions, regulatory parameters, and financing options across several interconnected modules. It handled multiple fuel markets separately, recognising that e-cooking and LPG have fundamentally different cost structures, infrastructure requirements, and revenue dynamics. It incorporated carbon credits as an additional revenue stream, with parameters for certification rates, liquidity, and price per tonne of CO2 avoided. And it produced financial statements, a uses-and-sources-of-funds table, and graphical outputs that allowed planners to compare alternative deployment strategies.

For the Rwanda exercise, this was a genuine contribution. It made visible things that the techno-economic model could not show: the viability gap (the difference between what tariffs can cover and what the plan costs), the structure and timing of financing needs, the role of different instruments, and the sensitivity of financial outcomes to changes in tariff levels, adoption rates, and financing terms. The Rwanda National Integrated Clean Cooking Plan was, in part, grounded in the outputs of this model.

2.3.2 Limitations of the Excel-based approach

Despite what it achieved, the Financial Planner illustrated exactly the kind of problems that motivate this project. The model was built in Excel, which means its financial logic was encoded in cell formulas spread across multiple spreadsheets. It worked, but it was hard to audit. Following the logic of a calculation that spans dozens of sheets, with intermediate results stored in named ranges and formulas referencing cells across tabs, requires significant effort even from someone who has built a similar model before. For a new team picking it up for the first time, the learning curve was steep and the risk of misinterpreting or inadvertently breaking a formula was real.

The model also had limited reproducibility. Running it for a new country meant starting, in practice, from scratch. The structure could be reused as a template, but adapting the formulas, the sheet layout, the parameter ranges, and the input connections to a different country's data required substantial manual work. There was no systematic way to package a scenario, share it, and have another team reproduce exactly the same results. The assumptions embedded in the model were not always visible without opening and inspecting the underlying cells.

The de Cuadra et al. working paper is candid about these limitations [1]. The model depended on accurate and timely input data; any discrepancy in updating those data would propagate through the projections in ways that were difficult to trace. The rapid evolution of technologies and markets could render embedded assumptions obsolete without any clear mechanism for updating them systematically. And the distribution of the model, or collaboration around it, was constrained by the proprietary nature of the Excel format and the difficulty of version control.

There is also a technical issue that Excel-based approaches handle awkwardly. The three financial statements are not independent. Net income feeds retained earnings, which affect equity on the balance sheet. Debt levels affect interest expense on the income statement. The regulatory asset base, which drives allowed revenues, depends on accumulated investment net of depreciation, which feeds back into tariff calculations and therefore into revenues. These circular dependencies can be handled in Excel using iterative calculation settings, but doing so reliably, across a model of this complexity, requires care and is not transparent to users who are not deeply familiar with how Excel resolves circular references. The results depend on convergence settings that are rarely documented.

Beyond Rwanda, the broader landscape of clean cooking financial analysis confirms the pattern. Country-specific Excel models have been built for other national programmes, often supported by development finance institutions or bilateral donors. Each one represents a substantial investment of time and expertise. Each one is largely non-transferable. The field has not converged on a shared, open methodology that any team can pick up and apply.

2.4 Summary: the gap this project addresses

The picture that emerges from this chapter is fairly clear.

On the techno-economic side, the field has genuinely sophisticated tools. REM for electrification and the ICCP for clean cooking can handle national-scale problems with high spatial granularity, realistic adoption modelling, and rich output sets. What they produce, though, stops at the techno-economic boundary. They do not generate financial statements, they do not model the interaction between tariffs and cost recovery, and they do not tell a planner whether a given deployment strategy is actually financeable.

The methodology to go further does exist. The work of Díaz-Pastor and Pérez-Arriaga shows that it is possible to build a computationally coherent financial model integrating cost-of-service regulation, blended capital structures, and the three interconnected financial statements, revealing insights simply inaccessible to techno-economic

models alone. But this methodology has been applied primarily to electrification, and the implementation has not been released as an open, reusable tool.

In terms of actual tools for clean cooking specifically, the Rwanda Financial Planner is the only direct antecedent at national scale. It demonstrated that the problem is tractable and that the outputs are meaningful for planning. It also demonstrated, in practice, everything that comes with an Excel-based approach: opacity, limited reproducibility, and the difficulty of picking the model up and applying it somewhere new.

The gap is real and it has consequences. Organisations working on clean cooking (ministries of energy, multilateral development banks, NGOs, energy access consultancies) currently have no shared, open tool they can reach for when they need to answer the question of financial viability at country scale. Every team builds something from scratch. Most of what they build cannot be shared, audited or adapted without a substantial reinvestment of time and expertise. The field ends up rebuilding the same model repeatedly, in slightly different forms, for slightly different countries, with no accumulated methodological capital to show for it.

CC-WBT was built to change that. It takes techno-economic outputs as its starting point, exactly as the Rwanda Financial Planner did. It computes a full set of integrated financial statements, exactly as the Díaz-Pastor framework does for electrification. But it does so as a robust, scalable, open-source platform: financial formulas in declarative configuration files that any technically informed user can read, check and modify; an interactive web interface that any planner can use without developer support; a computation engine that handles multiple country scenarios, fuel markets and techno-economic models simultaneously; and a scenario packaging mechanism that makes every result fully reproducible and transferable across teams and organisations.

For a ministry preparing a national clean cooking plan, CC-WBT provides the financial layer that techno-economic tools leave out. For a development bank evaluating whether to commit financing, it provides the auditable, period-by-period financial picture they need before committing capital. For a research team studying policy alternatives, it provides a transparent and extensible platform they can adapt without starting from zero. The following chapters describe exactly how it was built and how it performs.

Chapter 3

Solution Design

This chapter is about the thinking behind CC-WBT: what problem it had to solve, what constraints that imposed, and what architectural decisions followed from those constraints. The implementation — the computation engine, the iterative solver, the visualisation layer — is covered in Chapter 4. This chapter stays at the level of design.

3.1 Problem Statement

Clean cooking is still one of the big unsolved problems in global energy. More than two billion people worldwide cook every day with traditional biomass, with direct costs on their health, on the unpaid care time spent gathering fuel and cooking, and on carbon emissions. Moving a whole country away from that, towards electricity, LPG, natural gas or biofuels, is a very different financial problem from designing a single clean-cooking project. At national scale, planners have to juggle several fuel markets at once, compare alternative deployment strategies side by side, and keep track of the regulatory machinery that shapes how the service is priced. This includes the regulatory asset base, the annual cost of service, and the long-term subsidies that may be needed to close the gap. On top of that, the whole thing has to be funded through a mix of debt, equity, grants and potentially carbon credits, all while staying solvent year after year.

The tools available when this project started did not cover that combination well. The typical approach is a purpose-built Excel model, assembled from scratch for a specific study, useful for that single case but brittle, hard to audit and difficult to extend to a different country or scenario. The energy-planning platforms already on the market, discussed in detail in Chapter 2, focus on the techno-economic side (demand, coverage, infrastructure sizing), but rarely project the three full financial statements, and almost never capture the regulatory dynamics of cost of service or the long-term viability gap. As a result, the people who actually have to plan these transitions (ministries, regulators, utilities, development agencies) are left without an integrated, transparent and reproducible tool to answer the basic question of how much it costs to deploy a given strategy, how much of that cost is recovered through tariffs, and how much public or external financing is needed to close the gap.

Answering that question reliably shaped the requirements behind the platform. Multi-scenario scalability was non-negotiable: the tool has to let a planner define and compare several country scenarios at once, each with multiple techno-economic models and fuel markets, running independent and coherent financial computations for every combination. Financial consistency followed immediately: the three statements —

profit and loss, balance sheet, cash flow — have to be computed as a chained system, not independently, so that what flows out of one feeds correctly into the next and the balance sheet holds every period.

Traceability mattered too, and by design rather than as a side effect. Every financial formula needed to be readable, checkable and editable directly in configuration files, without any need to touch the underlying engine — as long as the operation it relies on is already in the engine's catalogue. That keeps the business logic visible to an auditor and accessible to a domain expert at the same time. The target audience, after all, is not developers but planners: economists, engineers, consultants with the domain knowledge to interpret the numbers, who need a guided interface but cannot be expected to dig into Python to adjust a formula.

Extensibility and reproducibility rounded out the set. Adding a new fuel market, country or scenario should never require a developer. And every scenario should be exportable as a self-contained bundle — inputs, configuration and assumptions packed together — so that any team can open it later and reproduce the results exactly.

All these requirements shape the design that comes next, in Section 3.2. To make sure the platform actually delivers on them, Chapter 5 puts it through three checks: a head-to-head comparison against the Rwanda reference Excel model, a balance-sheet consistency audit (Assets = Liabilities + Equity, every period), and a convergence test on the iterative solver described in Chapter 4.

3.2 Solution Design

The design that emerged rests on four building blocks: a split architecture that separates what the user sees from what does the computation; a hierarchical data structure that lets many scenarios live side by side without interfering with each other; a financial model that extends the standard project-finance template with the regulatory layer that clean-cooking deployments actually need; and a set of deliberate modelling choices, discussed separately in Section 3.2.4, that reflect how the platform handles edge cases and structural ambiguity.

3.2.1 System architecture

The whole platform is written in Python, and rests on two complementary frameworks. Streamlit handles the frontend and FastAPI handles the backend. Keeping them separate is a deliberate choice for scalability and robustness. Streamlit on its own simply does not have the computational backbone needed to handle heavy financial simulations at scale. Moving all the computation and data management to a dedicated backend means the system can grow in complexity over time without slowing down the interface that the user actually sees.

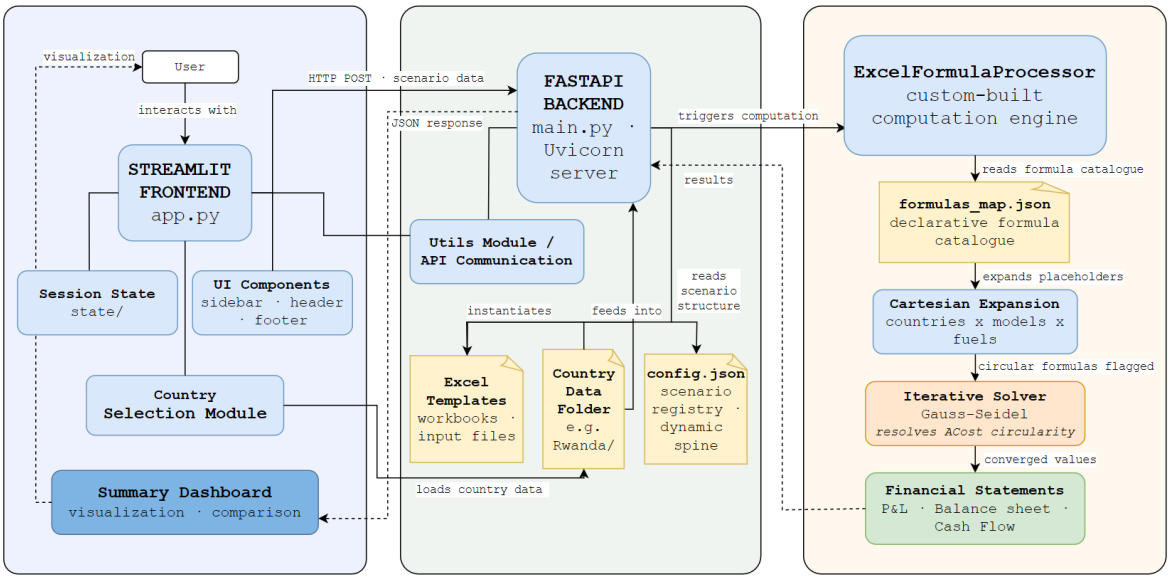


Figure 3.1: Information flow through the three layers of CC-WBT: frontend, backend computation engine, and financial model.

3.2.1.1 Frontend: Streamlit

Streamlit was chosen because it builds interactive, data-oriented web interfaces straight from Python scripts, with no web-development stack to worry about. The interface revolves around a persistent sidebar that adapts to what the user is doing. Before any model has been loaded, the sidebar only shows the sections that apply to the whole country scenario: *Country Selection*, *Financial Inputs* (one block per fuel market), *Manage Techno-Economic Models*, and an *Outputs* block with *Carbon Credits* and *Summary Financing*. The moment a model is selected, a second layer unfolds: a model-specific *Inputs* block (*Techno-Economic Inputs*, *Tariffs*, *Upstream Costs of Energy*, *CAPEX Fuel Market*), a *Design Capital Structure* block with the *Financial Plan*, and the *Financial Statements* that consolidate the results for that model. The *Summary Financing* view at the bottom acts as a cross-model dashboard, letting the user compare the key financial indicators of all loaded scenarios side by side. Navigation itself is state-aware: a country scenario and at least one techno-economic model have to exist before the downstream sections become accessible, which keeps the data flow coherent and guided.

3.2.1.2 Backend: FastAPI

All the computational logic, data persistence and file management is handled by a FastAPI server running locally on top of Uvicorn. Every user action that touches data triggers its own dedicated endpoint. This endpoint-oriented architecture is what gives the platform the power to simulate complex financial scenarios across many deployment strategies at once without slowing down the interface, and lays a clean foundation for future extensions.

3.2.1.3 Iterative development and open-source release

From day one the platform has been developed locally and published in a public GitHub repository. Development followed an iterative path. The first version supported a single country scenario at a time. Multi-country scalability was added later, in response to feedback from the partner organisation, which needed the platform applicable to several country contexts within the same instance. The final version generalises that approach to multiple country scenarios running at the same time, each with its own independent set of financial computations, including the three full financial statements.

3.2.2 Data structure and input management

One of the central design goals was to give the user a structured and scalable way of managing inputs across many scenarios and fuel markets at once, without losing track of which input belongs where. That required a careful separation at the data level between the parameters that apply to the whole country scenario and those that are specific to each techno-economic model. Country-level parameters need to stay shared across every model defined inside the scenario: the time horizon, the tax rate, the inflation rate and the financial parameters of each fuel market. Model-level parameters, on the other hand, are exactly where the user varies assumptions to explore different deployment strategies.

3.2.2.1 Input architecture

User inputs are organised on two levels. The first holds the inputs that apply to the whole country scenario. *Financial Inputs* covers the commercial and regulatory parameters of each fuel market: expected non-technical losses, trade receivables (days of revenues), and trade payables (days of OPEX costs). *Carbon Credits Financial Inputs* defines the assumptions on emissions-based revenues: the share of avoided CO_2 certifiable as credits, market liquidity, price per tonne, and years over which credits can be sold. The scenario time horizon is set the very first time the user enters *Manage Techno-Economic Models*: start year and end year must be defined before any model can be created, and that year range is then propagated automatically to every model and every backend Excel template.

The second level is model-specific. For each techno-economic model, the user fills in *Techno-Economic Inputs* (annual demand, CAPEX, OPEX, depreciation schedules), *Tariffs*, *Upstream Costs of Energy*, *CAPEX Fuel Market*, and the *Financial Plan* under *Design Capital Structure* (the equity/grants/debt split, cost of capital, grace and amortisation periods).

3.2.2.2 Declarative parameters in configuration

Not every input lives inside an Excel file. A large share of the platform's state is held in a single declarative file, *config.json*, shared across the whole system.

Structural parameters live there: the year range, the tax rate, the inflation rate, tariff curves per fuel market, and upstream energy costs. These are stable values that do

not benefit from Excel's tabular interface and are easier to inspect and version-control in one centralised place.

The dynamic structure of the platform also lives there. *config.json* holds a registry of which countries exist, which techno-economic models live inside each country, which fuel markets are active, and how those fuel markets are organised into template variants (*normal*, *expanded*, *with carbon credits*). It also holds the catalogue of Excel files managed for every model, encoded as templated routes such as *technoeconomic-inputs-{model}.xlsx*, where the *{model}* placeholder is expanded automatically when a new model is defined.

In other words, *config.json* is not a passive settings file. It behaves as a dynamic spine that the platform reads on every action to know what scenarios exist, what models live inside them, what fuel markets they include, and which Excel pages should be generated. Whenever the user edits a tariff or an upstream cost through the UI, the platform writes the configuration file, and every downstream simulation picks up the new value on the next recalculation.

3.2.2.3 Dynamic templates and file management

The rest of the input data is stored in structured Excel files instantiated from predefined blank templates. When a new model is created, all its Excel files are generated automatically. The platform fills the *{model}* placeholder with the new model's name and drops a fresh copy of every relevant template into the country's folder, ready to be filled in.

To reduce manual effort, the platform also supports a download-upload flow. The user can download a single consolidated Excel template, fill it in offline, and upload it back. The upload stages the file inside the country's folder and drops an *upload flag* next to every dependent section file. The flag tells the engine that on its next run it should read inputs from the uploaded file. This decouples the upload from the computation, avoids race conditions, and lets the platform commit the bulk update atomically when the next recalculation runs.

Every Excel file associated with a model or country can also be exported as a clean ZIP archive containing only the files meaningful at the user's level, which makes it straightforward to move scenarios between machines, share them, or archive for reproducibility.

3.2.2.4 Extensible list of fuel markets

The platform treats the set of fuel markets as something that can grow and shrink over time, beyond the two defaults of *Electricity* and *LPG*, with *Carbon Credits* available as an optional add-on. When a new market is added, the platform detects the missing sheets across all workbooks and creates them automatically from the corresponding template.

Electricity is the one explicit exception to the otherwise symmetric treatment of fuels. Access to electricity is not binary: households fully connected to the grid, households with limited access, and households adopting e-cooking on top of existing electricity represent three distinct demand profiles, cost structures and financing needs.

The platform therefore splits electricity internally into several access-level variants. This asymmetry runs through almost every generic mechanism of the platform, and its implications are discussed in more detail in Section 3.2.4.

3.2.3 Financial model

The financial model behind the platform is designed to reflect what it actually takes to deploy clean cooking at country scale. It goes beyond the conventional project-finance template and adds the regulatory layer that shapes how revenues, costs and subsidies are accounted for in a public-utility setting. The result is a model that combines the rigour of standard corporate finance, a full set of interlocking financial statements, a clean balance-sheet identity, an explicit treatment of cash flows, with the regulatory accounting logic that planners and regulators use to decide how much of the cost of service is recovered through tariffs and how much has to be covered by long-term subsidies.

3.2.3.1 Scope and structure

Rather than evaluating one isolated project, the model computes financial results separately for each fuel market and each techno-economic scenario, projected over the full planning horizon. This is what lets planners compare the financial sustainability of alternative deployment strategies under a single, consistent set of assumptions, and simulate the financial impact of policy changes such as tariff adjustments, subsidy schedules or changes in the capital structure.

For every (market, scenario) pair, the model produces the three classic financial statements: a profit and loss statement (P&L), a balance sheet (BS) and a cash flow statement (CFS). Around them, the same workbook computes supporting schedules: the property, plant and equipment schedule tracking CAPEX additions and depreciation; the working capital details; the equity schedule; the capital structure tables for grants and debt; and the Annual Cost of Service block, the regulatory layer.

None of these tables are computed in isolation. The outputs of one block feed directly into the others, which is exactly what keeps the three statements internally consistent and what made the underlying engine non-trivial to build.

3.2.3.2 Regulatory asset base and annual cost of service

At the heart of the model sits a regulatory accounting framework built around two key concepts from how regulated utilities are priced.

The first is the *Regulatory Asset Base* (RAB), which tracks the net value of capital assets in service in a given period. It evolves through two opposing forces: new investment adds to it through CAPEX, and depreciation slowly drains its book value. Formally, its evolution is given by

$$\mathrm{RAB}(t) = \mathrm{RAB}(t-1) + \mathrm{CAPEX}(t) - \mathrm{D\&A}(t) \tag{3.1}$$

where $\mathrm{CAPEX}(t)$ captures new investment entering service during period t and $\mathrm{D\&A}(t)$ the depreciation and amortisation that drains book value from already-installed assets.

The second concept is the *Annual Cost of Service* (ACoSt), the model's representation of what it actually costs, in any given year, to deliver the energy service to its end users. ACoSt is built up from: the return on the asset base, weighted by the cost of capital that funded it; the operating expenses of running the service; depreciation; the variation in working capital; upstream energy costs; and taxes. Putting all those components together,

$$\mathrm{ACoSt}(t) = \mathrm{WACC} \cdot \mathrm{RAB}(t) + \mathrm{OPEX}(t) + \mathrm{D\&A}(t) + \Delta\mathrm{WC}(t) + \mathrm{Upstream}(t) + \mathrm{Taxes}(t) \tag{3.2}$$

where WACC is the weighted average cost of capital and $\Delta\mathrm{WC}(t)$ is the variation in working capital between periods $t-1$ and t.

This last ingredient is what makes ACoSt structurally difficult to compute in closed form. Taxes depend on the operating result, the operating result depends on the long-term subsidies needed to close the gap between revenues and costs, and those subsidies are defined as the amount required to cover the very ACoSt being computed. This circular dependency is resolved numerically by the iterative solver described in Chapter 4.

Taken together, RAB and ACoSt give planners a rigorous, auditable measure of what it really costs to deliver the service, period by period: exactly the figure that simpler planning tools leave implicit and that this model surfaces explicitly.

3.2.3.3 Long-term subsidies and the viability gap

A structural feature of clean-cooking markets is that tariff revenues typically do not cover the full cost of service, especially in the early years of deployment. The model makes this support explicit. For every period of the planning horizon, the long-term subsidy is defined as

$$\mathrm{LTS}(t) = \mathrm{ACoSt}(t) - \mathrm{TariffRev}(t) \tag{3.3}$$

where $\mathrm{TariffRev}(t)$ are the revenues recoverable through tariffs in period t. This figure is an *output* of the model, not an assumption fed in by hand. Planners get, period by period, a transparent measure of the public-financing requirement tied to each scenario, which is one of the most useful outputs of the whole analysis.

3.2.3.4 Debt dynamics and solvency indicator

The model lays out how debt builds up and gets repaid over time through a dedicated schedule that distinguishes between grace periods (interest only) and amortisation periods (principal repayments start). The weighted average cost of capital (WACC) that feeds back into the regulated return on the RAB is defined as

$$\mathrm{WACC} = \frac{E}{E+D}\, r_E + \frac{D}{E+D}\, r_D\,(1-\tau) \tag{3.4}$$

where E and D are the equity and debt amounts, r_E and r_D their respective costs of capital, and τ the corporate tax rate that creates the tax shield on debt interest.

A dedicated solvency indicator inspects the closing cash balance of every period and flags any one in which it goes negative, giving the planner an immediate signal

that the current financing structure is insufficient and that additional external support is needed.

3.2.4 Design decisions

Some of the choices made during development deserve a closer look — not because they are unusual, but because they reflect deliberate trade-offs that shape the whole model.

3.2.4.1 Electricity access levels

Access to electricity is not binary. The platform therefore subdivides electricity into *Electricity & E-Cooking* (full access plus electric cooking), *Electricity (Low Access)* (basic services only), and a derived analytical category *Electricity (Only E-Cooking)*, computed as the difference between the two. This decomposition makes visible the marginal cost and financing requirement of the clean-cooking transition over and above basic electrification, an angle that would be lost in the aggregate electricity figure.

3.2.4.2 On-grid vs off-grid split

Inside *Techno-Economic Inputs* and *CAPEX Fuel Market*, electricity is further split between on-grid and off-grid systems. Grid-connected systems have centralised infrastructure and regulated tariffs; off-grid systems require distributed generation, storage and per-household equipment, translating into substantially higher per-user CAPEX. Averaging the two regimes would misrepresent the investment needs of off-grid deployment and distort the economics of grid expansion.

3.2.4.3 Carbon credits as a separate financing stream

Carbon credits are kept on their own track rather than merged into general tariff revenues, for both mechanical and substantive reasons. Mechanically, they are computed from a completely different parameter set: certifiable fraction, market liquidity, price per tonne, and years over which credits can be sold. Substantively, keeping the stream separate lets the user see exactly how much of the financial picture is leaning on carbon revenues and gauge its impact on overall viability without having to untangle it from the rest.

3.2.4.4 Multi-scenario architecture and reference scenario

The platform is designed to define and compare several techno-economic scenarios within the same country simultaneously. One rule is fixed by the platform itself: a Baseline scenario must be created first, and the name *Baseline* is reserved exclusively for that anchor role. The Baseline carries the planning horizon and the reference trajectory that all other scenarios are read against. It is the country's counterfactual, the path expected if no deliberate clean-cooking effort took place, and the emissions benchmark against which carbon credits and the relative financial gains of every alternative scenario are computed.

Chapter 4

Platform Implementation

This chapter describes how CC-WBT is implemented across three layers that work together to make the platform function as a whole.

The interface the user sees is a Streamlit frontend: a guided, state-aware web application that collects inputs through validated editable tables and renders results through a structured comparison dashboard. All computation is handled separately by a FastAPI backend running locally. The centrepiece of that backend is the *ExcelFormulaProcessor*, a custom-built engine that represents most of the implementation effort in this project. It reads every financial formula from a declarative configuration file, expands them automatically across the full space of active country scenarios, models and fuel markets, resolves the dependency graph, and applies each formula to live Excel workbooks — effectively building a controlled, auditable and fully automated spreadsheet engine in Python.

The third layer is the financial model itself. This is where the platform answers the hard questions: how revenues, costs, subsidies and capital returns are computed year by year, how the balance sheet stays consistent across periods, and how the circular dependency at the core of the cost-of-service calculation is resolved iteratively. The logic of that third layer is what the sections below describe.

4.1 Editable tables with input validation

Even though the platform is designed for an informed user, economists, engineers, planners with the domain knowledge to interpret the numbers it produces, usability was treated from the start as a first-order concern. A single typo in the wrong cell can quietly contaminate every downstream simulation. The platform therefore catches input mistakes the moment they happen rather than later, when they would already have crept into the financial results.

All input tables are rendered as editable grids using *streamlit-aggrid*. Editability is controlled at two levels. The first is column-based: the columns holding input names, units and section headers are locked programmatically, and only the year columns the table is meant to receive data in remain editable. The second is row-based, configured per section through a dedicated parameter: each section module declares which rows are not supposed to be filled in by the user (outputs computed elsewhere, or rows left for the engine to populate), and the editor blanks those rows on load and prevents them from being edited.

Within the editable cells, a dedicated validation layer enforces what each cell is supposed to contain. The platform reads the unit declared for the row and uses it

to choose the right validator: percentages must lie between 0 and 100, day and year fields must be positive integers, monetary fields must be positive numbers, and any cell supposed to be numeric refuses a non-numeric entry. When a value fails its check, the platform disables the *Save* button so that the invalid table cannot be persisted, and surfaces a row-by-row warning pointing to the exact cell at fault, restating the rule broken and showing the value the user entered. The messages are deliberately concrete: *Value must be between 0 and 100*, *Value must be a positive integer*, *Value must be positive*, *Not a valid number*, each accompanied by the row name, the year column, and the invalid value.

This validation layer reflects a broader design principle: keep the interface intuitive enough that an informed user can move through it quickly, but be strict about input errors that would otherwise silently propagate through the chained financial statements.

4.2 Dynamic computation engine

The computation engine is where most of the implementation effort went. Its job is to take the financial formulas expressed in a compact, declarative form and simulate the full set of financial outcomes for every active combination of country, scenario and fuel market, automatically and without any developer intervention. The engine is implemented as the *ExcelFormulaProcessor* class and is driven by a single configuration file, *formulas_map.json*.

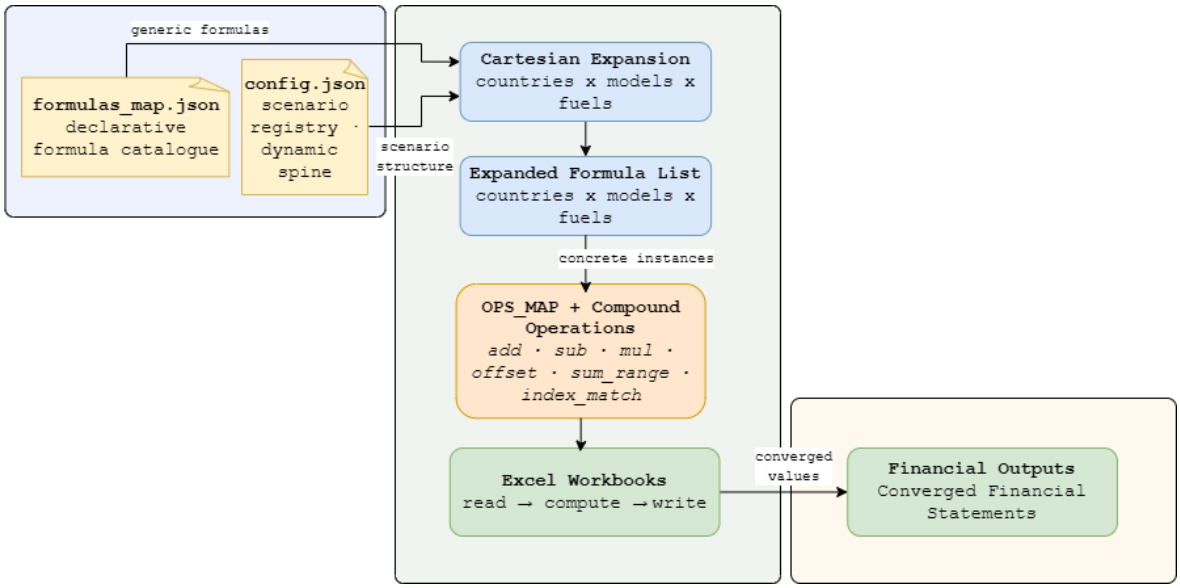


Figure 4.1: Processing pipeline of the *ExcelFormulaProcessor*: from declarative formula definitions to concrete financial outputs.

4.2.1 Declarative formula catalogue: `formulas_map.json`

Every financial computation in the platform is written in *formulas_map.json* in a generic, scenario-agnostic form. For each Excel file and each sheet inside it, the file lists the sequence of operations to apply and the source and target cell references those operations should act on. Those references are not written for any specific country or model. They use named placeholders: *{country}*, *{model}*, *{fuel}* and *{sheet}*, that have no fixed value at the level of the formula definition itself.

This declarative approach achieves a clean separation between what the model computes and how it is executed. Adding a new formula, adjusting a financial relationship, or modifying how a tax rate is applied requires only an edit to *formulas_map.json*: the engine picks up the change automatically on the next run, with no need to touch a single line of Python. This is the mechanism that makes CC-WBT auditable and extensible at the same time.

A concrete example makes the structure easier to read. The entry below defines the RAB update rule generically, without reference to any specific country or model:

```
{
  "target": "RAB",
  "sources": [
    "{country}/capex-fuels-{model}.xlsx::{sheet}::RAB",
    "{country}/capex-fuels-{model}.xlsx::{sheet}::Total CAPEX",
    "{country}/capex-fuels-{model}.xlsx::{sheet}::Total Annual
       Depreciation"
  ],
  "formula_steps": [
    { "op": "offset",
      "operands": [["index", 0], ["literal", 1]],
      "result": "temp1" },
    { "op": "addition",
      "operands": [["index", "temp1"], ["index", 1]],
      "result": "temp2" },
    { "op": "subtraction",
      "operands": [["index", "temp2"], ["index", 2]],
      "result": "final" }
  ]
}
```

The three entries in `sources` list the cell references the formula reads from, indexed 0, 1 and 2. The `formula_steps` then operate on those indices: `offset` retrieves RAB from the previous period, `addition` adds the current CAPEX, and `subtraction` removes depreciation. The placeholders `{country}`, `{model}` and `{sheet}` are substituted at run time by the Cartesian expansion, turning this single entry into as many concrete formulas as the scenario requires.

4.2.2 Cartesian expansion over the scenario space

The key step that turns the compact formula catalogue into a concrete computation plan is the Cartesian expansion. Whenever the engine needs to apply formulas to a given Excel file, it first consults *config.json* to retrieve the full structure of the active scenario: which countries $\mathcal{C}$ exist, which techno-economic models $\mathcal{M}(c)$ live inside each country c, and which fuel markets $\mathcal{L}(c, m)$ are active in each model m. With that information in hand, it walks through every generic formula entry in *formulas_map.json* and instantiates one concrete formula for each element of the set

$$\mathcal{F}_{\text{expanded}} = \Big\{ f(c,\, m,\, \ell) \;\Big|\; c \in \mathcal{C},\; m \in \mathcal{M}(c),\; \ell \in \mathcal{L}(c, m) \Big\} \tag{4.1}$$

where $f(c, m, \ell)$ is the result of substituting the placeholders $\{country\} \to c$, $\{model\} \to m$ and $\{fuel\} \to \ell$ into the generic formula entry f.

The practical consequence is significant. A single formula entry in the configuration file, written once, expands at run time into as many concrete formulas as there are (country, model, fuel) triples active in the scenario. A deployment covering three country scenarios, each with four techno-economic models and five fuel markets, would expand a single generic formula into $3 \times 4 \times 5 = 60$ concrete instances, all optimised and executed automatically. Every financial number the user sees on screen is the output of exactly one of these expanded instances.

4.2.3 Operation language and the OPS_MAP

Each formula is expressed as an ordered sequence of elementary steps. Every step takes operands (cell references or values pulled from *config.json*), applies an operation, and writes the result either into an intermediate slot or, as the final step, into the target cell.

Most steps follow a column-by-column rhythm: for every year column the formula has to fill in, the engine reads the same column in each source cell, applies the operation, and writes the result into the same column of the target row. Operations of this kind are implemented as compact lambdas and collected in a flat dictionary called *OPS_MAP*. Between them, *OPS_MAP* covers most of the day-to-day arithmetic and logic the financial model relies on: addition, subtraction, multiplication, division (with explicit protection against divide-by-zero through *safe_divide*), comparisons (greater-than, less-than, equal), the conditional *if*, plain *copy*, absolute value, minimum, maximum, conversion between fractions and percentages, and integer truncation.

Some formulas do not fit into the same-year discipline: a working capital figure may need the previous year's trade receivables; a depreciation schedule may need to accumulate CAPEX from every year up to the current one; a debt schedule may need to look up a value in a labelled table. These cases are handled through a second layer of compound operations implemented as dedicated methods: *get_cell* reads a fully-resolved reference; *offset* picks the value a given number of positions to the left or right (the engine's analogue of *the previous year*); *sum_range* adds up a contiguous sequence of cells (the analogue of Excel's *SUM* over a range); *value* injects a constant or a configuration-defined number; and *index_match* implements the equivalent of

Excel's *INDEX/MATCH* lookup, navigating across rows and columns to find a match in a labelled table.

The set of operations is also open to extension: adding a new primitive requires nothing more than registering a lambda in *OPS_MAP* or implementing a dedicated method, at which point it becomes immediately available to any formula in the catalogue. In practice, the current catalogue already covers the full range of mathematical operations the financial model requires — from year-by-year arithmetic and conditional logic to cross-period accumulations and labelled lookups — with no gaps that would force a workaround outside the engine.

4.2.4 Dynamic sheet and year-horizon management

The engine operates on workbooks whose structure — number of sheets and range of year columns — is not fixed at build time but determined by the active scenario. Rather than maintaining a rigid template for every possible configuration, the platform adapts each workbook to match the current scenario before any computation runs.

Two synchronisation routines handle this. The first, *sync sheets with fuels*, aligns the sheet topology with the fuel markets declared in *config.json*: sheets for fuel markets that have been added are created from the corresponding blank template, and sheets for markets that have since been removed are pruned. This means that extending the platform to a new fuel market never requires manual spreadsheet surgery — the routine handles the structural change automatically on the next pass.

The second routine manages the year horizon. When the planning range changes — because the user has extended or shortened the scenario — the routine prunes year columns that fall outside the active range and inserts the missing ones at the right position. The behaviour differs depending on the role of the sheet: in input tables, new year columns are added empty, ready for the planner to fill in; in computed sheets, the engine fills them in immediately as part of the same pass. In both cases, the workbook always presents exactly the years the scenario requires, no more and no less, without any manual intervention.

4.3 Resolving the circularity of the financial model

One of the structural challenges of the financial model is a circular dependency that sits right at the heart of the cost-of-service calculation. It is not a modelling mistake or an edge case — it is built into the economics of a regulated utility. To know the Annual Cost of Service, you need taxes. To know taxes, you need the taxable result. To know the taxable result, you need total revenues, which include the long-term subsidy. And the long-term subsidy is defined as whatever it takes to cover the Annual Cost of Service. The loop closes on itself:

$$ACoSt \rightarrow \textit{Long-term subsidies} \rightarrow \textit{Total revenues} \rightarrow \textit{Taxes} \rightarrow ACoSt$$

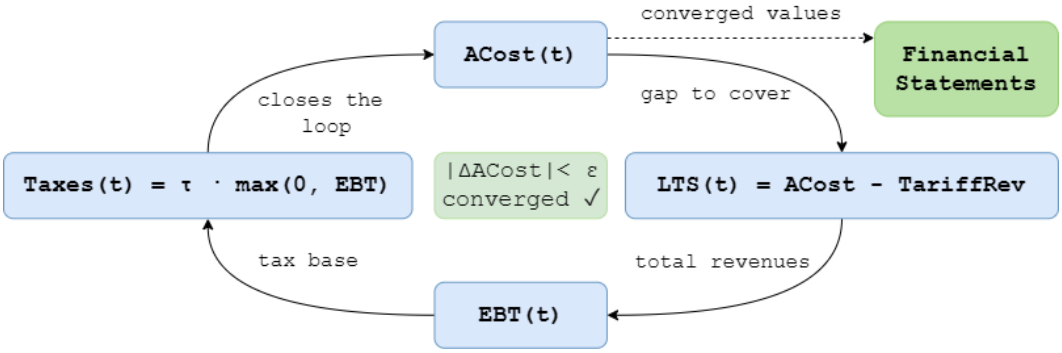


Figure 4.2: The four-step circular dependency in the cost-of-service calculation and its resolution via the Gauss-Seidel iterative solver.

There is no calculation order that breaks the cycle cleanly. Picking any one of the four quantities as a starting point and treating the others as known produces a model that is internally inconsistent — the balance sheet closes only by coincidence, and the cost of service it reports is not the cost the underlying numbers actually imply. The platform resolves the loop properly, through an iterative solver described below. Figure 4.2 illustrates the dependency loop and the three-phase resolution strategy.

4.3.1 Iterative resolution: Gauss-Seidel scheme

The solution is to iterate. Start with an initial guess, update each of the four quantities in turn using the most recent values of the others, and repeat until nothing moves. This is the classical Gauss-Seidel idea, applied here directly inside the declarative engine without any separate solver, without symbolic algebra, and without having to invert a system of equations by hand.

To make this work cleanly within the existing architecture, every formula in *formulas_map.json* carries a tag that tells the engine which phase it belongs to. Formulas tagged *circular* — ACoSt, LTS, total revenues, EBITDA, EBIT, EBT and taxes — are the ones inside the loop and will be iterated. Formulas tagged *post_circular* are those that derive from converged values and must wait until the loop has fully settled before they run: growth rates, revenue ratios and similar derived quantities fall into this group. Everything else is untagged and runs exactly once, before the iteration begins, establishing the baseline values that the loop itself depends on.

Each computation pass therefore unfolds in three phases. First, all untagged formulas execute once, laying down the quantities that do not participate in the circularity at all. Second, the engine enters the iterative loop over the circular formulas, updating all four quantities in sequence at every iteration until the solution converges. Third, once convergence is confirmed, the post-circular formulas execute once on the settled values. Splitting the pass this way is what guarantees that derived quantities like growth rates are never computed on an intermediate, still-changing value from inside the loop — they always see a financially coherent, fully converged picture.

Formally, at each iteration k and for each period t, the four quantities update in

sequence:

$$\mathrm{LTS}^{(k+1)}(t) = \mathrm{ACoSt}^{(k)}(t) - \mathrm{TariffRev}(t) \tag{4.2}$$

$$\mathrm{EBT}^{(k+1)}(t) = \mathrm{TariffRev}(t) + \mathrm{LTS}^{(k+1)}(t) - \mathrm{OPEX}(t) - \mathrm{D\&A}(t) \tag{4.3}$$

$$\mathrm{Taxes}^{(k+1)}(t) = \tau \cdot \max\Big(0,\, \mathrm{EBT}^{(k+1)}(t)\Big) \tag{4.4}$$

$$\begin{aligned}\mathrm{ACoSt}^{(k+1)}(t) = {} & \mathrm{WACC} \cdot \mathrm{RAB}(t) + \mathrm{OPEX}(t) + \mathrm{D\&A}(t) \\ & + \Delta\mathrm{WC}(t) + \mathrm{Upstream}(t) + \mathrm{Taxes}^{(k+1)}(t)\end{aligned} \tag{4.5}$$

The loop stops when the largest change in ACoSt across all periods drops below a tolerance of 10^{-10}:

$$\max_t \left|\mathrm{ACoSt}^{(k+1)}(t) - \mathrm{ACoSt}^{(k)}(t)\right| < \varepsilon, \qquad \varepsilon = 10^{-10} \tag{4.6}$$

or after at most $k_{\max} = 10$ iterations if convergence has not been reached by then.

One practical advantage of this design is that extending the circular boundary in the future requires no changes to the engine itself. If a new quantity needs to be folded into the loop, it is enough to tag its formula as *circular* in *formulas_map.json*. The iteration picks it up automatically on the next run.

4.3.2 Convergence validation

The behaviour of the solver was validated by tracing the value of each circular target across successive iterations for all Rwanda scenarios. In practice, every fuel market and techno-economic scenario converged within two or three iterations, with final residuals of order 10^{-13} — well below the 10^{-10} stopping tolerance and already at the practical limit of double-precision floating-point arithmetic. The loop is well-conditioned: it does not need many iterations to settle, and when it does settle, it has genuinely converged rather than stalled.

As a separate check, the balance-sheet identity — Assets equal Liabilities plus Equity, every period — was verified after each run. It held within floating-point noise across all scenarios and all planning years. That matters because the identity is not enforced by the solver directly: it emerges from the financial logic of the three chained statements. If the converged values were internally inconsistent, the balance sheet would not close. The fact that it does, consistently, confirms that the iterative solution is not just numerically stable in isolation but financially coherent across the full model.

4.3.3 Dashboard visualisation engine

The Summary Financing view has a specific job: let the planner compare the key financial indicators of every loaded scenario at a glance, without having to navigate into individual financial statements one by one. Doing that well requires its own computation layer — one that is clean, self-contained, and completely independent of the main engine. The platform builds exactly that inside the frontend, driven by a dedicated configuration file: `visualizations_map.json`.

This second engine borrows the same general philosophy as the main one. It is declarative, placeholder-driven, and built from a small catalogue of named operations. But it deliberately does much less, and for good reason. By the time the dashboard runs, all values have already converged in the financial statements and carbon-credits sheets; what remains is simply to combine those settled figures into the series and totals the charts need. There are no conditionals, no complex cross-period lookups, and certainly no iterative solving. The operation catalogue reduces to a handful of primitives — copy, addition, negation — which turns out to be sufficient to compute every dashboard metric. Keeping the catalogue that small also means the visualisation layer stays just as auditable as the rest of the platform: any domain expert can open `visualizations_map.json` and read exactly what each chart is computing.

The dashboard itself is organised around two families of views, each answering a different kind of question. Graph filters render time series that compare scenarios along specific axes: the evolution of the financing mix over the planning horizon, or how the capital structure shifts year by year as debt is repaid and equity grows. Value filters go in the opposite direction. Rather than showing how a quantity evolves over time, they collapse it to a single number per scenario — total revenues, EBITDA, grants relative to CAPEX, new debt raised, equity contributed, total CAPEX, or carbon-credit income — so the planner can read across scenarios in a single view without any further navigation. Both families draw from the same converged outputs and share a consistent colour palette: a given scenario always appears in the same colour wherever it shows up, across every chart and every view.

The decision to keep the two engines separate rather than extending the main one was deliberate. The main engine has a hard job: produce a fully converged, internally consistent set of financial figures across all periods, models and fuel markets. The visualisation engine has a much simpler one: take those numbers and arrange them into the views the dashboard wants. Mixing the two would mean that a change to a dashboard chart could, in principle, introduce a side effect into the financial computation. Separate files, separate configurations and a clean boundary between the two layers make that impossible.

Chapter 5

Results

This chapter presents the results of the project in two parts. The first is a walkthrough of the platform itself: what the user actually sees at each stage, what the interface enables, and how the different sections connect into a coherent workflow from first login to a fully populated set of financial statements. The second part zooms in on the Rwanda case study — the three scenarios loaded into the platform, the financial outputs they produce, and the checks that confirm those outputs are both internally consistent and numerically correct.

5.1 Platform walkthrough

The walkthrough follows the natural sequence a user would take when setting up and running a new country scenario from scratch. The point is not to document every button and dropdown, but to show how the different pieces of the platform connect in practice and what kind of experience the interface is designed to deliver.

5.1.1 Welcome screen and country selection

The platform opens on a welcome screen that identifies the tool and invites the user to begin by selecting a scenario (Figure 5.1, left). There is exactly one action available at this stage: a single *Select scenario* button. That is a deliberate choice. An empty dashboard with too many options at once is one of the fastest ways to lose a non-technical user; starting with a single, unambiguous next step removes that friction entirely, regardless of the user's background.

From there, the country selection page lets the user either pick from the countries already registered in the platform or add a new one (Figure 5.1, right). Adding a country requires only three pieces of information: a name, a corporate tax rate, and an inflation rate. These two parameters are not decorative — they propagate automatically to every financial computation run within that country context. The tax rate feeds directly into the iterative solver and the WACC calculation; the inflation rate conditions cost projections across the planning horizon. Setting them once at the country level means the planner never has to re-enter them at any later stage, and there is no risk of inconsistency across models or fuel markets within the same scenario. For Rwanda, the platform is configured with a tax rate of 28 % and an inflation rate of 5 %, consistent with the national planning documents the case study draws on.

The multi-country design is intentional and goes beyond convenience. The platform is built to scale: once a country is registered, the rest of the structure — techno-

economic models, fuel markets, Excel templates — is generated automatically around it. A second country can be added without touching the configuration of the first, and both can run independent financial computations within the same instance of the platform. This is what allows a development agency or a research team to maintain and compare clean-cooking financial models for several countries simultaneously, without maintaining separate tools or reconciling results by hand.

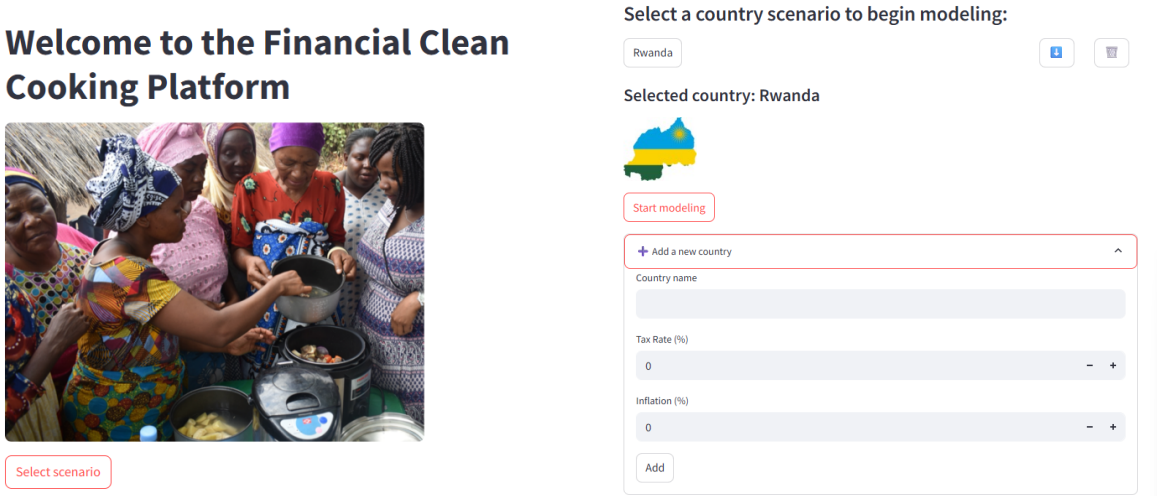


Figure 5.1: Left: welcome screen of CC-WBT, the entry point to the platform. Right: country selection page, with Rwanda configured at 28 % tax and 5 % inflation; new countries can be added at any time.

5.1.2 Main dashboard and scenario management

Once a country is selected, the platform opens on the main dashboard. This is where the planning horizon is set and where the techno-economic models that will populate the analysis are created and managed (Figure 5.2). For Rwanda, the horizon runs from 2023 to 2034 — twelve years that cover the period of active deployment and the early years of steady-state operation. Two active scenarios sit alongside the mandatory Baseline: *CleanStep*, which represents a moderate, phased transition, and *Aligned*, which targets a more ambitious deployment pace in line with national electrification commitments.

The sidebar that appears at this stage is the main navigation spine of the platform. Its structure is not static: it adapts to what the user is doing. Before any model is selected, it shows only the sections that apply to the whole country scenario. The moment a model is selected, a second layer unfolds with the model-specific inputs, the capital structure designer, and the financial statements for that model. This progressive disclosure keeps the interface from feeling overwhelming at the start, while making everything accessible once the user is ready for it.

Scenario management also handles the practical side of moving data in and out of the platform. Each scenario can be downloaded as a ZIP archive containing all its input

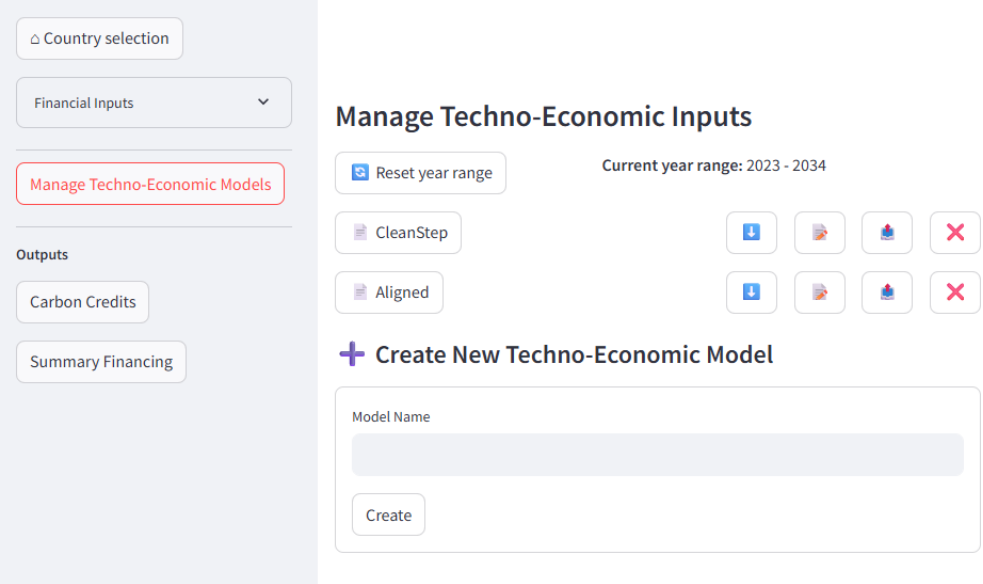


Figure 5.2: Main dashboard showing the sidebar navigation and the scenario management panel, with CleanStep and Aligned loaded for Rwanda (year range 2023–2034).

and output files — a self-contained bundle that can be shared, archived, or reopened later and reproduce exactly the same results. For data entry, the platform supports a download-upload flow: the user downloads a blank Excel template, fills it in offline at their own pace, and uploads it back. The platform stages the uploaded file and applies its contents on the next computation pass, decoupling data entry from computation and avoiding any risk of partial updates mid-run.

5.1.3 Financial inputs

The *Financial Inputs* section holds the parameters that apply uniformly across every model within the country (Figure 5.3). For each active fuel market, the planner sets the expected level of non-technical losses, the trade receivables expressed as days of revenue, and the trade payables expressed as days of OPEX. Non-technical losses capture the share of energy that is delivered but not billed, directly reducing the revenues the model can count on. Trade receivables and payables shape the working capital dynamics: how much cash the operator has tied up waiting to collect from customers, and how much it owes to suppliers. Because these values are shared across all models, they are set once and propagated automatically to every scenario, with no risk of inconsistency creeping in between them. The fuel markets themselves — Electricity, LPG and Carbon Credits in the Rwanda case — are not hardcoded. They can be extended or reduced at any time, and the platform updates its internal template structure accordingly, creating the missing sheets and propagating the new market's parameters to every workbook that needs them.

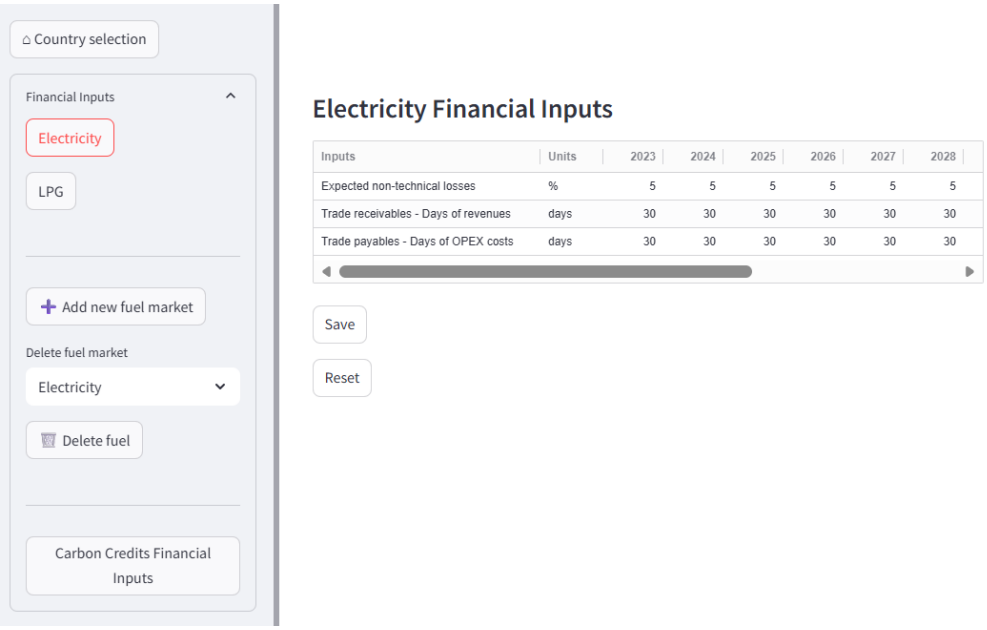


Figure 5.3: Financial Inputs section: country-level parameters shared across all techno-economic models, shown here for the LPG fuel market.

5.1.4 Techno-economic inputs and tariffs

At the model level, the *Techno-Economic Inputs* section is where the deployment assumptions become numbers (Figure 5.4). The view presents two layers side by side. The upper layer shows the country-level financial parameters inherited from the previous section — non-technical losses, trade receivables, trade payables — displayed here for reference but locked from editing, since they apply uniformly across all models. The lower layer is the model-specific input table, where the planner enters annual demand, CAPEX, OPEX and depreciation schedules for each fuel market and each year of the planning horizon.

The electricity sector gets special treatment throughout. Because access to electricity is not binary, the platform distinguishes between on-grid and off-grid systems and between two access tiers — full access with e-cooking (*Electricity & E-Cooking*) and basic low-access electrification (*Electricity Low Access*) — each carrying its own demand profile, cost structure and depreciation timeline. This decomposition matters: lumping the two together would mask the marginal cost of the clean-cooking transition over and above basic electrification, which is precisely the figure planners and regulators need to see.

The validation layer described in Section 4.1 is active throughout this table. Cells that fall outside their declared bounds are flagged immediately, the save button is disabled, and the planner sees exactly which row and year column contain the offending value. This catches unit errors — a CAPEX figure entered in millions when the column expects thousands, for instance — before they can propagate silently into the financial statements.

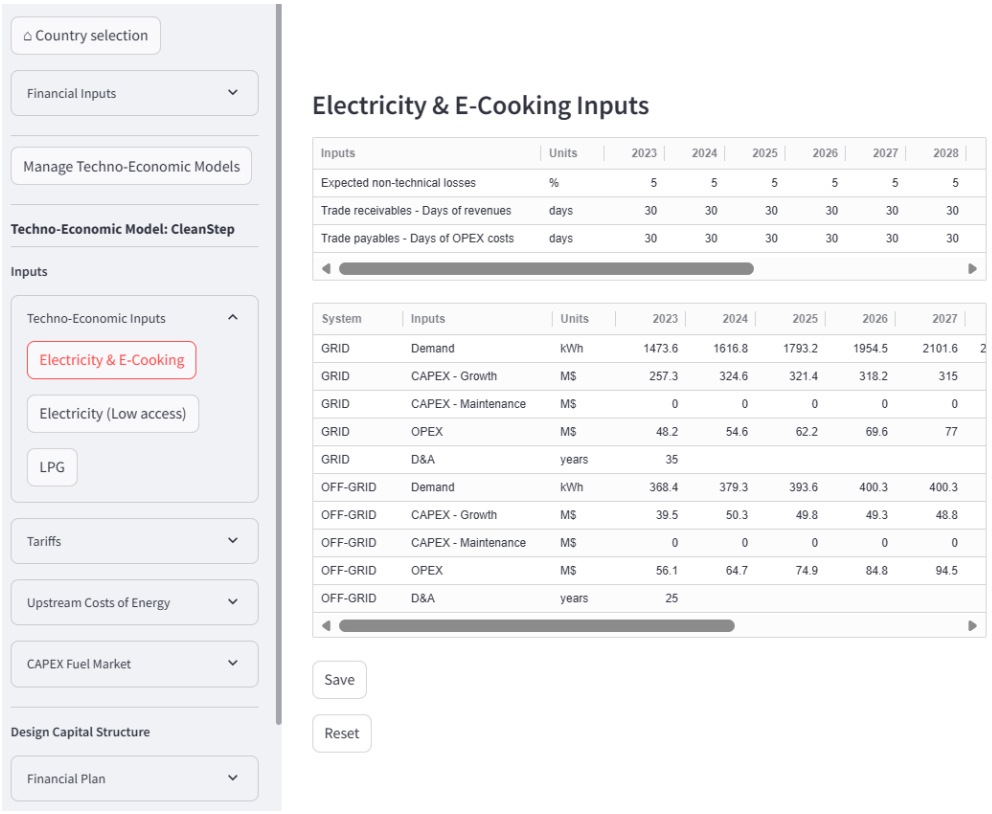


Figure 5.4: Techno-Economic Inputs for the CleanStep scenario: on-grid and off-grid demand, CAPEX, OPEX and depreciation for the Electricity & E-Cooking fuel market, 2023–2034.

Tariffs are configured separately for each fuel market and each model (Figure 5.5). The planner has two options. The first is to enter an initial tariff value and let the platform project it forward automatically by compounding the country's inflation rate year on year — a reasonable default when the pricing trajectory is not yet determined. The second is to enter annual values manually, which gives full control over the pricing scenario and allows the planner to model tariff steps, regulatory resets or deliberate departures from inflation-linked pricing.

The interface makes the projected trajectory visible before the user saves it, which matters more than it might seem. A tariff path that looks reasonable as an initial value and an inflation rate can still produce a curve that grows too slowly to close the viability gap, or too quickly to be politically defensible. Seeing the full twelve-year projection laid out year by year gives the planner the chance to spot those problems immediately — and to switch to manual entry for the years where the automatic projection diverges from what the regulatory environment actually allows. For Rwanda, electricity tariffs in the Aligned scenario start at 0.15 c$/KWh in 2023 and rise gradually to 0.26 c$/KWh by 2034, a trajectory designed to close the gap between the initial subsidised rate and the long-run cost of service gradually, rather than through a single corrective step.

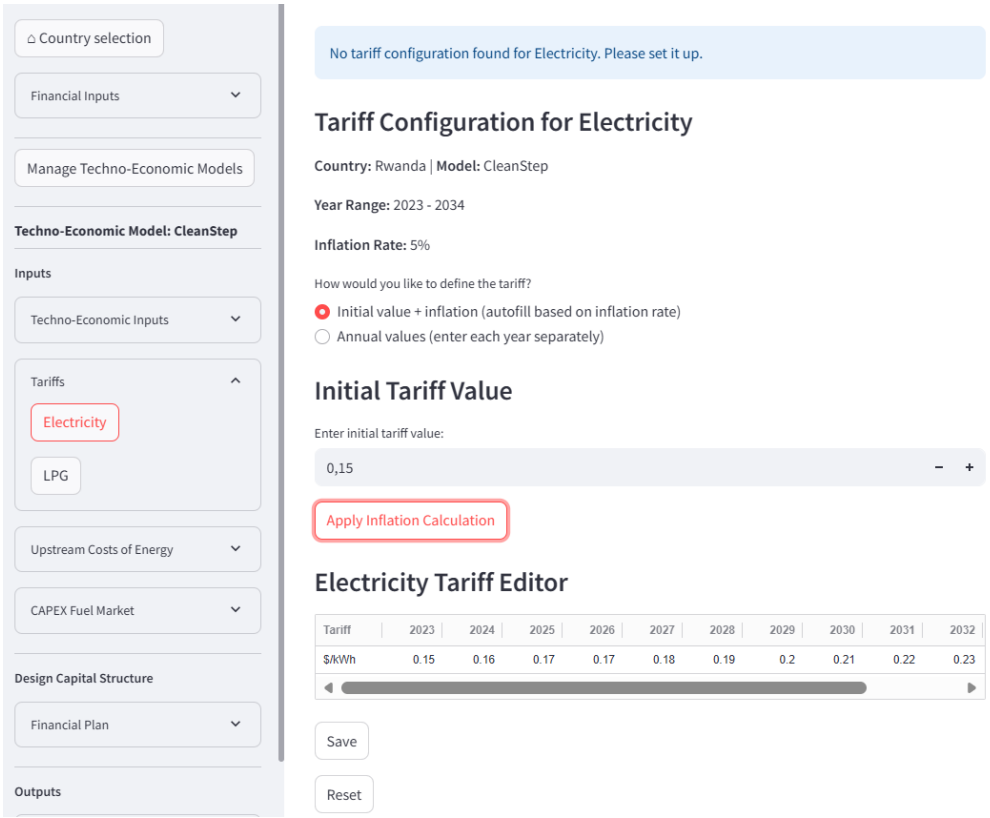


Figure 5.5: Tariff configuration for Electricity in the Aligned scenario: initial value with automatic inflation projection, or manual annual entry.

5.1.5 CAPEX fuel markets and capital structure

The *CAPEX Fuel Market* section is a read-only view of the investment schedule the backend has derived from the techno-economic inputs (Figure 5.6). The planner does not enter CAPEX here — that already happened in the previous section. What this view provides is the result of the engine's processing: growth CAPEX, maintenance CAPEX, accumulated investment totals, annual depreciation, and the evolving Regulatory Asset Base (RAB), all computed automatically for every fuel market and every year of the horizon.

The RAB is central to the whole cost-of-service logic. It is the net value of capital assets in service at any given point in time, and it is the base on which the regulated return is calculated each year. Its evolution follows a simple but consequential rule:

$$\mathrm{RAB}(t) \;=\; \mathrm{RAB}(t-1) + \mathrm{Growth\,CAPEX}(t) - \mathrm{Depreciation}(t) \tag{5.1}$$

As long as new investment arrives faster than existing assets depreciate, the RAB grows — expanding the capital base on which the cost of service is built and, with it, the allowed return the operator can recover through tariffs. Once deployment tapers off and the investment wave subsides, depreciation begins to dominate and the RAB gradually unwinds. Front-loading investment therefore has a direct and lasting effect: it inflates the RAB in the early years, raises the cost of service that those years carry,

and increases the financing gap that tariffs and subsidies must close. The platform makes this dynamic visible immediately, without any manual recalculation.

The on-grid and off-grid split runs through this view as well. The two systems have different cost structures, different asset lives and different depreciation timelines, so averaging them into a single CAPEX figure would distort the economics of both. For the Electricity & E-Cooking sector in the CleanStep scenario, on-grid growth CAPEX opens at 257.3 M$ in 2023 with a starting RAB of 249.9 M$, while the off-grid tranche adds a further 39.5 M$ in the same year with a corresponding RAB of 38.0 M$. Both series grow in parallel as investment compounds across the planning horizon, and their combined trajectory feeds directly into the cost-of-service calculation that closes the financial statements.

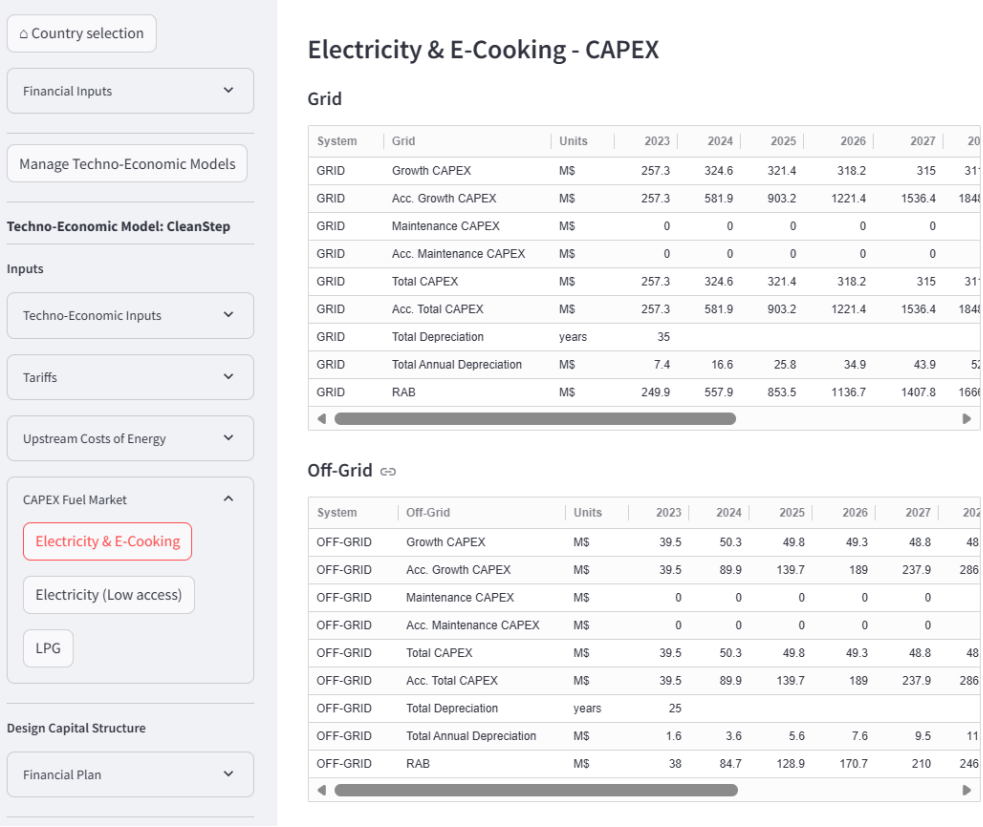


Figure 5.6: CAPEX Fuel Market view for Electricity & E-Cooking in the CleanStep scenario: backend-calculated on-grid and off-grid investment schedules, RAB evolution, and depreciation.

The *Design Capital Structure* section is where the financing plan takes shape (Figure 5.7). For each fuel market, the planner specifies the equity share and its required return, the grants share and its realisation schedule, and the debt share with its interest rate, grace period and amortisation tenor. The platform responds immediately: it computes the resulting WACC, lays out the full financing breakdown, and flags any year in which the debt service schedule would exceed the cash available — a solvency warning the planner can act on before committing to a structure.

The WACC is the blended cost of the capital deployed, weighted by the share each

source contributes and adjusted for the tax treatment of debt:

$$\text{WACC} \;=\; w_e\, k_e \;+\; w_d\, k_d\,(1-\tau) \;+\; w_g \cdot 0 \tag{5.2}$$

where w_e, w_d and w_g are the equity, debt and grants shares, k_e and k_d their respective costs, and τ the corporate tax rate. Grants enter the blend at zero cost because they carry no financial obligation to the operator: no interest, no principal repayment, no required return. A high grants share therefore pulls the blended WACC down substantially, even when the equity and debt tranches are expensive. The tax shield on debt — the $(1-\tau)$ factor — adds a further reduction: because interest payments are tax-deductible, the effective cost of debt to the operator is lower than the nominal rate the lender charges.

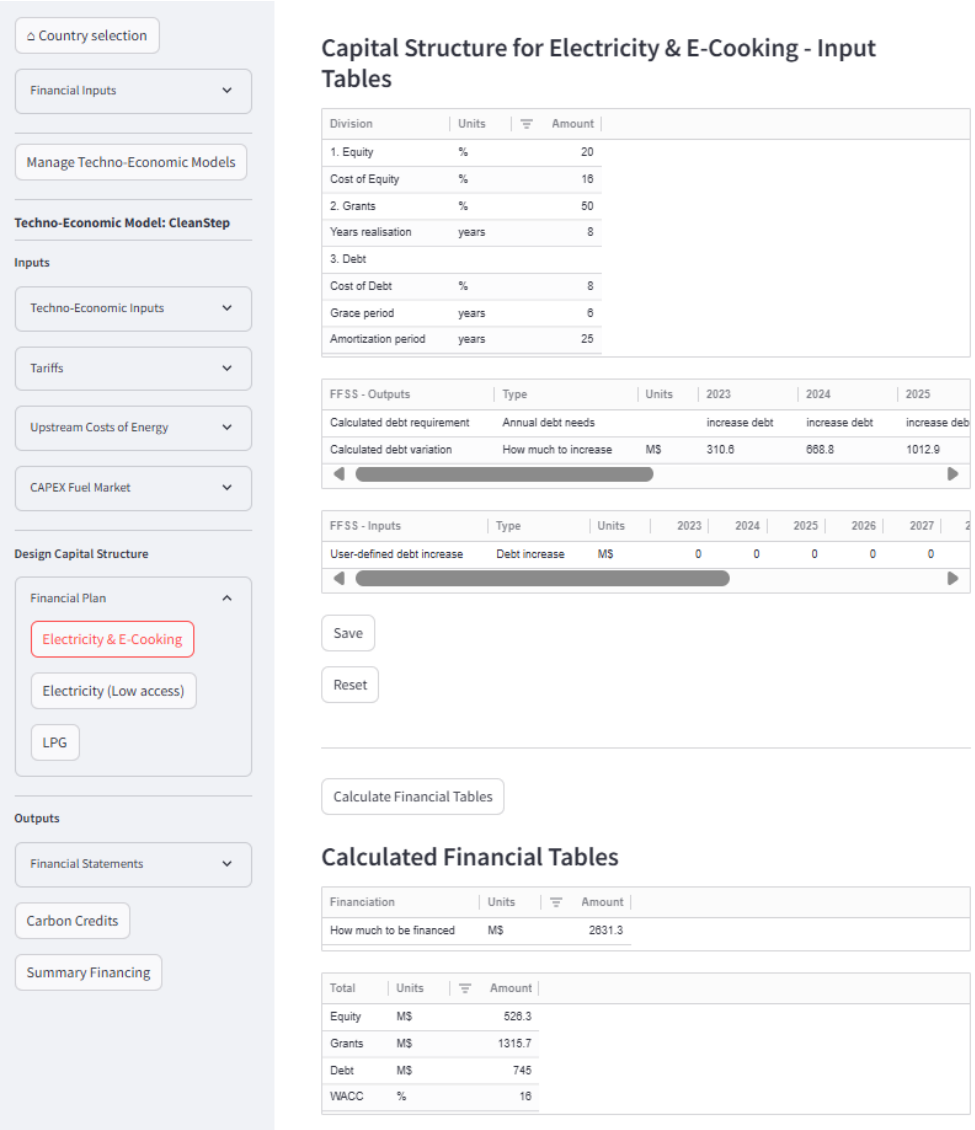


Figure 5.7: Design Capital Structure for Electricity & E-Cooking in the CleanStep scenario: equity, grants and debt parameters, with the computed WACC and total financing breakdown.

The solvency indicator matters even when the WACC looks reasonable. A capital structure can produce an acceptable blended cost while still creating years in which the required debt service — principal plus interest — outstrips the cash the operation actually generates. This tends to happen when grace periods end abruptly, when revenues are still ramping up in the early years of deployment, or when the amortisation schedule front-loads repayments too aggressively. The platform surfaces these mismatches directly, period by period, so the planner can adjust tenors or shares before the structure is locked in.

One feature of the capital structure designer deserves explicit mention. Alongside the financing parameters, the platform displays a read-only row labelled *Calculated debt requirement*: a period-by-period estimate of how much new debt would need to be raised to keep the cash balance non-negative, given the investment schedule and the current financing mix. This is the platform's answer to the question of how much external borrowing the plan actually implies. Directly below it sits an editable row, *User-defined debt increase*, where the planner specifies how much debt is actually raised in each period. The two rows do not have to match. A planner might choose to raise more debt than strictly required in an early period to build a liquidity buffer, or less if grant disbursements are expected to arrive ahead of schedule. The platform accepts whatever the planner enters and propagates it through the financial statements, while the calculated row stays visible as a reference — a continuous reminder of the floor the plan needs to stay above to remain solvent.

For the Electricity & E-Cooking sector in the CleanStep scenario, the total capital requirement stands at 2,631.3 M$. It is split across 20 % equity at a 16 % required return (526.3 M$), 50 % in grants with an eight-year realisation schedule (1,315.7 M$), and 30 % in debt at 8 % interest with a six-year grace period and a 25-year amortisation tenor (745.0 M$). The resulting WACC is 16 %, driven by the equity tranche: although the grant component is large and enters the blend at zero cost, the equity investors still require a 16 % return on the capital they contribute, and that requirement is what dominates the blended rate.

5.1.6 Financial statements

The *Financial Statements* section is where the platform delivers its core output (Figure 5.8). For every fuel market within a scenario, it presents the full three-statement model — Profit & Loss, Balance Sheet and Cash Flow — as an integrated, year-by-year table. The word integrated matters here. The three statements are not computed independently and then placed side by side; they are chained. Net income from the P&L flows into retained earnings on the balance sheet. Depreciation charged in the P&L reduces the book value of PP&E on the asset side. Debt drawdowns and repayments alter both the liability side of the balance sheet and the cash flow from financing activities. A change anywhere in the chain propagates through the rest automatically, and the platform enforces consistency at every period without any manual reconciliation.

The P&L structure runs from tariff revenues and any upstream income down through OPEX, depreciation and interest charges to reach earnings before tax. Corporate tax is then applied at the country rate — 28 % for Rwanda — and the residual is net income. One line in the P&L deserves particular attention: the Long-Term Sub-

sidy. It sits between revenues and the Annual Cost of Service and captures exactly the gap between what the tariff recovers and what the full cost of delivering the service actually requires. In the early years of a deployment, when tariffs are still below cost-recovery levels and investment is ramping up fast, this line can be substantial. As tariffs rise with inflation and the debt burden begins to reduce, the subsidy need typically narrows — and the model shows that narrowing, year by year, for every scenario and every fuel market. For the Electricity & E-Cooking sector in the CleanStep scenario, total revenues open at 315.5 M$ in 2023 and reach 1110.1 M$ by 2034, while the LTS starts at zero and grows to 252.7 M$ as investment and the associated cost of service accelerate ahead of tariff recovery.

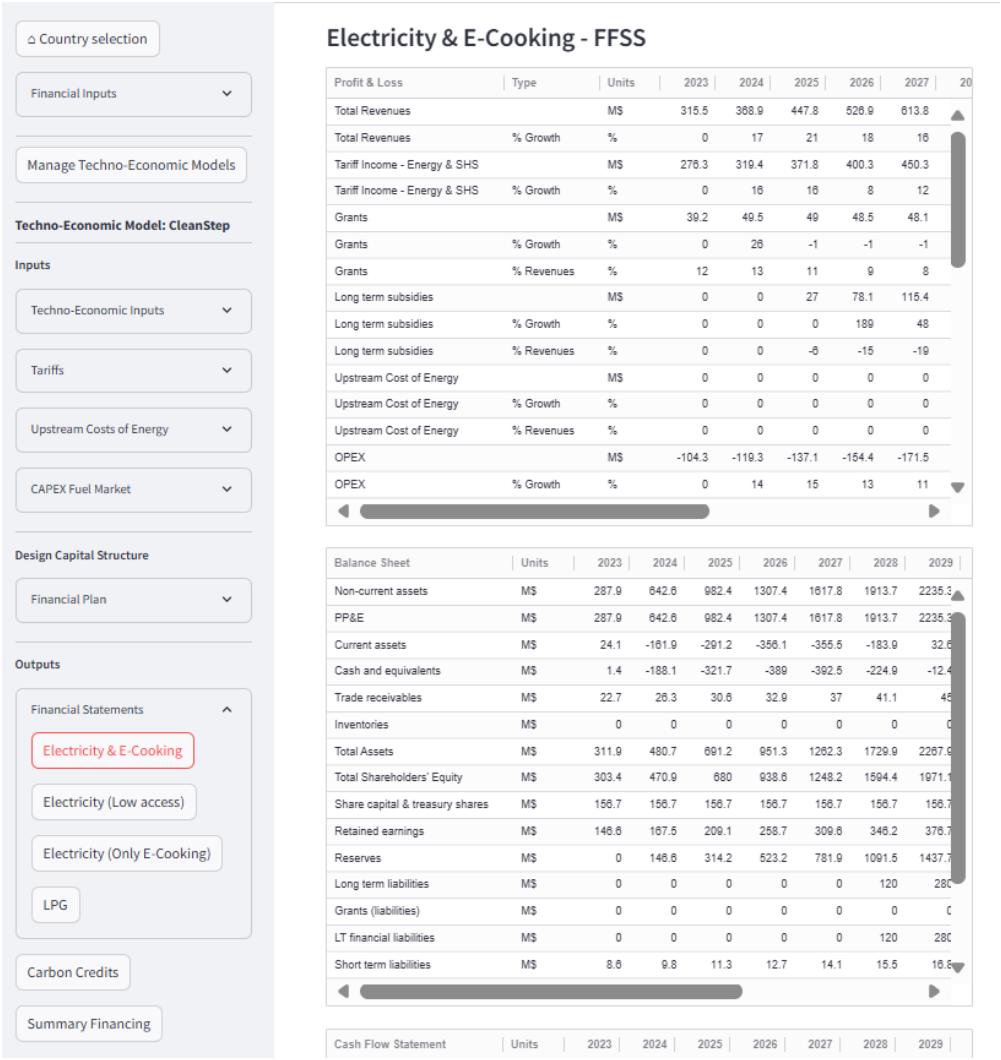


Figure 5.8: Financial Statements output for the Electricity & E-Cooking sector in the CleanStep scenario: P&L (2023–2027) and Balance Sheet (2023–2029) shown as excerpts; the full planning horizon runs to 2034.

On the balance sheet, total assets grow from 311.9 M$ in 2023 to 4,259 M$ by 2034, driven almost entirely by the accumulation of PP&E as CAPEX compounds year on year. This is what a large-scale infrastructure deployment looks like from a balance sheet perspective: each year of new investment adds to the asset base faster than depreciation drains it, and the cumulative effect over twelve years is a balance sheet that is more than thirteen times larger than it was at the start. The liability and equity side must grow in exactly the same proportion at every step — debt drawdowns, grant injections, retained earnings and equity contributions have to add up to the same total as the assets they are funding, period by period, with no slack. That identity is the most fundamental consistency check any financial model has to pass, and it is not trivially satisfied in a model where three chained statements evolve simultaneously over twelve years. That it holds throughout, as Section 5.3 confirms in detail, is one of the clearest signs that the underlying computation is doing what it is supposed to do.

5.1.7 Carbon credits

The financial statements show whether a deployment plan is viable. The *Carbon Credits* section shows something they cannot: how much of the emissions reduction each scenario achieves could be converted into income (Figure 5.9). The logic is straightforward. The user enters annual CO_2 emissions for the Baseline and for each active scenario. The platform calculates the difference year by year, applies a certification factor that reflects what fraction of those avoided emissions can realistically be converted into tradeable credits, and then applies a liquidity discount to account for the fact that not all certified credits will find a buyer at the full market price. The result is a year-by-year estimate of the carbon-credit income each scenario could generate — a figure the planner can stress-test freely by adjusting the certification fraction and the carbon price to cover optimistic and conservative market assumptions alike.

Formally, the avoided emissions for a scenario s in year t are:

$$\Delta E_s(t) \;=\; E_{\text{Baseline}}(t) - E_s(t) \tag{5.3}$$

where both quantities are expressed in CO_2eq MT/year. The carbon-credit income then follows from multiplying the certifiable and liquid fraction of $\Delta E_s(t)$ by the assumed price per tonne.

For Rwanda, the Baseline emissions trajectory opens at 22 CO_2eq MT/y in 2023 and declines to around 16 MT/y by 2034 through organic transition alone — households switching fuels for economic reasons, grid coverage expanding at its own pace. That is a meaningful reduction, but a slow one. What the two active scenarios add on top is considerably sharper.

CleanStep ramps up faster in the early and mid periods: by 2029 it is avoiding 7.3 MT/y against the Baseline, roughly 40 % of what the Baseline still emits at that point. Aligned follows a steadier pace and reaches 5.2 MT/y of avoided emissions in the same year, closing the gap progressively until the two scenarios nearly converge by 2034 at around 6.7 and 6.5 MT/y respectively. The story the data tells is that the choice between the two plans is less about the destination — both end up in a similar place by the end of the horizon — and more about how quickly the reductions

materialise and what that implies for the carbon-credit income in the years when the financing gap is at its widest.

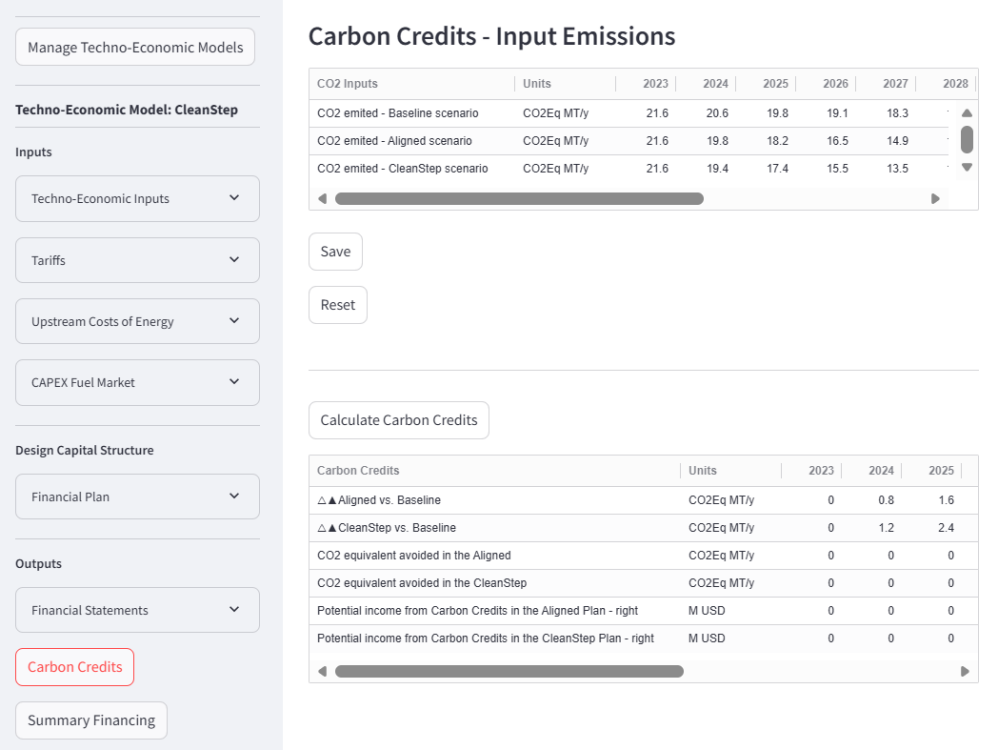


Figure 5.9: Carbon Credits section: avoided CO_2 emissions relative to the Baseline scenario and the corresponding potential carbon credit income for the Aligned and CleanStep scenarios.

One thing worth flagging: the carbon credit income computed in this section is not fed into the financial statements. Carbon markets for clean cooking are still an emerging mechanism and, at the time of writing, not yet systematically implemented in the Rwanda plan. The figures presented here are therefore best read as a sensitivity analysis — an estimate of the additional revenue stream that could become available if and when a carbon credit mechanism is formalised, rather than as an input to the base financial projections.

5.1.8 Summary Financing dashboard

By the time a planner has worked through the financial inputs, the capital structure and the full three-statement model for each scenario, they are sitting on a lot of numbers. The *Summary Financing* dashboard exists to answer the one question all those numbers are ultimately trying to answer: which scenario, and why (Figure 5.11). Rather than navigating back and forth between individual scenario views, the planner sees everything side by side through a compact set of dynamic Plotly charts, each one cutting the comparison along a different axis.

The capital structure charts show how the financing mix — equity, grants and debt — is distributed across fuel sectors for each scenario. This view is more useful than

it might first appear. A capital structure that looks reasonable at the aggregate level can hide a fuel market that is almost entirely grant-dependent, or one where the debt tranche is so large that a modest shortfall in revenues would trigger a solvency problem. Placing the scenarios side by side makes those imbalances visible at a glance, without having to open each one individually and reconstruct the comparison mentally.

The CAPEX profiles plot the year-by-year investment requirements across the planning horizon. This is where the difference between CleanStep and Aligned is most legible: Aligned front-loads spending in the early years, which translates directly into a larger RAB, a higher Annual Cost of Service and a wider financing gap precisely when the platform is least mature. CleanStep reaches a similar endpoint but spreads the investment more evenly, trading some early-year emissions impact for a smoother financing profile. Seeing both curves on the same chart makes that trade-off concrete rather than abstract.

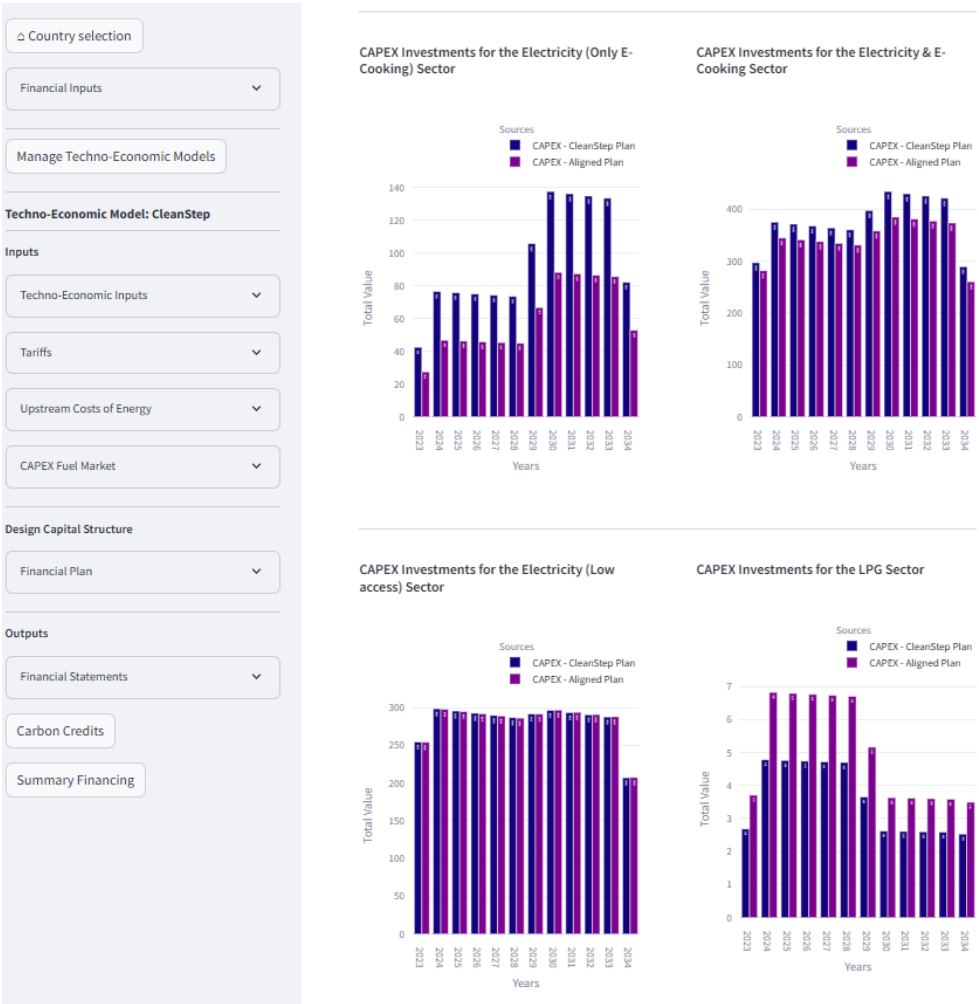


Figure 5.10: CAPEX investment profiles for CleanStep and Aligned across all fuel sectors: Electricity (Only E-Cooking), Electricity & E-Cooking, Electricity (Low Access) and LPG, 2023–2034.

The emissions trajectories complete the picture by placing the CO_2 reductions of each scenario on a single axis, so the environmental benefit can be read directly against the financial cost without switching views. A scenario that looks expensive in the CAPEX chart may justify that cost if its emissions curve drops faster and further — or it may not. The dashboard lets the planner make that call without reconstructing the comparison from separate tables.

A few design details make the dashboard easier to use in practice. A given scenario always appears in the same colour across every chart, so the eye can track a strategy from one view to the next without re-orienting. The charts are rendered interactively, which means the planner can zoom into a specific window of years or isolate individual series — particularly useful when a financing decision turns on what happens in a narrow stretch of the horizon rather than across the full twelve years. And the dashboard sits at the bottom of the sidebar, below the detailed financial statements, which is the right order: it is not a replacement for the numbers, but a way to orient the planner before they go looking for them.

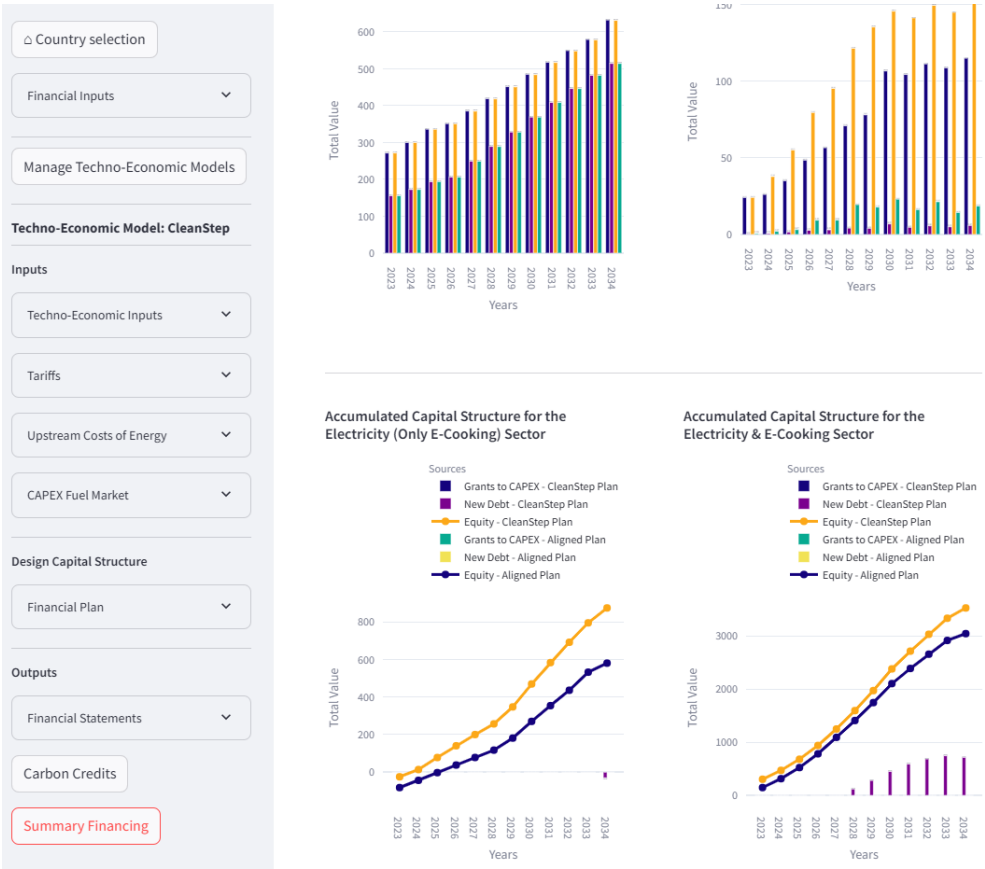


Figure 5.11: Summary Financing Dashboard: accumulated capital structure, CAPEX investments and CO_2 emissions for the CleanStep and Aligned scenarios side by side, across all fuel markets.

5.2 Rwanda case study

CC-WBT was built to address a gap that the NICCP process made concrete. Rwanda's National Integrated Clean Cooking Plan had already established the analytical framework: the fuel markets, the planning horizon, the financial structure. The IIT–SEforALL team had developed a reference financial model to go with it [2], a purpose-built Excel workbook that captured the core logic of the cost-of-service calculation for a single country scenario. What was missing was a platform that could take that logic, make it configurable and extensible, run it across multiple scenarios simultaneously, and produce a full set of auditable and internally consistent financial statements without rebuilding the model from scratch each time. That is what CC-WBT delivers. Because the NICCP reference model existed, Rwanda became both the natural test case and the most rigorous validation available: the platform had to produce, for the same inputs, the same outputs that the reference model had already established as correct.

5.2.1 Scenario descriptions

Three scenarios were loaded into the platform for the 2023–2034 planning horizon. The Baseline models the business-as-usual trajectory: no deliberate clean-cooking intervention, with the existing fuel mix and its associated emissions continuing on their current path. It serves two roles at once. It is the counterfactual against which the other two scenarios are evaluated, and the emissions reference from which carbon credits are computed. The platform enforces this: the Baseline must always be created first, and the name is reserved exclusively for that anchor role.

Table 5.1: Capital structure parameters for CleanStep and Aligned across the three active fuel markets in Rwanda.

Scenario	**Fuel market**	**Equity** (%)	k_e (%)	**Grants** (%)	**Debt** (%)	k_d (%)	**WACC** (%)
CleanStep	Electricity & E-Cooking	20	16	50	30	8	16
	Electricity (Low access)	25	12	55	20	8	12
	LPG	50	12	20	30	8	12
Aligned	Electricity & E-Cooking	20	12	48	32	8	12
	Electricity (Low access)	25	12	50	25	8	12
	LPG	40	12	0	60	8	12

CleanStep represents a moderate, phased transition: an organised rollout of LPG and electric cooking, front-loaded in the early years but deliberately paced to avoid the solvency stress that comes with scaling up too aggressively before revenues have had time to grow. Aligned shares the same tariff structure but applies a more ambitious deployment programme, with higher CAPEX in the early years, faster scale-up of both on-grid and off-grid infrastructure, and a financing mix that reflects greater confidence in the debt market. Keeping the tariff trajectory identical across both active scenarios

was a deliberate modelling choice. It isolates the effect of deployment pace and capital structure on financial outcomes, so the comparison between the two reads cleanly.

The two scenarios differ not just in pace but in how they are financed. Table 5.1 sets out the capital structure for each scenario and each fuel market. The most consequential difference sits in the Electricity & E-Cooking sector. CleanStep carries a cost of equity of 16 %, reflecting a higher risk premium from a more conservative investor base. Aligned assumes 12 %, which drives the WACC down from 16 % to 12 % for that sector. Since the WACC feeds directly into the Annual Cost of Service through the regulated return on the RAB, that four-point difference compounds across the entire planning horizon. The LPG sector tells a different story. Aligned relies entirely on debt and equity with no grant support, while CleanStep includes a 20 % grant tranche, making Aligned's LPG financing considerably more exposed to interest rate risk.

5.2.2 Validation against the NICCP reference model

The same Rwanda input data used to build the NICCP model was loaded into CC-WBT, and the outputs of the two were compared variable by variable, year by year, for every fuel market and every scenario.

Before any comparison could happen, the dependency structure of the model had to be understood. Financial variables chain into each other: CAPEX determines the RAB, the RAB feeds into the Annual Cost of Service, the ACoSt drives the Long Term Subsidies needed to cover it, those subsidies enter Total Revenues, revenues determine the taxable result, and Taxes flow back into the ACoSt. Mapping this hierarchy was the first piece of rigorous work. Starting from the most primitive inputs, a list of formulas was built up one by one, each added only after the variables it depended on had already been verified. The ordering mattered. A single upstream error contaminates every value downstream, making any comparison below that point meaningless. It was also during this process that the circular dependency at the core of the cost-of-service calculation surfaced clearly: the ACoSt cannot be computed without taxes, taxes cannot be computed without EBT, EBT depends on Long Term Subsidies, and Long Term Subsidies are defined as what it takes to cover the ACoSt. That loop is what motivated the iterative solver described in Section 4.3.1.

Figure 5.12 shows the CAPEX comparison for the Electricity & E-Cooking sector. The left panel is the NICCP reference model; the right panel is the equivalent view in CC-WBT. The point of the comparison is simple: the numbers are the same. Growth CAPEX, accumulated investment totals, annual depreciation and the RAB trajectory match across the full 2023–2034 horizon for both on-grid and off-grid tranches. That match, replicated across every variable and every fuel market, is what validation looked like in practice.

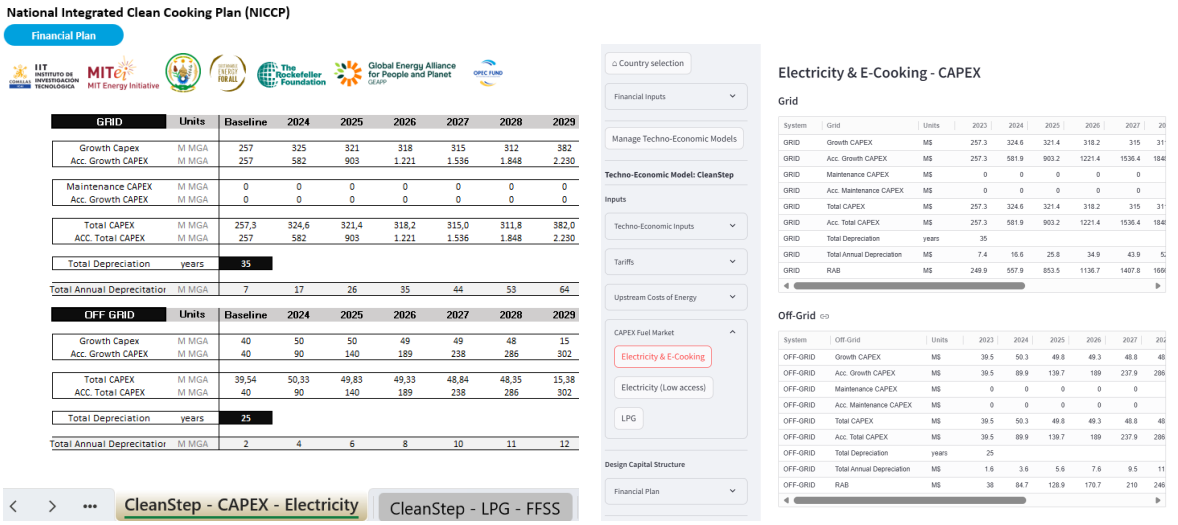


Figure 5.12: CAPEX validation for the Electricity & E-Cooking sector in the CleanStep scenario. Left: reference values from the NICCP model. Right: equivalent output from CC-WBT. Growth CAPEX, accumulated totals, depreciation and RAB match across the full 2023–2034 horizon for both on-grid and off-grid tranches.

Getting to that match took time. Many outputs did not agree at first, which was expected: the computation engine was being built at the same time as the formulas were being defined. Every new formula added to `formulas_map.json` introduced a new surface for error. A wrong operand index, a missing negation, a period reference off by one. When a value did not match, the debugging process could not stop at the row level. A discrepancy in the Long Term Subsidies line could originate anywhere upstream: in the ACoSt accumulation, in how the tax rate was applied, in the working capital variation. The only way to find it was to decompose both the NICCP formula and the `formulas_map` entry into their atomic operations and compare them step by step, printing intermediate values from the engine at each operation and checking the equivalent in the Excel formula bar, until the divergence became visible. The diagnosis then had to determine whether the problem was an implementation error in the engine or a modelling error in the formula specification.

There was a third possibility, and it came up more than once. Some formulas in the NICCP model were not yet fully defined, returning all zeros or structured in a way that did not match the intended logic. Resolving those cases required going back to the reference model's author directly and establishing the correct definition before any comparison could be meaningful. Slow, but it produced a cleaner reference model and a more rigorously specified `formulas_map` as a result.

The Long Term Subsidies formula illustrates how the declarative engine captures financial logic atomically:

$$\mathrm{LTS}(t) = \max\Big(0,\ \mathrm{ACoSt}(t) - \mathrm{TariffRev}(t)\Big) \tag{5.4}$$

The floor at zero reflects the fact that subsidies cannot be negative: if tariff revenues exceed the cost of service, the surplus is a profit, not a subsidy. In `formulas_map.json`, this translates into exactly two atomic steps, a subtraction followed by a max, with the `circular` tag marking it as part of the iterative loop:

```
{
  "circular": true,
  "target": "Long term subsidies",
  "sources": [
    "{country}/financial-statements-{model}.xlsx
      ::{sheet}::Annual Cost of Service",
    "{country}/financial-statements-{model}.xlsx
      ::{sheet}::Tariff Income - Energy & SHS"
  ],
  "formula_steps": [
    { "op": "subtraction",
      "operands": [["index", 0], ["index", 1]],
      "result": "temp1" },
    { "op": "max",
      "operands": [["literal", 0], ["index", "temp1"]],
      "result": "final" }
  ]
}
```

Decomposing every formula to this level of granularity was what made it possible to locate discrepancies that would have been invisible at the row level. A mismatch in `temp1` points directly at the subtraction step; a correct `temp1` but wrong `final` points at the max. There is no ambiguity about where to look.

The full validation covered every formula, every fuel market, and both active scenarios. Variables that the NICCP model does not compute directly, ACoSt, RAB and WACC, were audited separately by reconstructing their values from first principles using the converged outputs and checking consistency with the definitions in Chapter 3. The platform has since been adopted by SEforALL for active use in clean-cooking financial planning across multiple countries. That, more than any internal check, is the practical confirmation that the outputs are right.

5.3 Numerical validation

Two independent checks confirm that the platform produces financially consistent outputs, over and above whether the individual formula values match the reference model.

5.3.1 Balance sheet consistency

The most fundamental requirement of any financial model is that the balance sheet closes at every period. Total Assets must equal Total Liabilities plus Total Equity, year after year, throughout the full planning horizon. This is not trivially satisfied in a model of this complexity. Net income flows into retained earnings. Depreciation reduces the book value of PP&E on the asset side while accumulated D&A adjusts the equity side in step. Debt drawdowns alter both long-term liabilities and the cash position. Grant realisations reduce the grants liability and increase the cash balance

simultaneously. Every one of these movements has to stay internally consistent across three chained statements for the identity to hold at every period.

The check was run for both active scenarios across all fuel markets and all twelve periods of the Rwanda planning horizon. In every case, the identity

$$\text{Total Assets}(t) \;=\; \text{Total Liabilities}(t) + \text{Total Equity}(t) \tag{5.5}$$

held within floating-point numerical noise, with discrepancies of order 10^{-10} M$ or less, well below any economically meaningful threshold and consistent with rounding at the limit of double-precision arithmetic. The platform writes a dedicated *Check* row into every financial-statements workbook (stored in `backend/Rwanda/`) that displays this residual directly, so the result can be inspected without any additional computation. The same row is also visible in the platform's interface, in the Financial Statements section, as the last row of the balance sheet. Figure 5.13 shows the Check row for the Electricity & E-Cooking sheet of the CleanStep scenario: zero across every year of the horizon.

39	Annual Cost of Service	% Revenues	%	-78	-90	-95	-96	-97	-97	-97	-97	-100	-100	-100	-100
40	Balance Sheet	Type	Units	2023	2024	2025	2026	2027	2028	2029	2030	2031	2032	2033	2034
41	Non-current assets	-	M$	288	643	982	1.307	1.618	1.914	2.235	2.581	2.911	3.224	3.522	3.679
42	PP&E	-	M$	288	643	982	1.307	1.618	1.914	2.235	2.581	2.911	3.224	3.522	3.679
43	Current assets	-	M$	236	20	-139	-233	-261	-118	67	265	410	509	573	580
44	Cash and equivalents	-	M$	213	-6	-169	-266	-298	-159	22	215	355	450	510	510
45	Trade receivables	-	M$	23	26	31	33	37	41	45	50	55	59	64	70
46	Inventories	-	M$	0	0	0	0	0	0	0	0	0	0	0	0
47	Total Assets	-	M$	524	663	844	1.075	1.357	1.796	2.302	2.846	3.321	3.733	4.095	4.259
48	Total Shareholders' Equity	-	M$	515	653	832	1.062	1.343	1.660	2.006	2.379	2.713	3.030	3.332	3.525
49	Share capital & treasury shares	-	M$	392	392	392	392	392	392	392	392	392	392	392	392
50	Retained earnings	-	M$	123	138	180	230	281	318	345	373	334	318	301	193
51	Reserves	-	M$	0	123	261	441	670	951	1.269	1.614	1.987	2.321	2.638	2.940
52	Long term liabilities	-	M$	0	0	0	0	0	120	280	450	590	685	745	715
53	Grants (liabilities)	-	M$	0	0	0	0	0	0	0	0	0	0	0	0
54	LT financial liabilities	-	M$	0	0	0	0	0	120	280	450	590	685	745	715
55	Short term liabilities	-	M$	9	10	11	13	14	15	17	18	18	18	19	19
56	Trade payables	-	M$	9	10	11	13	14	15	17	18	18	18	19	19
57	ST financial liabilities	-	M$	0	0	0	0	0	0	0	0	0	0	0	0
58	Total Liabilities	-	M$	524	663	844	1.075	1.357	1.796	2.302	2.846	3.321	3.733	4.095	4.259
59	Check	-	-	0	0	0	0	0	0	0	0	0	0	0	0
60	Cash Flow Statement	Type	Units	2023	2024	2025	2026	2027	2028	2029	2030	2031	2032	2033	2034
61	- EBITDA (ex-Grants)	-	M$	158	184	243	304	372	434	499	574	646	716	783	838
62	- Taxes	-	M$	-42	-46	-52	-51	-57	-61	-63	-69	-76	-84	-92	-110

Electricity (Only E-Cooking) | Electricity & E-Cooking | LPG | Electricity (Low access)

Figure 5.13: Balance sheet consistency check in the `financial-statements-CleanStep.xlsx` workbook (`backend/Rwanda/`), Electricity & E-Cooking sheet. The *Check* row confirms Assets = Liabilities + Equity at every period within floating-point precision.

5.3.2 Iterative solver convergence

The circular dependency at the core of the cost-of-service calculation is resolved by the Gauss-Seidel solver described in Section 4.3.1. The loop that has to be closed at every period is:

$$\text{ACoSt}(t) \;\rightarrow\; \text{LTS}(t) \;\rightarrow\; \text{EBT}(t) \;\rightarrow\; \text{Taxes}(t) \;\rightarrow\; \text{ACoSt}(t) \tag{5.6}$$

In every fuel market and scenario combination tested for Rwanda, convergence was reached within two to three iterations. Final residuals were of order 10^{-13}, several orders of magnitude below the stopping tolerance of $\varepsilon = 10^{-10}$ and at the practical limit of double-precision arithmetic.

That alone is not enough to be sure. A solver can converge quickly to the wrong answer if the update sequence is subtly wrong or if the stopping criterion is met by accident. To rule that out, the convergence of the circular block was also verified manually. For selected periods, the four circular quantities, ACoSt, LTS, Total Revenues and Taxes, were traced through successive iterations by hand, replicating the update sequence of equations 4.2–4.5 outside the platform and confirming that the values the engine produced at each iteration matched the expected sequence. The manual trace confirmed both the iteration count and the final converged values, ruling out the possibility that the engine was stopping early or converging to a spurious fixed point.

Three independent checks, then: formula-level agreement with the NICCP reference model, the balance-sheet identity holding at every period, and manual verification of the iterative solver. Each one tests something different. Together they build a picture in which the individual formulas are correctly specified, the circular block converges to the right answer, and the three financial statements stay consistent throughout. That is what a working platform looks like.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This project set out to fill a specific and well-documented gap. The absence of an integrated, auditable, open-source tool for the financial planning of clean cooking transitions at country scale was not hypothetical. It showed up concretely in the kind of ad-hoc Excel models that had to be rebuilt from scratch for each study: hard to verify, harder to hand over, and structurally unable to produce the coherent three-statement picture that serious financial analysis requires. Planners working on national clean-cooking transitions had the techno-economic tools. What they did not have was a way to connect that technical layer to a rigorous, reproducible financial one.

CC-WBT was built to make that connection. At its core is a computation engine that reads every financial formula from a declarative configuration file, *formulas_map.json*, and expands those formulas automatically across the full space of active countries, techno-economic models and fuel markets. No manual replication. No separate model per scenario. A single formula entry in the configuration generates as many concrete instances as the deployment requires, all executed consistently and automatically on the next computation pass.

The financial model that runs on top of that engine goes beyond a standard project-finance template. It combines the three classic financial statements, profit and loss, balance sheet and cash flow, as a chained system where net income flows into retained earnings, depreciation reduces asset book values, and debt schedules evolve with grace periods and amortisation tenors that the planner configures directly. On top of that it adds the regulatory layer that clean-cooking deployments actually need: the Regulatory Asset Base, the Annual Cost of Service, and the long-term subsidy that closes the gap between what tariffs recover and what the service actually costs. Those quantities are not add-ons. They are the outputs that planners and regulators need to see, and they are computed here for the first time in an open, reproducible platform.

The circular dependency at the heart of the cost-of-service calculation, where ACoSt drives the subsidy, the subsidy enters revenues, revenues determine taxes, and taxes feed back into ACoSt, required its own solution. A Gauss-Seidel iterative solver resolves the loop within the existing declarative engine, without symbolic algebra and without any structural changes to how formulas are specified. In every scenario tested for Rwanda, it converged in two to three iterations with residuals of order 10^{-13}, at the practical limit of double-precision arithmetic.

The Rwanda case study put all of this to the test under realistic conditions. Three scenarios, Baseline, CleanStep and Aligned, ran across two active fuel markets and

twelve planning years. The balance sheet identity held at every period within floating-point noise of order 10^{-10} M$. Every formula was validated against the NICCP reference model developed by the IIT–SEforALL team, variable by variable and year by year. The platform passed that validation and has since been adopted by SEforALL for active use in clean-cooking financial planning across multiple countries. That is not a test result. It is a production deployment, and it is the most meaningful confirmation the platform could have received.

The Summary Financing dashboard completed the picture on the user-facing side. Rather than navigating into individual scenario views, the planner sees all scenarios side by side through a set of interactive charts, each one exposing a different dimension of the comparison: capital structure, CAPEX profile, emissions trajectory. The financial consequences of deployment pace, capital structure and tariff design can be read directly from a chart rather than reconstructed from tables.

From a broader perspective, CC-WBT is also a contribution to accessibility. Regulatory and financial modelling for energy access has historically been expert territory, the kind of work that requires both deep domain knowledge and the patience to maintain a complex spreadsheet under pressure. Moving that logic into a configurable, open-source platform that any technically informed team can inspect, extend and reproduce without developer involvement lowers a barrier that has kept clean cooking planning disconnected from the financial layer for too long. The platform is openly available at `https://github.com/SEforALL-IEAP/CC-WBT`.

The project connects to four of the United Nations Sustainable Development Goals. SDG 7 (Affordable and Clean Energy) is the most direct: the platform exists to support the financial planning of transitions that bring modern energy services to populations currently without them. SDG 3 (Good Health and Well-Being) and SDG 5 (Gender Equality) enter through the nature of the transition itself, since replacing solid-fuel combustion reduces the household air pollution and time burden that fall disproportionately on women and children. SDG 13 (Climate Action) is addressed through the carbon credits module, which quantifies avoided emissions against the Baseline scenario and estimates what those reductions could generate in carbon markets.

6.2 Limitations

Any honest account of the platform has to acknowledge where it currently falls short.

The most practical limitation is data availability. The techno-economic inputs, annual demand, CAPEX, OPEX and depreciation schedules per fuel market and per year, require careful preparation before they can be loaded into the platform. In the Rwanda case, this was done by hand, drawing on the existing IIT–SEforALL model and national planning documents. For a new country with no prior model, assembling those inputs remains a substantial manual effort that the platform does not yet automate.

Relatedly, the upload workflow currently depends on structured Excel templates that the planner fills in and feeds back. This decouples data entry from computation in a useful way, but it also means the platform inherits whatever errors or gaps exist in the input files. There is no automated validation layer that would catch structurally correct but financially implausible inputs. A CAPEX figure entered in the wrong units,

for instance, would propagate silently through the model.

The financial model is also deterministic. It takes inputs as given and produces outputs without any uncertainty quantification. A planner who wants to understand how sensitive the cost of service is to variations in the discount rate, or how the balance sheet shifts if CAPEX comes in 20 % over budget, has to run those variants manually as separate scenarios. A built-in sensitivity or scenario explorer would make that analysis considerably faster and more systematic.

Finally, validation has so far been limited to one country. Rwanda was a well-chosen test case: it has an existing national plan, reasonably detailed data, and a reference model to compare against. But it is a single data point. Whether the platform generalises cleanly to countries with very different regulatory environments, fuel market structures or institutional contexts remains to be confirmed.

6.3 Future work

Several directions could extend CC-WBT meaningfully from where it stands today. Some are close. Others are more speculative but worth naming.

The most impactful near-term addition would be automated data ingestion. Right now, getting a new country into the platform means populating Excel templates by hand: pulling annual demand figures from national planning documents, extracting CAPEX schedules from project appraisals, reconciling depreciation assumptions across fuel markets and years. For Rwanda, that work took weeks and relied on an existing IIT–SEforALL model to draw from. For a country with no prior model, it would take longer and be more uncertain. A data pipeline that could ingest standardised planning documents, NICCP-format inputs, IRENA databases, World Bank energy access datasets, and map them automatically to the platform's input structure would change that completely. The time from country selection to first financial results would collapse from weeks to hours. Even a partial automation that pre-fills the inputs that do not vary much across countries, standard depreciation schedules, reference tariff curves, typical non-technical loss assumptions, would already cut the setup effort by a meaningful fraction.

The second direction is sensitivity analysis, and the architecture is genuinely well suited for it. The computation engine is driven entirely by declarative configuration. Wrapping it in a parameter sweep that varies one input at a time is not architecturally complicated: it is a loop around the existing engine with a different value in the configuration at each iteration. What changes with that addition is the kind of questions a planner can ask. Not just what the cost of service is under a given set of assumptions, but how much it moves if the discount rate rises by two points, or if CAPEX comes in 20 % over budget, or if the grants tranche fails to materialise on schedule. Those are the questions that actually matter in a financing negotiation, and right now the only way to answer them is to set up a new scenario by hand for each variant. A built-in sensitivity explorer, with tornado charts showing which inputs drive the most uncertainty and scenario fans showing the distribution of outcomes across a range of assumptions, would turn CC-WBT from a deterministic planning model into a proper risk-assessment tool. That is a significant upgrade in what the platform can do for a

decision-maker.

Monte Carlo simulation is a natural extension of the same idea. Rather than varying one parameter at a time, a joint draw over the distribution of uncertain inputs, CAPEX, tariff trajectories, demand growth, carbon prices, would produce a distribution of financial outcomes rather than a single trajectory. That matters because the inputs to a clean-cooking financial model are genuinely uncertain, sometimes wildly so, and a single-point estimate of the cost of service or the long-term subsidy requirement does not capture the risk that a financing committee or a regulator actually has to manage.

The longer-term opportunity is to bring machine learning into the scenario exploration layer. The idea is roughly this: the platform already encodes, in its configuration and its outputs, a map from deployment assumptions to financial results. Trained on a large enough library of scenarios across countries, an encoder-decoder architecture could learn to run that map in reverse. A planner would specify outcomes, a target access rate, a maximum subsidy level, a carbon reduction goal, and the model would suggest the capital structure and tariff trajectory most likely to achieve them. Rather than exploring the space of scenarios by hand, the planner would arrive at a shortlist of viable configurations and explore from there. This remains genuinely exploratory. The training data does not yet exist at the scale needed, and the model would have to learn to respect the regulatory and financial constraints that the platform enforces. But the platform's clean separation between configuration and computation makes it a more natural substrate for this kind of integration than a monolithic spreadsheet would ever be.

The most immediate priority, though, is simply more countries. Rwanda was a rigorous test case, but it is one data point. A second country with a different regulatory tradition, a different debt market, a different mix of fuel markets, would surface assumptions that Rwanda cannot. It would also reveal whether the platform's input structure is genuinely general or whether it has Rwanda-specific quirks baked in that nobody noticed because there was nothing to compare against. A third country would start to look like a library: open, reproducible financial models for clean-cooking planning that any team anywhere can download, inspect and build on. That accumulation of validated, comparable models is, in itself, one of the things the project was trying to contribute. The platform is ready for it.

Bibliography

[1] F. de Cuadra, P. Dueñas, E. Sánchez-Jacob *et al.*, "Models and Tools for Integrated Clean Cooking Planning: Case Example of Rwanda," IIT–MIT Working Document, 2024.

[2] R. Amatya *et al.*, "Computer-aided electrification planning in developing countries: The Reference Electrification Model (REM)," IIT Working Paper 18-112-A, Universidad Pontificia Comillas, 2018.

[3] S. J. Díaz-Pastor and I. J. Pérez-Arriaga, "An integrated regulatory–financial proposal for universal electrification in Uganda," *Energy Economics*, Elsevier, 2025.

[4] I. J. Pérez-Arriaga, *Regulation of the Power Sector*. London: Springer, 2014.

[5] World Health Organization, "Household air pollution and health," WHO Fact Sheet, 2024. Available: `https://www.who.int/news-room/fact-sheets/detail/household-air-pollution-and-health`

[6] World Bank, "The State of Access to Modern Energy Cooking Services," World Bank Group, Washington, DC, 2020.

[7] International Energy Agency, "World Energy Outlook," IEA Publications, Paris, 2023.

[8] A. González-García, S. J. Díaz-Pastor, and A. Moreno-Romero, "A Comprehensive Approach to the Governance of Universal Access to Sustainable Energy," *Sustainability*, vol. 15, no. 22, p. 15813, 2023.

Appendix A

Appendix I — Supplementary Material

This appendix collects reference material complementary to the main body of the thesis: a glossary of the financial terms used throughout the document, a step-by-step user guide for the CC-WBT platform, and an extended selection of `formulas_map.json` entries illustrating the range of operations the declarative engine can express.

A.1. Glossary of Financial Terms

Term	Definition
ACoSt *Annual Cost of Service*	Total annual cost of providing the energy service. Includes the return on the RAB (WACC × RAB), operating costs (OPEX), depreciation and amortisation (D&A), changes in working capital, and taxes. It is the quantity that determines how much long-term subsidy is needed when tariff revenues fall short.
CAPEX *Capital Expenditure*	Investment in capital assets. Split into *Growth CAPEX* (capacity expansion to meet growing demand) and *Maintenance CAPEX* (replacement or upgrade of existing assets to sustain operations). Total CAPEX in a given period is the sum of both.
D&A *Depreciation & Amortisation*	Annual accounting allocation of capital assets over their useful life. It is a non-cash expense that reduces earnings before tax but involves no cash outflow.
Debt	External financing raised through loans. It generates periodic obligations: interest payments on the outstanding balance and principal repayments according to the defined amortisation schedule (grace period plus repayment period).
EBITDA	Earnings before interest, taxes, depreciation and amortisation. Measures the operational profitability of the project: total revenues minus upstream costs, OPEX and bad-debt provisions. Also computed as *ex-Grants* to reflect performance without external support.

Term	Definition
EBT *Earnings Before Tax*	Pre-tax profit. Equals EBITDA minus D&A and financial expenses. The platform also computes it excluding grants and subsidies (*ex-Grants, ex-Subsidies*) to assess viability without public support.
LTS *Long-Term Subsidies*	The long-term subsidy required to close the gap between ACoSt and tariff revenues. It quantifies the minimum public support that makes a deployment strategy financially viable, and it participates in the model's circularity alongside ACoSt and taxes.
Non-technical losses	Revenue losses not attributable to technical failures, but to non-payment, theft or commercial inefficiencies. Modelled as a percentage of tariff revenues; they generate a bad-debt provision in the income statement.
OPEX *Operating Expenditure*	Recurring operating costs: fuel procurement, logistics, labour and routine maintenance. They affect cash and profitability directly but do not create assets.
PP&E *Property, Plant & Equipment*	Net book value of physical assets in use at the end of the period. On the balance sheet it is the main component of non-current assets and evolves with CAPEX additions and accumulated depreciation.
RAB *Regulatory Asset Base*	The net value of assets in service on which the investor's return is calculated. It evolves each period as $\mathrm{RAB}(t) = \mathrm{RAB}(t-1) + \text{Total CAPEX}(t) - \mathrm{D\&A}(t)$. It is the central quantity in the cost-of-service framework.
Working Capital	The difference between current assets (trade receivables, inventories, cash) and current liabilities (payables). Changes in working capital affect operating cash flow even though they do not appear directly in the income statement.
WACC *Weighted Average Cost of Capital*	The blended cost of the capital financing the project. Combines the cost of equity and the after-tax cost of debt, weighted by their shares in the capital structure: $\mathrm{WACC} = \frac{E}{E+D}\, r_E + \frac{D}{E+D}\, r_D\,(1-\tau)$.
Viability gap	The difference between ACoSt and the revenues the market can sustainably generate. When positive, the project requires subsidies or concessional financing to be viable without raising tariffs beyond users' ability to pay.

A.2. CC-WBT User Guide

This section describes the standard workflow in CC-WBT, from starting the platform to viewing the financial statements.

Step 1 — Starting the platform. CC-WBT requires two processes running simultaneously: the FastAPI backend and the Streamlit frontend. In two separate terminals, inside the repository virtual environment:

```
# Terminal 1 - backend
cd CC-WBT && venv/Scripts/activate
cd backend && uvicorn main:app --reload --port 8001

# Terminal 2 - frontend
cd CC-WBT && venv/Scripts/activate
cd frontend && streamlit run app.py
```

Listing A.1: Start-up commands for CC-WBT

Once both processes are running, the interface is available at `localhost:8501`.

Step 2 — Welcome screen. The home screen presents the *Select Scenario* button, which opens the country management page (Figure A.1).

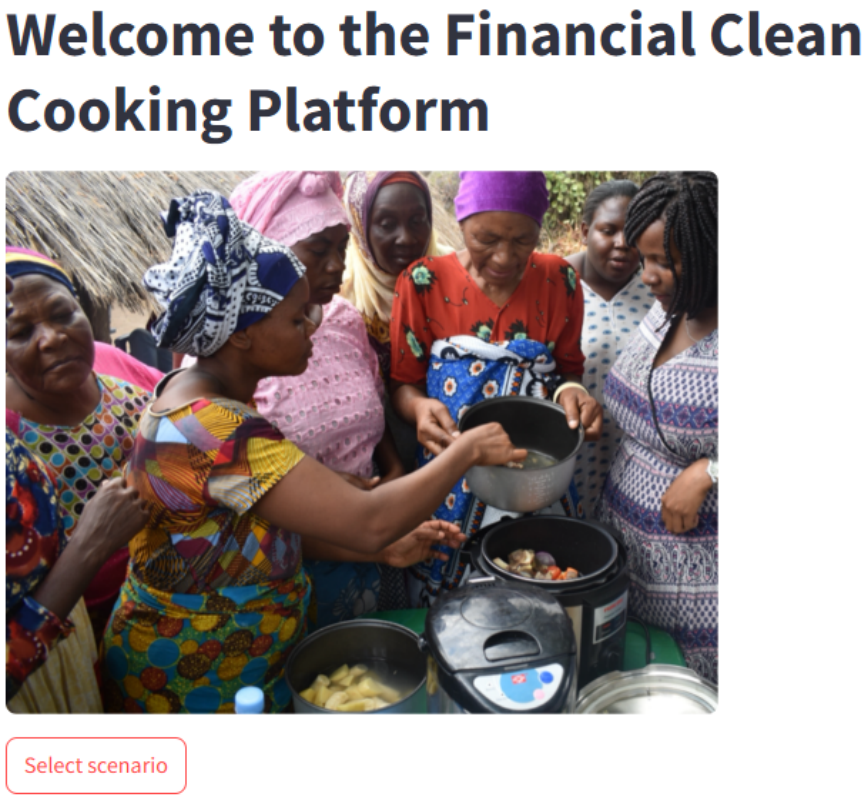


Figure A.1: CC-WBT welcome screen.

Step 3 — Country and model selection. The country page lists all available scenarios. Each card allows the user to download input files, delete the scenario, or enter the modelling workflow via *Start Modeling*. New countries (name, tax rate, inflation) and new techno-economic models (name, start year, end year) can also be created from here.

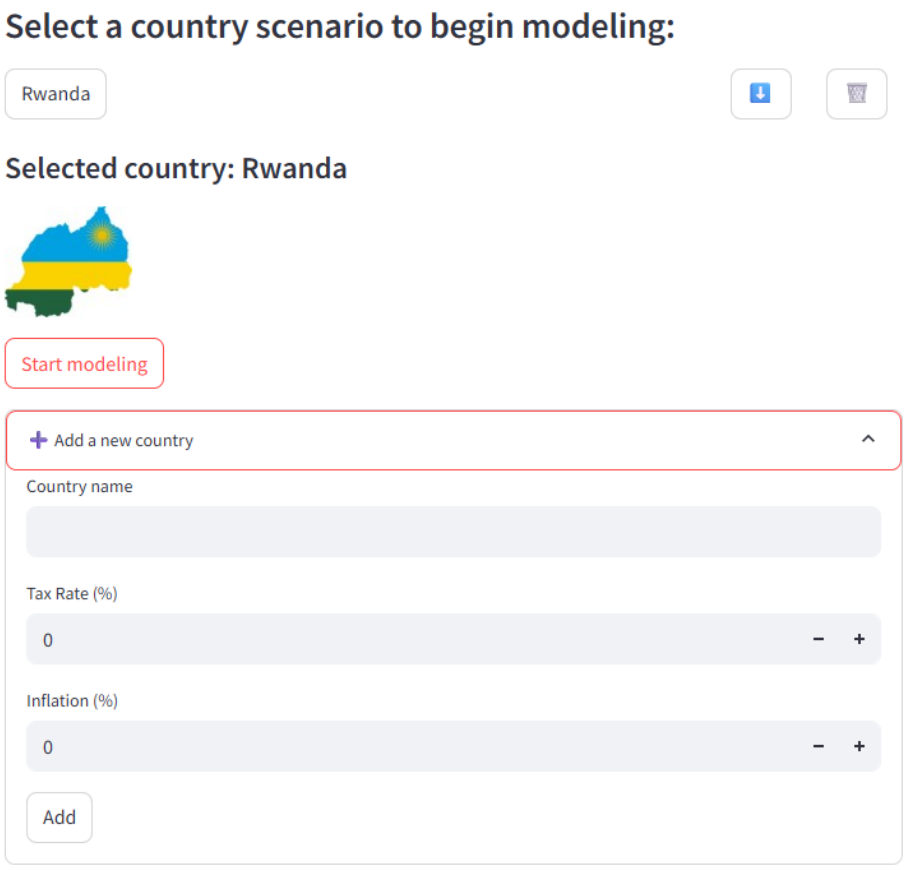


Figure A.2: Country selection page with available models.

Step 4 — Techno-economic inputs. Inside a model, the *Inputs* section displays the physical variables for each fuel market: demand, growth CAPEX, maintenance CAPEX, OPEX and D&A, broken down by year. Values are loaded via the Excel template downloadable from the model card.

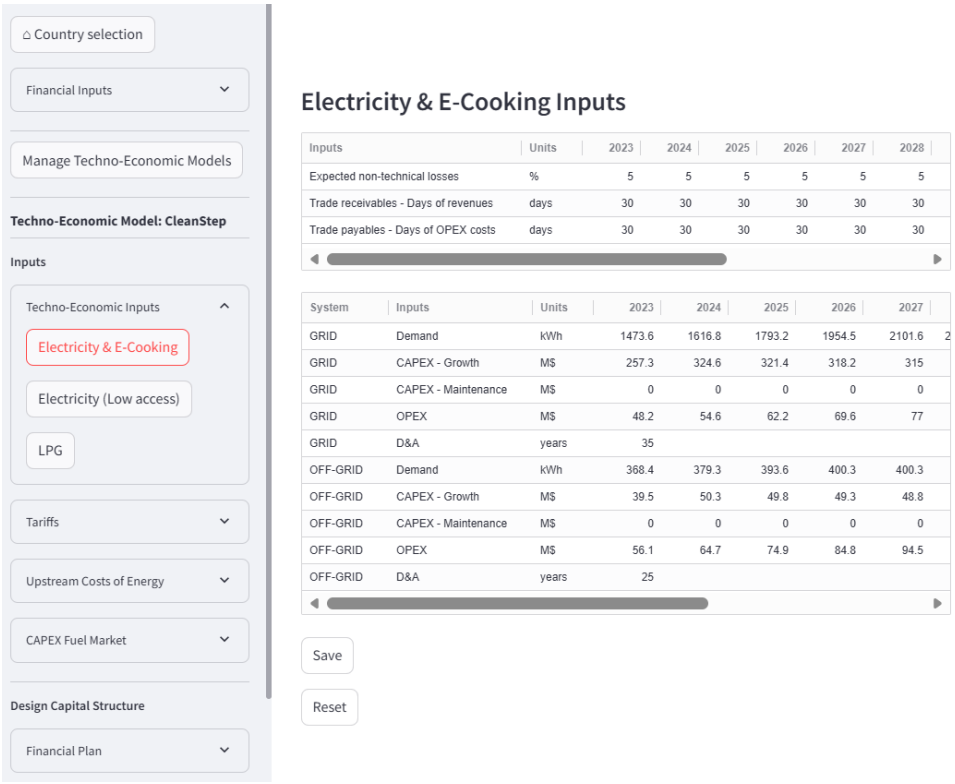


Figure A.3: Techno-economic inputs view for a fuel market.

Step 5 — Financial inputs. The *Financial Inputs* sidebar gives access to the financial assumptions for each market: tariff, upstream energy cost, non-technical losses, and carbon credit parameters. All input pages include save and reset buttons.

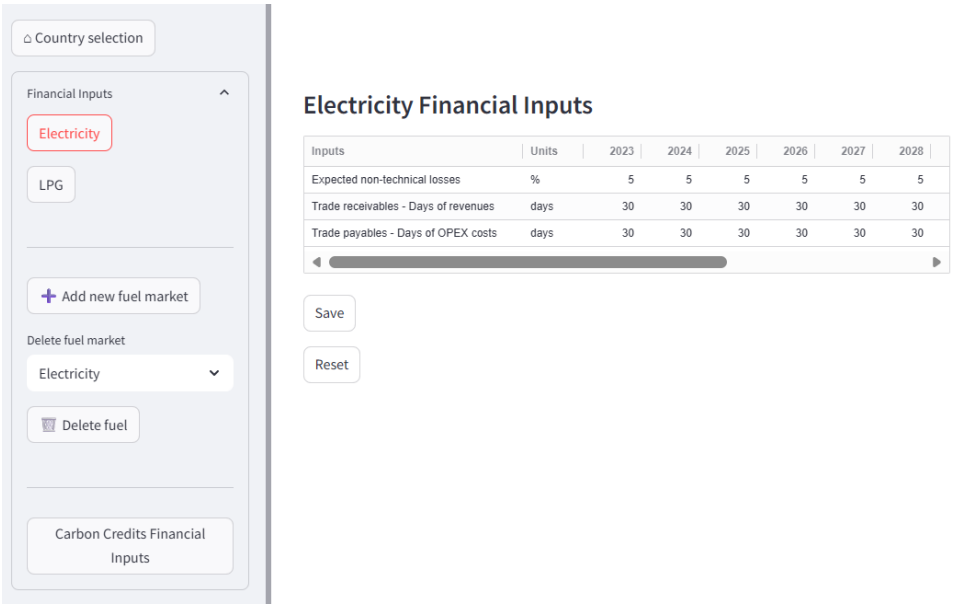
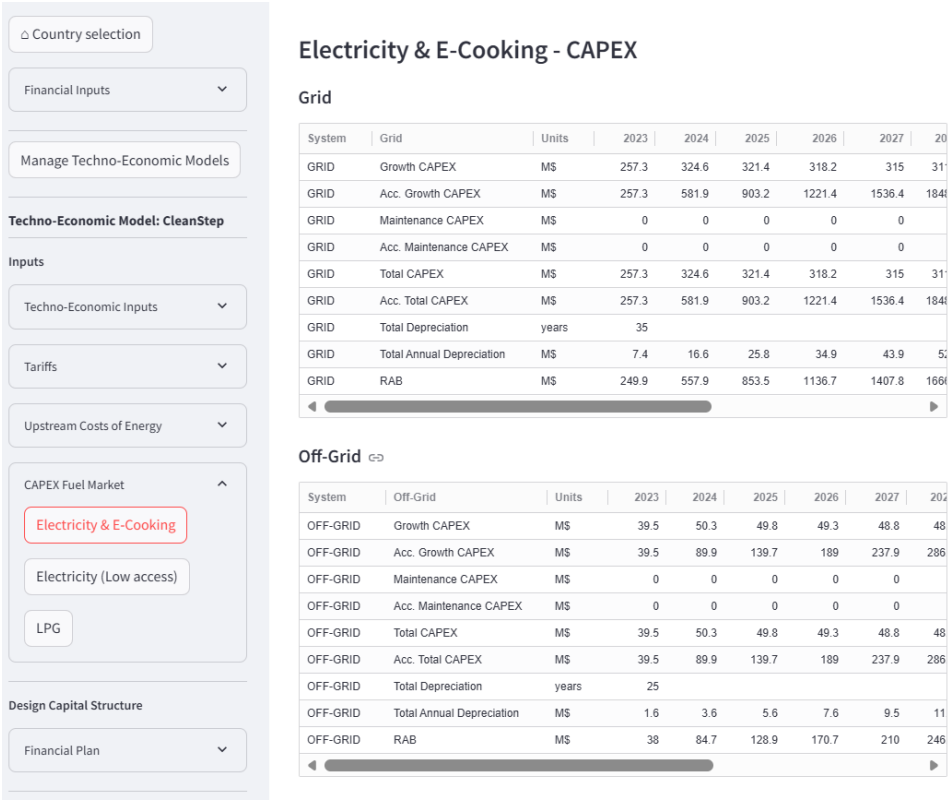


Figure A.4: Financial inputs page for a fuel market.

Step 6 — CAPEX and RAB. The CAPEX section displays the annual evolution of the asset base: growth and maintenance CAPEX, accumulated CAPEX, annual depreciation, and RAB — all computed automatically from the techno-economic inputs.



System	Grid	Units	2023	2024	2025	2026	2027	20
GRID	Growth CAPEX	M$	257.3	324.6	321.4	318.2	315	31
GRID	Acc. Growth CAPEX	M$	257.3	581.9	903.2	1221.4	1536.4	184
GRID	Maintenance CAPEX	M$	0	0	0	0	0	
GRID	Acc. Maintenance CAPEX	M$	0	0	0	0	0	
GRID	Total CAPEX	M$	257.3	324.6	321.4	318.2	315	31
GRID	Acc. Total CAPEX	M$	257.3	581.9	903.2	1221.4	1536.4	184
GRID	Total Depreciation	years	35					
GRID	Total Annual Depreciation	M$	7.4	16.6	25.8	34.9	43.9	5
GRID	RAB	M$	249.9	557.9	853.5	1136.7	1407.8	166

System	Off-Grid	Units	2023	2024	2025	2026	2027	202
OFF-GRID	Growth CAPEX	M$	39.5	50.3	49.8	49.3	48.8	48
OFF-GRID	Acc. Growth CAPEX	M$	39.5	89.9	139.7	189	237.9	286
OFF-GRID	Maintenance CAPEX	M$	0	0	0	0	0	
OFF-GRID	Acc. Maintenance CAPEX	M$	0	0	0	0	0	
OFF-GRID	Total CAPEX	M$	39.5	50.3	49.8	49.3	48.8	48
OFF-GRID	Acc. Total CAPEX	M$	39.5	89.9	139.7	189	237.9	286
OFF-GRID	Total Depreciation	years	25					
OFF-GRID	Total Annual Depreciation	M$	1.6	3.6	5.6	7.6	9.5	11
OFF-GRID	RAB	M$	38	84.7	128.9	170.7	210	246

Figure A.5: CAPEX and RAB table for a fuel market.

Step 7 — Capital structure. *Design Capital Structure* captures the financing assumptions: equity, debt and grant shares, associated costs, and the realisation and amortisation schedules. From these inputs the platform computes the WACC and determines the debt requirement for each period.

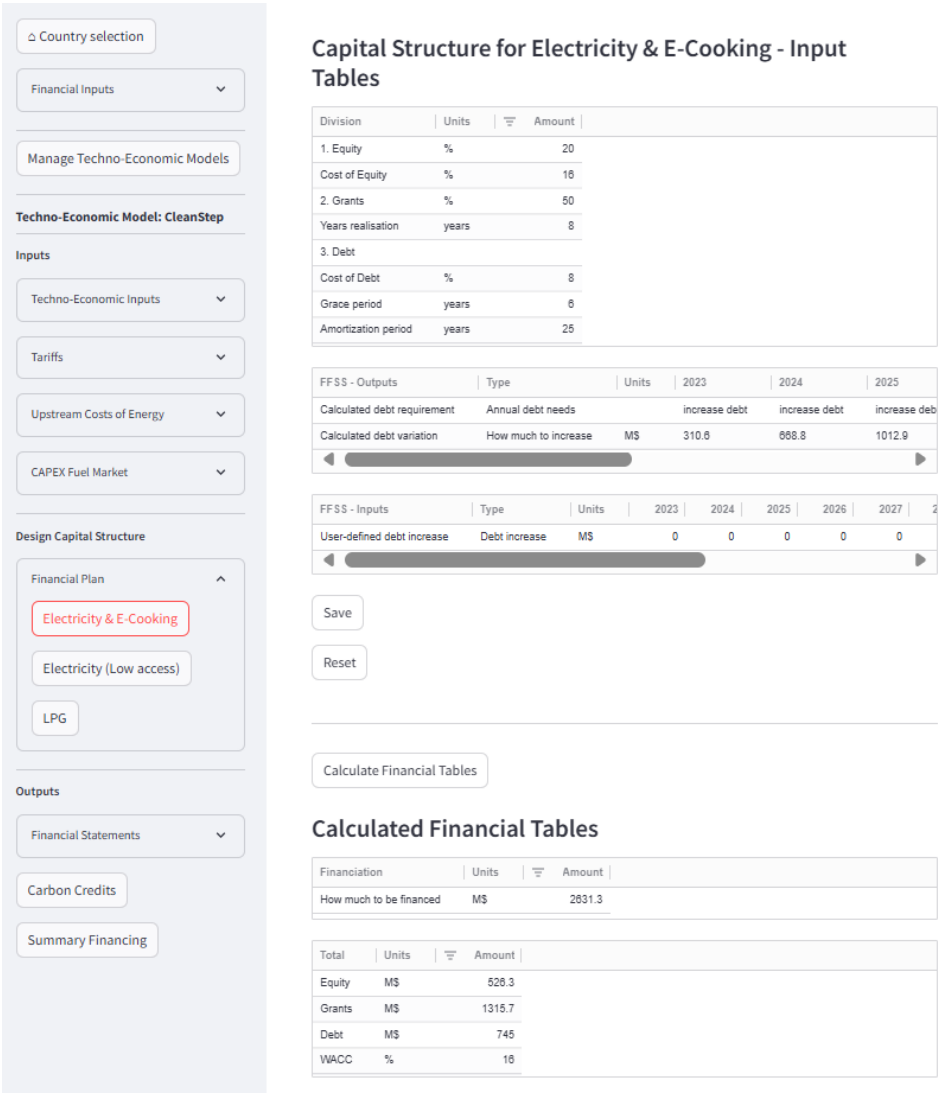


Figure A.6: Capital structure design page.

Step 8 — Financial statements. The *Outputs* → *Financial Statements* section presents the three chained statements — income statement, balance sheet and cash flow statement — for each fuel market and year. The balance sheet check appears in the last row and must equal zero in every period.

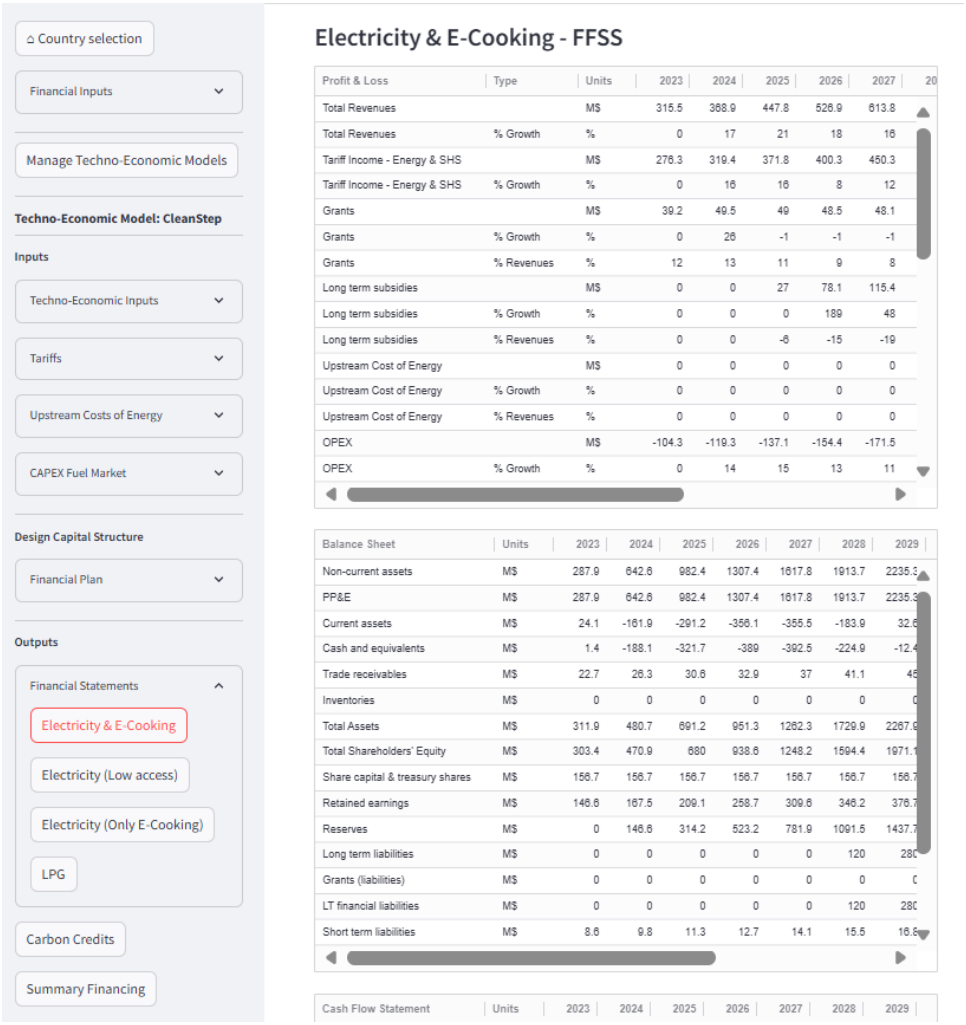


Figure A.7: Integrated financial statements view.

Step 9 — Summary dashboard. The dashboard aggregates results across all markets and scenarios in a single interactive panel: RAB, ACoSt, LTS, cash flow and profitability indicators, with the option to export the underlying data.

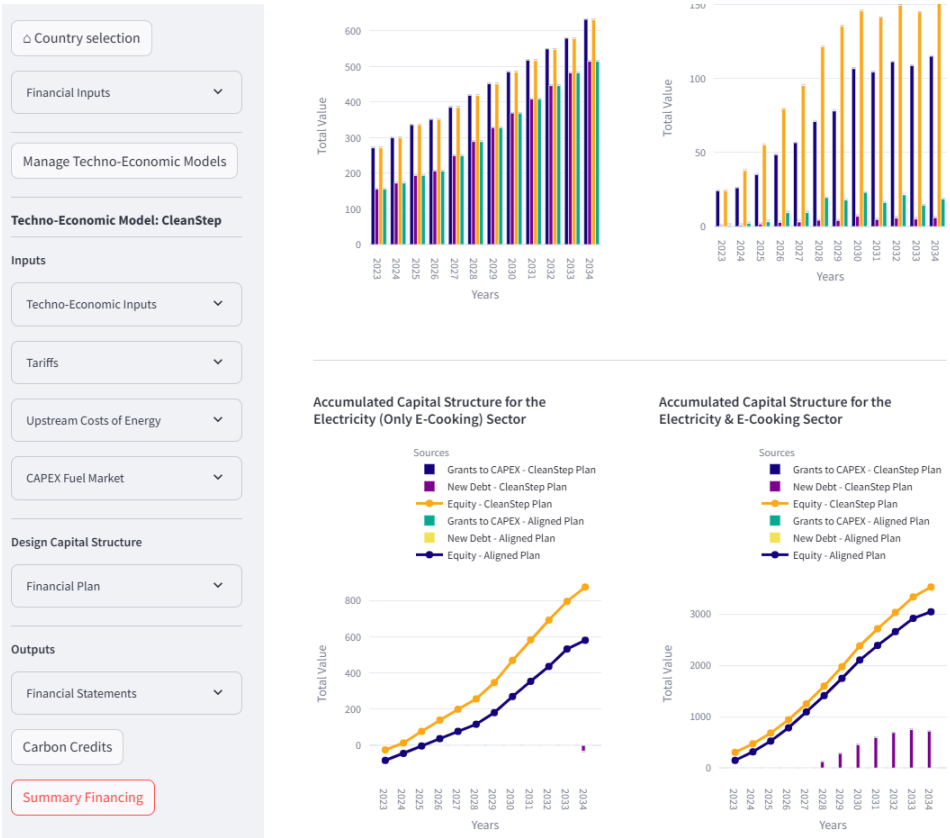


Figure A.8: Aggregated results dashboard by scenario.

A.3. Selected `formulas_map.json` Entries

The `formulas_map.json` file defines all financial logic in CC-WBT declaratively. Each entry specifies a `target` (the variable to compute), a `sources` array (paths to the origin cells, parametrised with `{country}`, `{model}` and `{fuel}`), and a `formula_steps` sequence that transforms the sources into the final result. Three entries are shown below in order of increasing complexity.

Entry 1 — Simple arithmetic: Total CAPEX

The simplest pattern combines two sources with a single arithmetic operation. Total CAPEX in each period is the sum of growth and maintenance CAPEX:

```
{
  "target": "Total CAPEX",
  "sources": [
    "{country}\\capex-fuels-{model}.xlsx::{sheet}::Growth CAPEX",
    "{country}\\capex-fuels-{model}.xlsx::{sheet}::Maintenance CAPEX"
```

```
  ],
  "formula_steps": [
    {
      "op": "addition",
      "operands": [["index", 0], ["index", 1]],
      "result": "final"
    }
  ]
}
```

Listing A.2: Total CAPEX – addition of two sources

Entry 2 — Temporal accumulation: accumulated CAPEX

Accumulated CAPEX requires summing all past values of a series. The `sum_range` operator with `direction: backward` performs this accumulation from the start of the horizon up to the current period:

```
{
  "target": "GRID:Acc. Growth CAPEX",
  "sources": [
    "{country}\\capex-fuels-{model}.xlsx::{sheet}::GRID:Growth CAPEX"
  ],
  "formula_steps": [
    {
      "op": "sum_range",
      "operands": [["index", 0], ["literal", 0]],
      "direction": "backward",
      "result": "final"
    }
  ]
}
```

Listing A.3: Acc. Growth CAPEX – backward temporal accumulation

Entry 3 — Conditional logic: certifiable avoided CO_2

The most complex entry computes the CO_2 avoided that can be certified and sold in carbon markets, respecting the eligibility window defined by the user. It requires seven intermediate steps: building a time index, comparing it against the eligibility limit, conditional accumulation, applying the certification percentage, and clamping to zero for years outside the window.

```
{
  "target": "CO2 equivalent avoided in the {model}",
  "sources": [
    "{country}\\fuel-financial-inputs.xlsx::Carbon Credits::Number of
       years that you could sell those carbon credits",
    "{country}\\carbon-credits.xlsx::Carbon Credits::{model} vs.
       Baseline",
    "{country}\\fuel-financial-inputs.xlsx::Carbon Credits::CO2
       certificate (%) - from the total CO2 avoided"
  ],
```

```
  "formula_steps": [
    { "op": "range",        "operands": [["literal", 1]],
      "result": "temp1" },
    { "op": "value",        "operands": [["index", 0], ["literal",
       0]],
      "result": "temp2" },
    { "op": "gt",           "operands": [["index", "temp1"], ["index",
       "temp2"]],
      "result": "temp3" },
    { "op": "sum_range",    "operands": [["index", 1], ["index", "
       temp2"]],
      "direction": "backward",
      "result": "temp4" },
    { "op": "if",           "operands": [["index", "temp3"], ["index",
       "temp4"], ["index", 1]],
      "result": "temp5" },
    { "op": "multiply_per", "operands": [["index", "temp5"], ["index",
       2]],
      "result": "temp6" },
    { "op": "max",          "operands": [["index", "temp6"], ["literal
       ", 0]],
      "result": "final" }
  ]
}
```

Listing A.4: Certifiable avoided CO2 – seven-step conditional chain

The three entries show how the same declarative schema scales from a single arithmetic operation to a chain with conditional flow control and temporal accumulation, without any modification to the computation engine.

GARRIDO GARCIA-PITA ALMUDENA - 54210620B

Firmado digitalmente por GARRIDO GARCIA-PITA ALMUDENA - 54210620B
Fecha: 2026.05.25 22:01:02 +02'00'