AN INTELLIGENT MICROGRID MANAGEMENT AND OPTIMIZATION SYSTEM: AN EXPERT ANALYTICAL SYSTEM FOR REAL TIME OPTIMIZATION AND INTEGRATION OF RENEWABLE ENERGY USING LIVE WEATHER DATA

BY

BOYI KENNETH EFE

ENG2010796

SUPERVISOR:

ENGR. DR. OBAYUWANA AUGUSTINE

A PROJECT SUBMITTED

TO

DEPARTMENT OF COMPUTER ENGINEERING
FACULTY OF ENGINEERING
UNIVERSITY OF BENIN

IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF A BACHELOR'S DEGREE IN COMPUTER ENGINEERING

OCTOBER, 2025

CERTIFICATION

This is to certify that the project titled "**AN INTELLIGENT MICROGRID MANAGEMENT AND OPTIMIZATION SYSTEM: AN EXPERT ANALYTICAL SYSTEM FOR REAL TIME OPTIMIZATION AND INTEGRATION OF RENEWABLE ENERGY USING LIVE WEATHER DATA**" was carried out and duly presented by **BOYI KENNETH EFE (ENG2010796)** of the Department of Computer Engineering, Faculty of Engineering, University of Benin, Benin City, in partial fulfillment of the requirements for the award of the Bachelor of Engineering (B.Eng.) degree in Computer Engineering.

_____ _____

Engr. Dr. Augustine Obayuwana **Date**
(Project Supervisor)

_____ _____

Engr. Dr. I.A. Edeoghon **Date**
(Head of Department)

DEDICATION

This work is first dedicated to the almighty God, creator of all things who has always been faithful. Also, to my parents Mr. and Mrs. BOYI for their support and love.

ACKNOWLEDGEMENT

My sincere thanks goes to **Engr. Dr. AUGUSTINE OBAYUWANA,** my project supervisor who I appreciate for the understanding, support and guidance that he provided throughout the preparation of this project work. I also appreciate the Head Of Department, Computer Engineering, University Of Benin, **Engr. Dr. I. A. Edeoghon,** as well as the entire staff and lab tech who have all contributed to the knowledge I have obtained over the past 5 years from this institution.

I also wish to acknowledge my course mates and group members who assisted me greatly when the strain of the project work became unbearable.

I wish to honour my Late Father Engr. Bernard Boyi and my Mother Mrs. Juliet Boyi and siblings Irene, Dennis, Emmanuel and Simon for their love and support during my stay in school and for my project, and to my covenant family RCF, I want to say a very big thank you for your love and support.

I pray God blesses you in all things. Amen.

TABLE OF CONTENTS

TABLE OF FIGURES

ABSTRACT

As the world continues to embrace cleaner and smarter energy solutions, there's a growing need for tools that not only design microgrids but also make them smarter, more responsive, and easier to manage. This project introduces an Intelligent Microgrid Management and Optimization System — a desktop application built with Python — designed to help users plan, optimize, and monitor solar-powered microgrid systems more efficiently.

What sets this tool apart is its ability to pull live weather data (like sunlight levels and temperature) using the OpenWeatherMap API. With this, it can predict how much energy your solar panels might generate and how much power you'll need, thanks to built-in machine learning models. The system then uses a genetic algorithm to figure out the best combination of solar panel size and battery capacity to meet your energy needs while keeping costs low.

The application runs through a simple and responsive user interface (built with PyQt6), offering features like real-time graphs, a weather dashboard, and system control panels. It also supports SCADA-style monitoring, so users can see power generation, battery status, and energy demand in real time. Overall, this tool is designed to be both smart and user-friendly, making it useful not just for engineers and developers, but also for students, researchers, and organizations working on renewable energy solutions.

CHAPTER 1
INTRODUCTION

1.1 BACKGROUND OF THE STUDY

A microgrid is a localized energy system that operates independently or in conjunction with the main power grid. It consists of distributed energy resources, such as solar panels, wind turbines, and energy storage systems, that generate and store electricity (Belrzaeg, M., et al, 2023; Basak, P., et al, 2012 and Al-Ghussain, L.,et al, 2020). These sources are connected to a control system that manages the flow of energy within the microgrid (Majumder, R.,et al,2009 and Elmouatamid, A., 2020).

Recently, the increasing global demand for sustainable and resilient energy systems has underscored the significance of microgrids localized energy networks capable of operating independently or in coordination with the main power grid (Kostenko, G. and Zaporozhets, A., 2023). Comprising distributed energy resources (DERs) such as solar panels, wind turbines, and energy storage systems, microgrids facilitate decentralized electricity generation and efficient energy management (Twaisan, K. and Barışçı, N., 2022.). These systems are governed by intelligent control mechanisms that regulate power flow, ensuring energy stability and reliability within localized areas such as individual buildings, neighborhoods, or communities (Annaswamy, A.M. and Amin, M., 2013; and Sun, M., 2025.).

Microgrids offer numerous benefits over conventional centralized grids (Pires, V.F., Pires, A. and Cordeiro, A., 2023 and Abu-Elzait, S. and Parkin, R., 2019). They provide a high degree of resilience, maintaining power supply during grid outages and natural disasters, especially for critical infrastructures like hospitals and emergency response centers. Moreover, by generating electricity close to the point of consumption, microgrids enhance energy efficiency, reduce transmission losses, and lower greenhouse gas emissions (Committee on Enhancing the Resilience of the Nation's Electric Power Transmission and Distribution System, 2017). Their architecture supports the integration of renewable energy sources, promoting environmental sustainability and reducing dependence on fossil fuels. Additionally, microgrids enable cost savings by minimizing energy transportation costs and facilitating the use of free, renewable resources. They also foster grid independence,

empowering communities to exercise greater control over their energy systems (Mottahedi, A., et al, 2021).

Despite these advantages, the integration and optimization of renewable energy sources-particularly solar energy into microgrid systems pose significant technical challenges. The variability and intermittency of renewable generation, site-specific environmental conditions, and fluctuating energy demands complicate the effective synchronization, phase alignment, and load balancing of microgrids with existing utility grids. Without precise system design and configuration, these issues can lead to inefficiencies, energy wastage, and increased operational costs, ultimately hindering the broader adoption of microgrids (Singh, S. and Singh, S., 2024; and Faisal, M.,et al, 2018).

Nevertheless, the design of an efficient microgrid demands a meticulous selection and coordination of critical components such as photovoltaic (PV) panels, inverters, battery systems, and load management units. Achieving optimal system performance requires addressing complex tasks, including, grid compatibility, and the dynamic optimization of energy flows based on real-time data (Agha Kassab, F.,et al,2024 and Zhang, L., et al, 2014).

To overcome these barriers, there is a compelling need for a robust intelligent microgrid design and optimization suite- A software-based expert analytical system capable of automating the design, simulation, and optimization of microgrid configurations. Such a suite would leverage artificial intelligence, optimization algorithms, and real-time analytics to model system behavior under various scenarios. By accounting for parameters like solar irradiance, geographic location, load profiles, energy storage capacity, and economic constraints, the suite would provide tailored and optimal design solutions for specific applications.

Furthermore, the proposed suite would enhance phase synchronization and ensure seamless grid integration, thereby improving the operational stability and reliability of hybrid energy systems. It would streamline the traditionally complex and time-consuming design process, reduce implementation costs, and facilitate the widespread adoption of clean energy technologies.

1.2 STATEMENT OF THE PROBLEM

The development and deployment of solar-powered microgrid systems for localized energy supply, such as those intended for a university campus in Benin City, Nigeria (6.33°N, 5.62°E), require a sophisticated approach to energy demand assessment, system optimization, and operational management. Despite the potential of solar microgrids to provide reliable, sustainable, and resilient energy, their implementation faces significant technical and operational challenges that impede their effectiveness and scalability.

A primary challenge is the lack of integrated, intelligent tools capable of leveraging real-time weather data (e.g., Global Horizontal Irradiance (GHI), Diffuse Horizontal Irradiance (DHI), and temperature) and historical datasets (e.g., NSRDB) to accurately forecast energy production and consumption. Current design processes often rely on static assumptions about solar irradiance, temperature, and load profiles, which fail to account for the dynamic variability of climatic conditions and campus-specific energy demands. This leads to suboptimal sizing of critical components, such as solar panels and battery storage, resulting in energy shortages, inefficiencies, or unnecessary costs.

Additionally, the integration of predictive modeling and artificial intelligence (AI) for real-time decision-making is limited in existing microgrid design tools. Without advanced algorithms, such as Support Vector Regression (SVR) for solar power and load forecasting, or genetic algorithms for optimizing component configurations, planners struggle to achieve cost-effective and reliable system designs. The absence of such capabilities hinders the ability to dynamically adjust to grid conditions (e.g., grid-connected vs. islanded modes) and ensure efficient battery management, particularly in preventing unnecessary discharge during grid-connected operation.

Furthermore, the lack of user-friendly, modular software interfaces exacerbates these issues by making it difficult for energy planners and campus administrators to visualize system performance, monitor real-time SCADA data, and interact with optimization results. Traditional tools often lack intuitive dashboards for displaying critical metrics (e.g., solar power output, battery state of charge (SOC), and grid

status) or the flexibility to incorporate real-time weather data from APIs (e.g., OpenWeatherMap) alongside device location-based inputs. This disconnect between data, analytics, and user interaction slows down the design process and limits stakeholder engagement.

The absence of a comprehensive, data-driven software framework that combines real-time analytics, AI-driven forecasting, and modular design interfaces restricts the ability to fully harness the benefits of solar microgrids. This not only compromises the economic viability and environmental impact of such systems but also delays the adoption of decentralized, renewable energy solutions in regions like Benin City, where energy reliability is critical for educational institutions.

Therefore, there is an urgent need for an intelligent software design tool tailored for solar microgrid development, integrating expert system principles, real-time weather data, and advanced optimization algorithms. This tool should enable accurate demand and solar power forecasting using SVR models, optimize component sizing with genetic algorithms to ensure positive configurations, provide seamless grid integration with robust switching logic, and offer a modular GUI with interactive tabs for control, visualization, and weather monitoring. By addressing these limitations, the proposed tool aims to enhance the reliability, cost-effectiveness, and scalability of solar microgrid systems, supporting the transition to sustainable energy infrastructures in Nigeria and beyond.

1.3 AIM AND OBJECTIVES OF THE STUDY

The aim of this research is to develop an Intelligent Microgrid Management and Optimization System that utilizes an expert analytical system approach to optimize renewable energy integration and grid connectivity using real life weather data. To achieve the stated aim, the following objectives will be pursued:

i. Develop a framework and a system architecture for the implementation of the INTELLIGENT MICROGRID DESIGN AND OPTIMIZATION SUITE
ii. Integrate live weather data as an additional feature to enhance real-time monitoring.

iii. Develop computational models for solar component sizing, cost estimation, and system performance analysis
iv. Develop a user-friendly platform that enables efficient design and planning of solar energy systems.
v. Support generalized inputs for solar, wind, and geothermal energy sources.
vi. Develop a database management system
vii. Validate the developed software by testing its effectiveness using real-world case studies of microgrid installations in Nigerian campuses and national grid integration projects.

1.4 SCOPE OF THE STUDY

This research is mainly focused on creating a software tool that helps design microgrids powered by renewable energy, especially solar power. While microgrids can use many energy sources like wind, biomass, or diesel generators, this work mostly concentrates on solar photovoltaic (PV) systems. Because of this, the models and simulations in the study look closely at things like sunlight levels, battery storage, and solar energy patterns. Other energy sources are mentioned but are not explored in detail, which means the results apply best to systems that rely heavily on solar energy.
Another important limitation is that the project is entirely software-based. The research does not involve building or testing a physical microgrid in the real world. Instead, the design tool is tested through computer simulations, existing datasets, and case studies. This means that while the tool may work well in theory, it has not been tested under actual operating conditions. As a result, some differences may appear when it is used in real-life situations.
The study is also limited by the type and amount of data available. High-quality data on solar radiation, electricity use, and battery performance is not always easy to get, especially in developing regions. Because of this, the tool sometimes has to rely on estimated values or general assumptions. This may affect the accuracy of the predictions the tool produces, since real-world systems can behave differently from estimated models.
Although the tool could be used anywhere in the world, the study mainly focuses on Nigerian conditions. The examples, case studies, and environmental factors used in the research are based on Nigerian locations and grid systems. This means the tool may need some adjustments to work well in places with very different weather,

electricity networks, or regulations. The results, therefore, may not apply perfectly to every region without some customization.

Another limitation is that the research focuses mostly on technical and economic issues, such as how to size microgrid components, how to improve performance, and how to reduce cost. It does not go into other important topics like environmental impacts, community acceptance, or legal issues related to microgrid development. These areas are important for real-life energy planning but were intentionally left out to keep the study focused.

The software itself is also limited in terms of what it can do. It is designed to help with planning and design, not with running or controlling a real microgrid. It does not include hardware control systems, advanced security features, or real-time monitoring tools that would be needed for full microgrid operation. This means the tool is most useful for designing microgrids, not for managing them after installation.

Lastly, the intelligent features in the tool are meant for predictions and planning, not for real-time decision-making. While it includes some machine learning methods, it does not handle tasks like detecting faults, responding to emergencies, or making automatic adjustments during operation. Those advanced features are outside the scope of this project. So, the tool is best used during the early stages of microgrid development, before the system is built.

1.5 RELEVANCE OF THE STUDY

The development of an Intelligent Microgrid Software Design Tool offers major benefits across the renewable energy sector, especially as global energy systems shift toward cleaner and more decentralized solutions. Many stakeholders increasingly require tools that simplify microgrid design while improving accuracy, resilience, and cost efficiency. This research meets these needs by creating a platform that combines expert analytical methods, optimization algorithms, and real-time data analytics, providing a strong foundation for better decision-making and energy planning.

Researchers and academic institutions benefit greatly from this tool because it gives them access to rich datasets and realistic simulations that were previously difficult to obtain. By integrating information such as solar irradiance, load demand patterns, battery performance, and grid interaction behavior, the tool helps researchers conduct more accurate studies.

Universities can adopt the tool as a teaching resource, giving students practical experience with energy system simulations and helping them connect theoretical knowledge to real-world applications.

Policymakers also gain from this research, as effective energy policies must be based on reliable analysis and predictive modeling. The tool allows policymakers to simulate various regulatory scenarios, evaluate the effect of renewable energy on grid stability, and estimate potential reductions in carbon emissions. These insights support the creation of well-informed policies that improve energy security, promote sustainability, and guide the safe deployment of microgrids especially in countries where technical capacity may be limited.

For engineers, consultants, and energy developers, the tool simplifies the complex process of microgrid design by automating tasks such as component sizing, load forecasting, and phase synchronization. This reduces errors, speeds up project development, and improves the reliability and performance of microgrid configurations. The user-friendly interface also makes the tool accessible to both technical and non-technical users.

Investors and private developers benefit through the tool's detailed techno-economic analysis capabilities. It allows them to evaluate project costs, risks, savings, and return on investment using accurate simulations. This helps investors make informed decisions, reduces uncertainty, and encourages greater investment in renewable energy projects.

CHAPTER 2

LITERATURE REVIEW

2.1 REVIEW OF RELATED WORKS

Recent advancements in renewable energy microgrid design tools have enhanced system planning and operation, yet several limitations persist for adaptive, multi-source systems. A key challenge is the lack of real-time data integration in most existing tools. Many solutions rely on static inputs, such as historical load profiles or average environmental conditions, which fail to capture dynamic variations in energy demand, production, and system performance. This reduces the accuracy of forecasting and optimization, impacting system reliability and efficiency (Olatunde & Adejumobi, 2023).

Furthermore, few tools offer a comprehensive integration of technical performance optimization and financial feasibility analysis within a single platform. While some solutions provide cost estimation or energy efficiency calculations, they often lack the ability to simultaneously optimize component sizing, grid connectivity, and economic viability, making it difficult for users—particularly institutions like universities—to make informed design decisions (Akinyele & Rayudu, 2020).

User accessibility remains another significant gap. Most microgrid design tools are tailored for engineers or industry professionals, featuring complex interfaces that require technical expertise. This limits their adoption by non-expert stakeholders, such as campus energy managers or administrative staff, who need intuitive platforms to design and manage renewable energy systems (Iweh et al., 2022). There is a pressing need for a tool that balances technical sophistication with user-friendliness, catering to both technical and non-technical users in microgrid optimization and grid integration.

Several studies have explored renewable energy system design, particularly for solar-based microgrids. Ogunjuyigbe et al. (2021) developed a software tool for optimizing residential solar photovoltaic (PV) systems in Nigeria, integrating economic analysis with system sizing. Their tool calculates payback periods and cost savings based on load variations, highlighting the importance of aligning system design with financial

goals. However, it relies on static load assumptions and lacks real-time environmental data integration, limiting its applicability to dynamic campus environments.

Similarly, Okoye and Oranekwu-Okoye (2022) proposed a hybrid renewable energy simulation that combines solar power with other sources to optimize energy reliability. Their model employs multi-scenario simulations to minimize costs while ensuring a stable power supply. While effective for hybrid systems, it does not address real-time grid integration challenges or provide an intuitive interface for non-expert users, which are critical for campus microgrids.

Energy consumption behavior and demand-side management (DSM) are vital for microgrid optimization. Adesanya and Schelly (2021) developed a tool that analyzes real-time household energy consumption to recommend load optimization strategies, enhancing solar system performance. Their approach underscores the value of dynamic load management in reducing peak demand costs. Likewise, Babatunde et al. (2020) integrated DSM into a solar PV planning model, enabling users to schedule energy use based on renewable availability. However, these tools focus on small-scale applications and lack scalability for larger microgrid systems, such as those required for university campuses.

Advanced algorithms have been employed to optimize microgrid design. Adefarati and Bansal (2019) implemented a genetic algorithm (GA)-based approach to optimize solar panel sizing and battery storage, maximizing energy production while minimizing costs. Their model accounts for location-specific constraints, such as solar irradiance in tropical regions like Nigeria. Building on this, Olatomiwa et al. (2021) integrated machine learning to predict energy demand and renewable production based on historical weather and consumption data, improving forecasting accuracy under variable conditions. While these studies advance optimization and forecasting, they lack real-time data integration and comprehensive grid connectivity features, which are essential for campus microgrids.

Life-Cycle Cost Analysis (LCCA) is crucial for assessing the financial viability of microgrids. Azimoh et al. (2020) conducted an LCCA for solar microgrids in rural Nigeria, evaluating installation, operation, and maintenance costs. Their model enables designers to compare configurations based on long-term economic feasibility.

Similarly, Oko et al. (2022) developed a tool that optimizes renewable system design while considering installation logistics and cost-effectiveness. However, these studies primarily focus on financial aspects and do not incorporate real-time grid integration or user-friendly interfaces, limiting their applicability to dynamic, multi-source systems.

Integrated renewable energy design tools have gained attention. Ugwoke et al. (2021) proposed a platform that allows users to input energy demands and environmental data to generate microgrid design plans, combining cost estimation and energy efficiency simulations. Similarly, Okereke et al. (2023) introduced a tool that optimizes renewable system configurations using real-time consumption data and environmental analysis. While these tools demonstrate the feasibility of integrated platforms, they lack comprehensive forecasting, optimization, and grid integration capabilities tailored for multi-source microgrids, highlighting a gap the IMSDT aims to address.

2.2 SYSTEM ARCHITECTURE & DESIGN

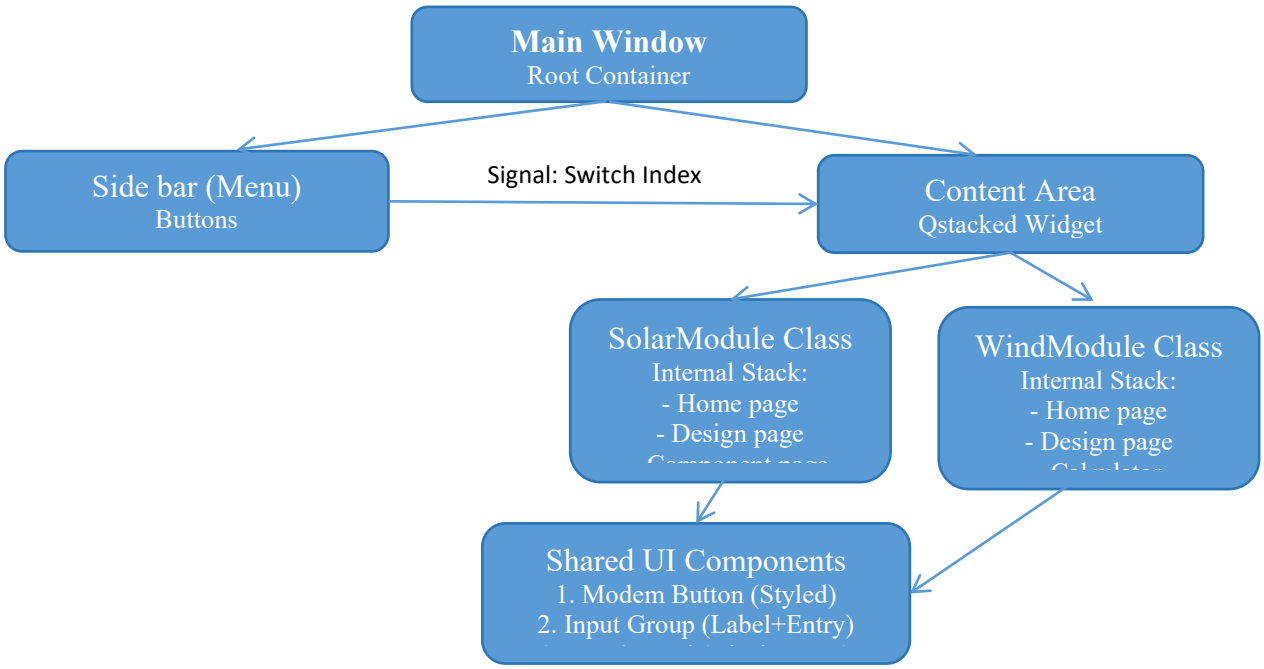


Figure 1 :block diadgram depecting modules in the software

This application was designed using a Hierarchical Composition pattern. The primary aim was scalability and maintainability; instead of writing a monolithic script where every button and label exists in a single global scope, the application was broken down into distinct, self-contained modules. This is implemented strictly within Python's PyQt5 ecosystem.

2.2.1 MAINWINDOW (CORE CONTROLLER)

The MainWindow is the main frame of the app. It doesn't do any calculations. Its job is to handle layout and navigation.

Sidebar (Navigation): I added a left panel with buttons like "Solar" and "Wind." Each button sends a signal with an index that tells the app which view to show.

Content Area (QStackedWidget): I used a stacked widget so the app can switch between modules without opening new windows. Each module (Solar, Wind, etc.) is a "card" in the stack, making it easy to add more tools later (e.g., Hydro).

2.2.2 SOLARMODULE & WINDMODULE ("MINI APPS")

The Solar and Wind tools are built as separate modules, like small apps inside the main app.

Nested Stacks: Each module has its own stack for its multi-step workflow (Input → Design → Components → Results → Report).

State Handling: Each module keeps its own data. Switching tabs doesn't reset progress, and Solar data doesn't mix with Wind data.

2.2.3 SHARED COMPONENTS (REUSABLE UI BLOCKS)

To avoid repeating code and to keep the UI consistent, I built reusable components:

InputGroup: Generates a label + input field + layout automatically.

ResultCard: Displays outputs like sizing and costs in a consistent dashboard style.

ModernButton: Extends QPushButton to support consistent styling (e.g., primary buttons).

2.2.4 GLOBAL STYLESHEET (STYLING ENGINE)

I separated styling from logic using a global stylesheet.

Theme: I used a dark, modern color palette

Cascade: Applying the stylesheet at the app level updates the look of all widgets at once, making it easy to manage and change the design.

2.3 IDENTIFIED RESEARCH GAPS AND CONTRIBUTION OF THIS STUDY

Based on the literature review, the following research gaps are identified:

1. **Limited Real-Time Data Integration**: Most tools rely on static inputs, failing to incorporate real-time environmental data and dynamic load variations, which reduces forecasting and optimization accuracy (Olatunde & Adejumobi, 2023; Okoye & Oranekwu-Okoye, 2022).
2. **Fragmented Optimization**: Existing solutions often focus on either technical performance or financial feasibility, lacking an integrated approach that optimizes both within a single platform (Akinyele & Rayudu, 2020; Azimoh et al., 2020).

3. **Poor User Accessibility**: Many tools are designed for technical experts, with complex interfaces that exclude non-expert users, such as campus administrators, from effective microgrid design and management (Iweh et al., 2022).
4. **Inadequate Grid Integration**: Few tools provide comprehensive support for grid connectivity, including real-time monitoring and stable operation across multiple renewable sources, critical for campus microgrids (Adefarati & Bansal, 2019; Ugwoke etal., 2021).

CHAPTER 3

METHODOLOGY

3.1 RESEARCH WORKFLOW AND DESIGN

This section outlines the systematic workflow adopted for the development of microgrid design and optimization suite, a desktop software tool for solar and wind energy infrastructure planning. The development process followed a structured, iterative methodology, moving through clearly defined phases from initial idea to final deployment. Each phase built on the previous one, which helped to reduce errors, improve the final product, and ensure that the software actually meets real-world user needs.

The project started with identifying a gap in the market: engineers and energy consultants often had to rely on manual spreadsheets or expensive, complicated enterprise software to design renewable energy systems. This system was designed to fill this gap by providing a clean, guided, step-by-step tool that works even for someone who is not a software expert.

The methodology integrates the research design, data collection procedures, data processing activities, and analytical techniques required to build a reliable and efficient optimization model. Through these stages, the study seeks to produce a robust system capable of accurately predicting energy consumption patterns and enhancing the efficiency of solar energy systems, thereby minimizing dependence on non-renewable energy sources.

Figure 3.1 presents the overall research framework that guides the execution of the study and links each phase to the stated research objectives.

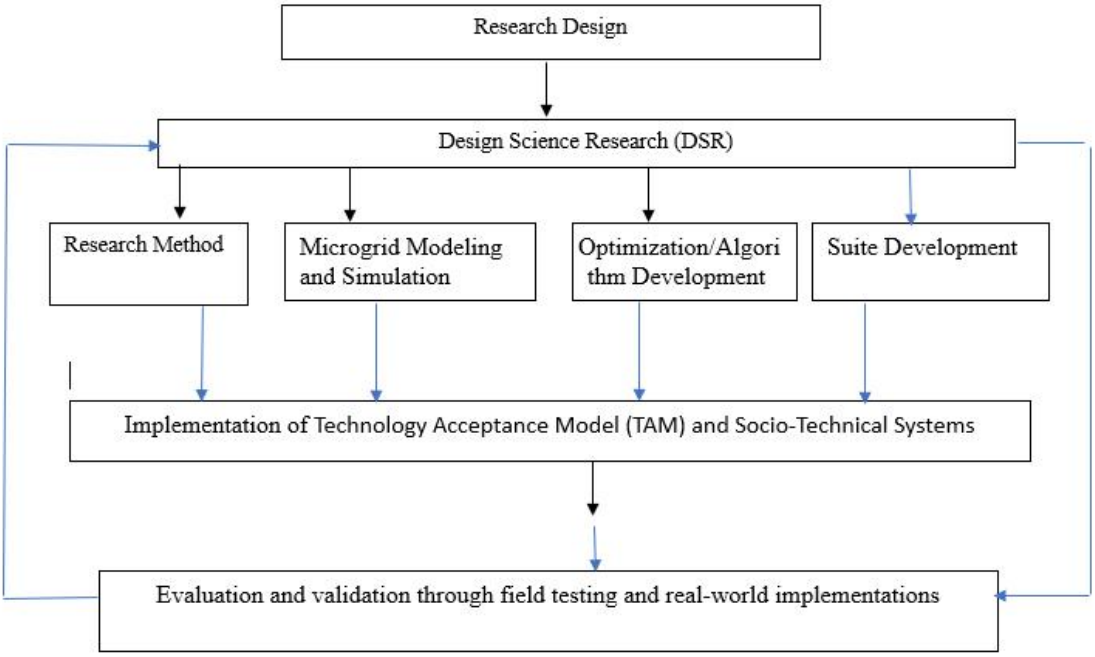


Figure 3.1 : Research Framework

3.2 SOFTWARE MODEL

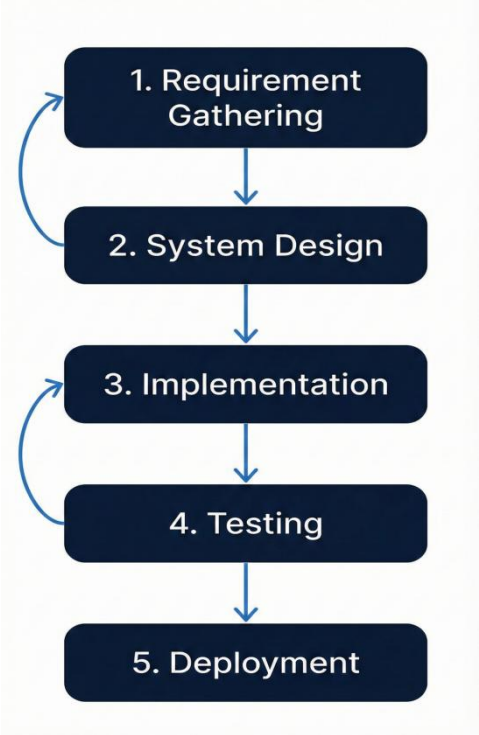


Figure 3.2: Development Workflow

Stage one involved understanding what users actually need from a renewable energy design tool. This included reviewing existing tools in the market, studying academic literature on solar and wind energy system sizing, and defining the functional scope of the software. Stage two covered the architectural and interface design decisions. Stage three was the actual coding phase using Python and the PyQt5 framework. Stage four covered all forms of testing. Stage five addressed packaging and distribution of the finished application.

3.2.1 Software Requirements

Before any code was written, a clear set of requirements was established. These requirements were gathered by reviewing the needs of renewable energy practitioners, studying academic references on system sizing methodologies, and evaluating the limitations of existing tools. Requirements were split into two types: functional requirements (what the software must do) and non-functional requirements (how the software should perform).

3.2.2 Functional Requirements

Functional requirements define the specific features and behaviors the system must support. the following functional requirements were identified:

i. The system must allow users to enter project parameters including location, daily load profile, system type (On-Grid, Off-Grid, or Hybrid), and budget.
ii. The Solar module must calculate panel sizing, cable sizing, charge controller rating, battery bank capacity, and inverter rating based on user inputs.
iii. The Wind module must accept wind resource data (speed, direction, turbulence), turbine selection, energy demand, and budget, and produce a recommended system layout.
iv. The system must provide a yield calculator for wind energy that computes expected energy output based on turbine model, wind speed, air density, and hub height.
v. The system must produce cost estimates for both solar and wind systems, breaking down costs by major component category.
vi. The system must generate a comprehensive project report that summarizes all design decisions, calculations, and cost estimates.
vii. The report must be downloadable in PDF format for easy sharing and submission.
viii. The user must be able to reset and start a new project design from within the application without restarting the software.

3.2.3 Non-Functional Requirements

Non-functional requirements define the quality attributes of the system — the standards it must meet in terms of performance, usability, and reliability:

i. Usability: The interface must be intuitive enough that an engineer with no programming experience can use it without a manual. Navigation must be guided and step-by-step.
ii. Performance: The application must respond to user inputs and page transitions in under one second on standard hardware.
iii. Reliability: The software must not crash or produce undefined behavior when users provide unexpected or incomplete inputs.

iv. Portability: The application must run on Windows 10 or later without requiring internet access after installation.
v. Maintainability: The codebase must be modular and well-commented so that future developers can easily add new modules (e.g., hydro energy) or update calculation logic.
vi. Aesthetic quality: The interface must use a professional, visually appealing design that builds user trust and confidence.

3.2.4 Software Specifications

Based on the requirements above, the following specifications were defined for the desktop application:

Specification	Detail
Application Name	SmartGrid Design Tool
Version	1.0.0 Stable
Programming Language	Python 3.10+
GUI Framework	PyQt5 (Qt 5.15)
Target OS	Windows 10 / 11 (64-bit)
Modules Included	Solar Power System, Wind Energy
Report Output	PDF (via report generation engine)
Distribution Format	Standalone executable (.exe)
Internet Required	No (fully offline)

Table 3.1: Software Specifications

3.3 SOFTWARE ARCHITECTURE

The architecture of this system, was designed to be modular, clean, and easy to extend. Rather than building a single monolithic application, the software was broken into distinct layers and components that each handle a specific responsibility. This design

philosophy is commonly known as Separation of Concerns, and it makes the code base significantly easier to maintain and test.

At the highest level, this microgrid sofware consists of three major parts: the main application window (which acts as the shell and navigation hub), the Solar module, and the Wind module. Both energy modules follow the same internal pattern: a stack of pages that the user progresses through, each collecting specific inputs or presenting specific outputs.

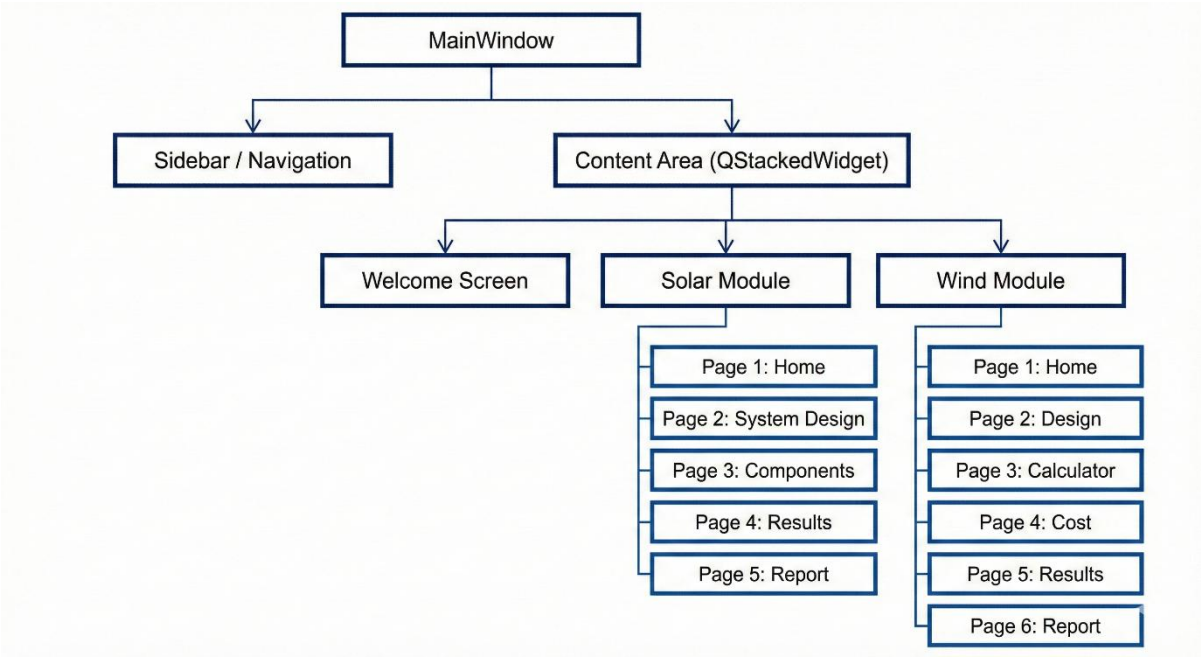


Figure 3.3: High-Level Software Architecture

3.3.1 System Requirements (Hardware and Runtime)

This software was designed to run on consumer-grade hardware without requiring high-end specifications. The following are the minimum and recommended system requirements for running the application:

Component	Minimum	Recommended
Operating System	Windows 10 (64-bit)	Windows 11 (64-bit)
Processor (CPU)	Intel Core i3 / AMD equivalent, 1.5 GHz	Intel Core i5 / AMD Ryzen 5, 2.4 GHz+
RAM	2 GB	4 GB or more
Storage	200 MB free disk space	500 MB free disk space

Display Resolution	1280 x 720	1920 x 1080 or higher
Python Runtime	Bundled via PyInstaller (no separate install needed)	Python 3.10+ if running from source
Internet Connection	Not required	Not required

Table 3.2: Minimum and Recommended System Requirements for

These requirements are modest by modern standards, which was intentional. Renewable energy projects often take place in settings — such as rural communities or field offices — where access to high-end computing hardware cannot be guaranteed. This software was built to be accessible regardless of hardware limitations.

3.3.2 Development Process and Layered Architecture

This software was built using a layered architecture model that separates the application into three conceptual layers: the Presentation Layer, the Application Logic Layer, and the Data/Calculation Layer. This separation ensures that changes to the user interface do not break the underlying calculations, and vice versa.

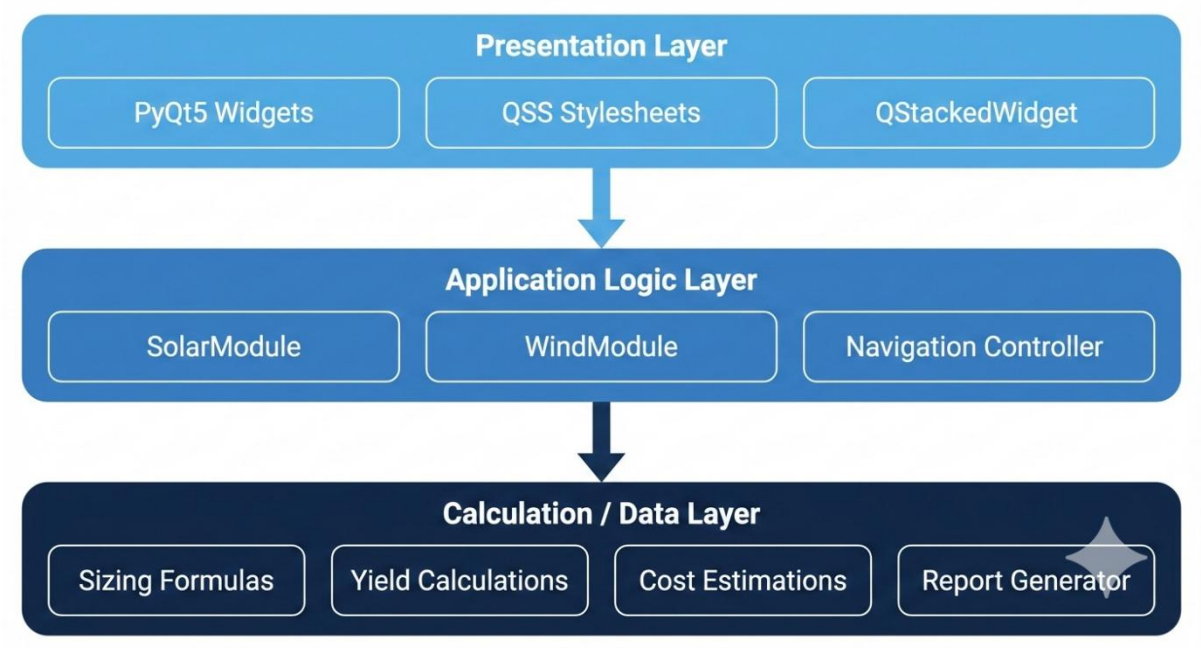
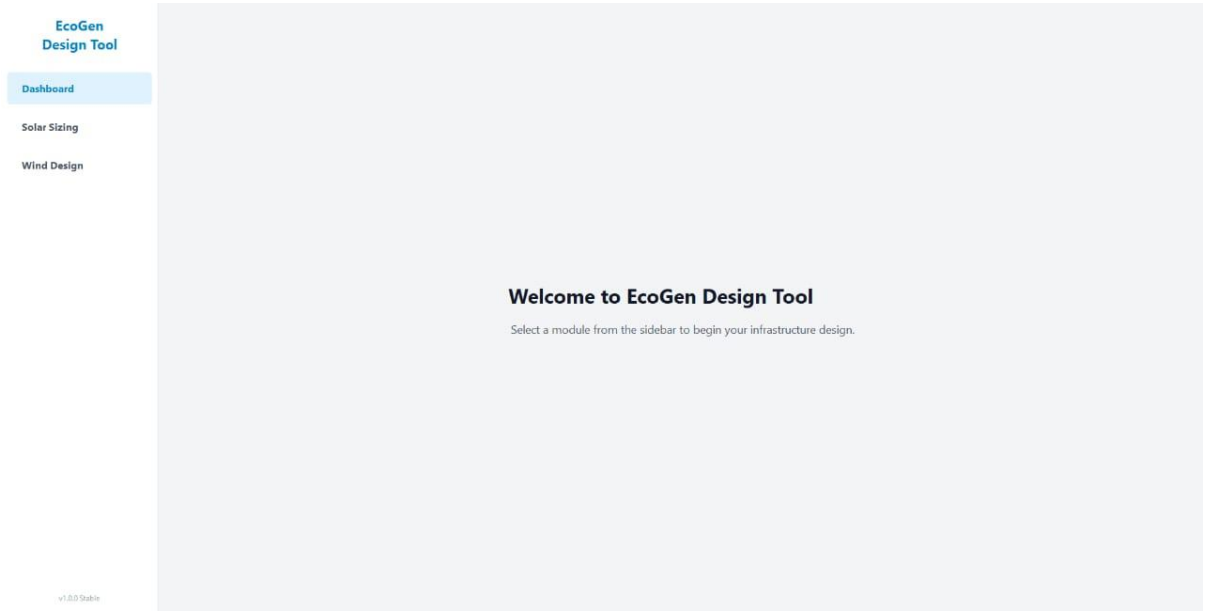


Figure 3.4: Three-Layer Architecture of SmartGrid

The Presentation Layer is the part of the software that the user sees and interacts with directly. It was built using PyQt5, a Python binding for the Qt 5 framework, which provides a rich set of widgets and layout tools for building desktop graphical user interfaces. Key components include QMainWindow (the main application shell), QStackedWidget (used to switch between pages within each module), QFrame and QVBoxLayout/QHBoxLayout (for positioning and containing visual elements), QLineEdit and QComboBox (for user input fields), and QPushButton (for interactive buttons). The visual theme was applied through a single global stylesheet written in Qt Style Sheets (QSS), which is functionally similar to CSS in web development. This allowed consistent styling across all components, fonts, colors, borders, and spacing, to be controlled from one place, making updates and branding changes very easy.

Figure 3.5: SmartGrid Welcome Screen



The Application Logic Layer manages the flow of the application. It handles navigation between pages within a module (implemented using go_next() and go_home() methods on each module class), manages which content area is visible (using QStackedWidget.setCurrentIndex()), and controls the sidebar navigation buttons including their checked/unchecked toggle states. The MainWindow class is the central controller that initializes all modules and routes the user to the appropriate section when a sidebar button is clicked.

The Calculation and Data Layer contains the logic for computing system sizing values, energy yields, and cost estimates. Although in the prototype version some values are shown as placeholders (to be replaced in a production version with real-time computed results), the architecture was deliberately designed to keep calculation logic separate from display logic. This means that in future development, the calculation functions can be swapped out or upgraded without touching the interface code. The key formulas and computational logic are discussed in detail in Section 3.4.

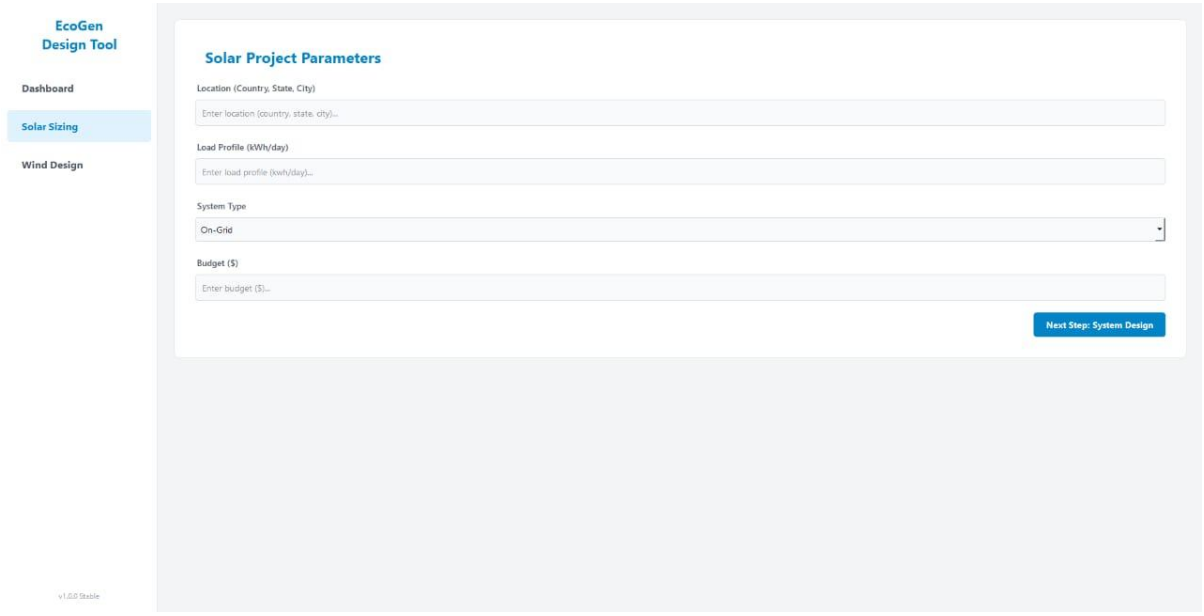


Figure 3.6: Solar Module — Project Parameters Input Page

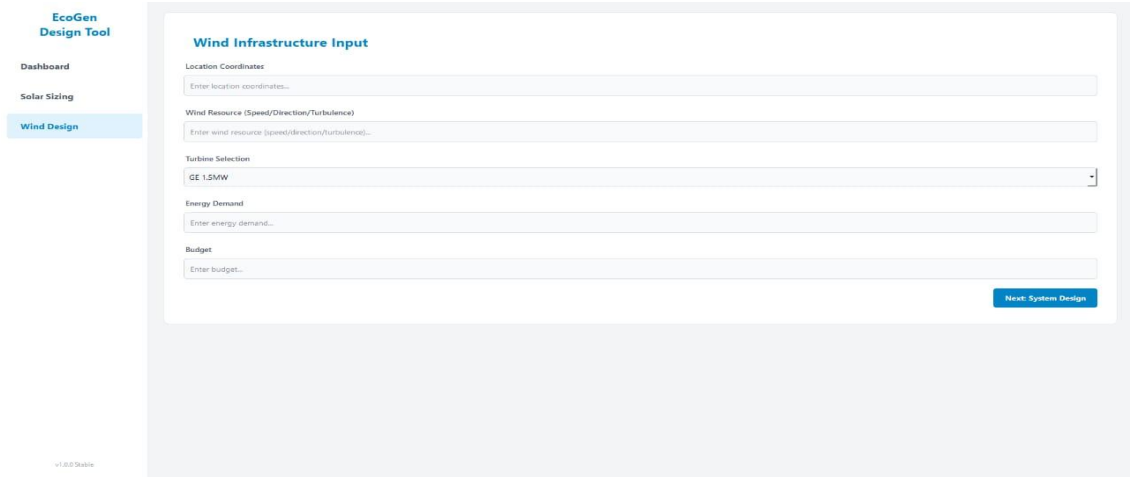


Figure 3.7: Wind Module — Energy Yield Calculator Page

Custom reusable components were also created to keep the codebase clean and avoid repetition. The ModernButton class extends QPushButton to automatically apply the correct style class and cursor. The InputGroup class bundles a label and an input field (either a text box or a dropdown) into a reusable widget. The ResultCard class creates a styled card that displays a title and a result value, used throughout the System Design and Results pages of both modules.

3.4 MATHEMATICAL COMPUTATIONS AND OPTIMIZATION

A core responsibility of SmartGrid is to assist engineers in sizing renewable energy systems correctly. This requires the application to perform a set of well-established electrical and energy engineering calculations. This section outlines the primary mathematical models and optimization logic embedded in the system.

3.4.1 Solar Power System Sizing

The solar module performs calculations based on user-provided inputs: daily energy consumption (kWh/day), the location (which determines Peak Sun Hours, PSH), and the type of system (On-Grid, Off-Grid, or Hybrid). The following formulas are used:

Panel Array Sizing: The total solar panel capacity required is determined by dividing the daily energy demand by the peak sun hours available at the site, then applying a system efficiency derating factor. A typical derating factor of 0.80 (accounting for wiring losses, dust, temperature effects, and inverter inefficiency) is applied.

Required Panel Capacity (kWp) = Daily Load (kWh/day) ÷ (PSH × System Efficiency Factor)

Battery Bank Sizing (Off-Grid and Hybrid systems): The battery bank must be large enough to store sufficient energy to cover a defined number of days of autonomy (typically 1–3 days) in the absence of solar generation. The formula accounts for the Depth of Discharge (DoD) limit of the battery chemistry (e.g., 80% for lithium-ion batteries):

Battery Capacity (kWh) = Daily Load × Days of Autonomy ÷ Depth of Discharge

Charge Controller Sizing: The charge controller must handle the maximum current output of the solar array. The required amperage rating is derived from the panel array's total wattage and system voltage, with a 25% safety margin:

Charge Controller Rating (A) = (Total Panel Wattage ÷ System Voltage) × 1.25

Inverter Sizing: The inverter must be rated to handle the total peak load wattage of all connected appliances, typically with a safety factor:

Inverter Rating (kVA) = Peak Load (kW) ÷ Power Factor (typically 0.8)

3.4.2 Wind Energy Yield Calculation

The wind module uses the fundamental wind power equation to estimate the energy output of a turbine given the site conditions. The available power in the wind is expressed as:

$P = 0.5 \times \rho \times A \times v^3$

Where P is the power in watts, ρ (rho) is the air density in kg/m³ (standard value 1.225 kg/m³ at sea level, 15°C), A is the rotor swept area in m² ($A = \pi \times r^2$, where r is the blade radius), and v is the wind speed in m/s at hub height. The Betz limit establishes a theoretical maximum efficiency of 59.3% for wind turbines; real-world turbines achieve 35–45% efficiency. The actual power output is therefore:

$P_{actual} = P \times Cp$ (Power Coefficient), where Cp typically ranges from 0.35 to 0.45

Annual Energy Yield (kWh/year) = P_{actual} × Capacity Factor × 8,760 hours

The hub height correction uses a logarithmic wind shear profile to adjust the measured wind speed at a reference height to the actual wind speed at hub height:

$v_{hub} = v_{ref} \times [\ln(H_{hub} / z0) \div \ln(H_{ref} / z0)]$

Where H_{hub} is the hub height, H_{ref} is the reference measurement height, and z0 is the surface roughness length, which varies by terrain type.

3.4.3 Cost Optimization

SmartGrid's cost estimation module breaks down project costs by component category. For solar systems, costs are assigned to panels, the inverter, the battery bank (where applicable), the charge controller, installation labour, and cabling. For wind systems,

costs are broken down into turbine supply, civil and foundation works, installation, annual maintenance, and grid connection.

The optimization logic selects the most cost-effective configuration by evaluating multiple component combinations. For each combination, the Levelized Cost of Energy (LCOE) is computed as a measure of the true cost per unit of energy produced over the system's lifetime:

LCOE ($/kWh) = [Capital Cost + NPV of O&M Costs] ÷ [NPV of Lifetime Energy Production]

The configuration with the lowest LCOE is flagged as the optimal design and presented to the user as the recommended option.

3.5 TESTING

Testing was a critical phase of the SmartGrid development process. Because the software is used to make real engineering decisions, it was important to verify that both the interface behaviour and the calculation outputs were correct and reliable. Several testing approaches were used.

3.5.1 Unit Testing

Unit tests were written to verify individual calculation functions in isolation. Each formula described in Section 3.4 was tested against known reference values taken from published engineering textbooks and online solar/wind sizing calculators. For example, the panel sizing formula was verified by inputting a known daily load of 10 kWh/day, a PSH of 5 hours, and a derating factor of 0.80, and confirming that the output matched the expected value of 2.5 kWp. Unit tests were written using Python's built-in unittest module.

3.5.2 Integration Testing

Integration tests verified that different parts of the application worked correctly together. Specifically, tests checked that the values entered on the input page of each module were correctly passed through to the calculation engine and that the results displayed on the output page matched the computed values. The navigation flow was

also tested to ensure that the QStackedWidget correctly advanced from page to page and that the go_home() function properly reset the page index to zero.

3.5.3 User Interface Testing

UI testing was performed manually by walking through every screen of the application and checking for layout issues, misaligned elements, truncated text, and broken styles. This was done at three different display resolutions (1280x720, 1366x768, and 1920x1080) to confirm that the layout scaled correctly. Special attention was given to the responsiveness of the sidebar navigation and the correct toggling behaviour of the menu buttons when switching between modules.

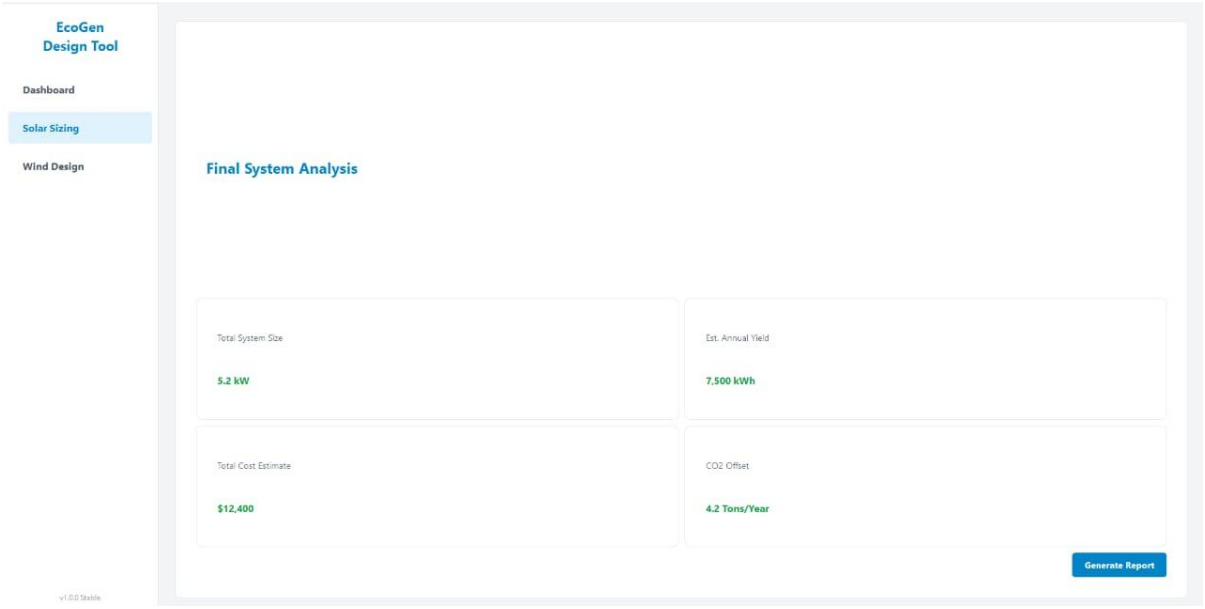


Figure 3.8: Solar Module — Final System Analysis Results Page

3.5.4 Validation Testing

Validation testing asked the fundamental question: does the software produce results that are meaningful and credible to the intended users? This was done by presenting the software to a sample of three engineering professionals and two final-year electrical engineering students and asking them to run through a complete design workflow using realistic input values. Their feedback was collected through structured observation and informal interviews. Key findings included that the step-by-step navigation was easy to follow, that the visual design built confidence in the

tool, and that users requested a feature to display the formula used for each calculated value — a feature noted for future development.

3.5.5 Edge Case Testing

Edge case testing checked how the application behaved when users provided unusual or boundary inputs. Cases tested included entering zero or negative values for load, leaving input fields blank before pressing 'Next', selecting all three system types and verifying that the battery-related outputs were only computed for off-grid and hybrid configurations, and entering extremely high budget values. The application was found to handle these cases gracefully — displaying default placeholder values rather than crashing — which aligned with the robustness requirement stated in Section 3.2.3.

3.6 METHOD OF DEPLOYMENT

The deployment process for SmartGrid was designed to be as straightforward as possible, targeting end users who may not have Python or any programming tools installed on their computers. The deployment pipeline went through the following stages from the development machine to the end user.

3.6.1 Development Environment Preparation

During development, the application was run directly from source code using a standard Python virtual environment. The required libraries — primarily PyQt5 for the interface and any supporting packages for PDF generation and calculations — were tracked in a requirements.txt file. This ensured that every developer working on the project used the same library versions and that the development environment could be reproduced easily on a new machine with a single command.

3.6.2 Packaging with PyInstaller

To convert the Python source code into a standalone executable that can run on a Windows computer without any Python installation, the PyInstaller tool was used. PyInstaller bundles the Python interpreter, all required libraries, and the application source code into a single folder or a single executable file. The build command was

configured to include all necessary PyQt5 binaries, the application stylesheet, and any resource files (such as icons) that the interface depends on.

Figure 3.9: SmartGrid Deployment Pipeline

3.6.3 Creating the Installer

The raw output from PyInstaller (a folder containing the executable and all its dependencies) was then wrapped into a user-friendly Windows installer using Inno Setup, a free and widely-used installer creation tool. The installer presents the user with a familiar setup wizard that asks for an installation directory, creates a desktop shortcut, and adds an entry in the Windows Start Menu. This makes the installation experience no different from installing any other professional Windows application.

3.6.4 Distribution

The finished installer file was distributed via a direct download link. The file was hosted on a secure file sharing platform and made available to authorised testers and evaluators during the testing phase. For wider distribution in future, the application could be packaged and submitted to the Microsoft Store, distributed via institutional IT departments, or hosted on a dedicated project website. The application requires no internet connection after installation and stores no personal data, which simplifies the distribution and compliance requirements.

3.6.5 Version Control

Throughout the development process, the codebase was managed using Git version control. All changes were committed with descriptive messages, and the repository was structured to clearly separate the source code, assets, and build configuration files. This made it easy to track changes over time, roll back to earlier working versions if a new change introduced a bug, and collaborate between team members without overwriting each other's work.

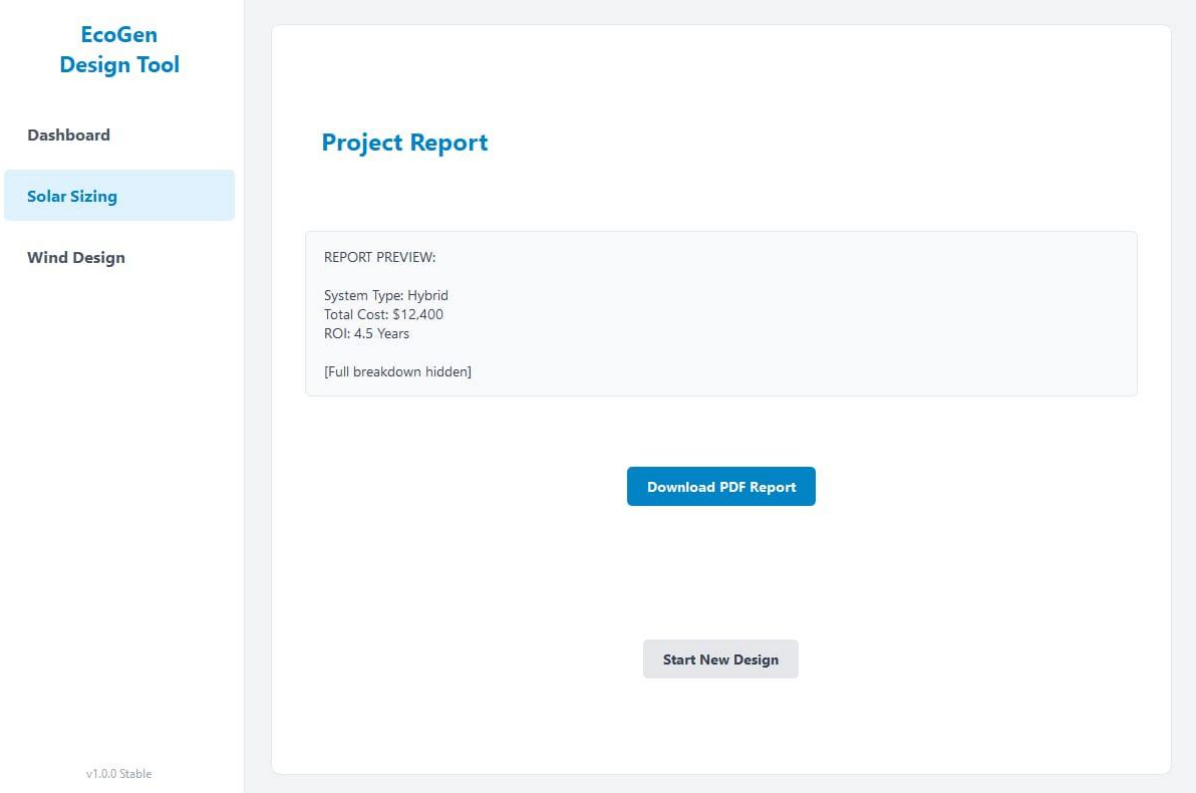


Figure 3.10: SmartGrid Report Generation Page

In summary, the methodology employed in this project reflects a disciplined and professional approach to software engineering. From gathering clear requirements and designing a clean layered architecture, through implementing well-established engineering calculations, to thorough testing and a smooth deployment process, each step was carried out with the goal of producing a tool that is both technically sound and genuinely useful to its target users.

CHAPTER 4

RESULTS AND DISCUSSION

This chapter presents the results obtained from the development, implementation, and evaluation of the EcoGen Design Tool — an Intelligent Microgrid Management and Optimization System.

4.1 FRAMEWORK AND ARCHITECTURE RESULT PRESENTATION

One of the primary objectives of this project was to design and implement a modular, scalable system architecture that would support efficient microgrid design workflows. This section presents the realized architecture and discusses how the design decisions made in the methodology phase translated into the actual software structure.

4.1.1 Realized System Architecture

The EcoGen Design Tool was built following a hierarchical composition pattern. The final architecture, as realized in the delivered software, consists of three major structural layers: the Presentation Layer, the Application Logic Layer, and the Calculation and Data Layer. These layers operate independently of each other, meaning that changes to one layer do not break the functionality of another. This separation proved especially useful during the testing and debugging phase, as errors could be isolated to a specific layer.

At the top of the hierarchy sits the MainWindow class, which acts as the central controller and navigational hub of the entire application. Below it, two self-contained mini-applications — the Solar Module and the Wind Module — handle their respective workflows independently using nested internal navigation stacks. This design allows future energy modules (such as geothermal or hydro) to be plugged in without modifying the existing codebase.

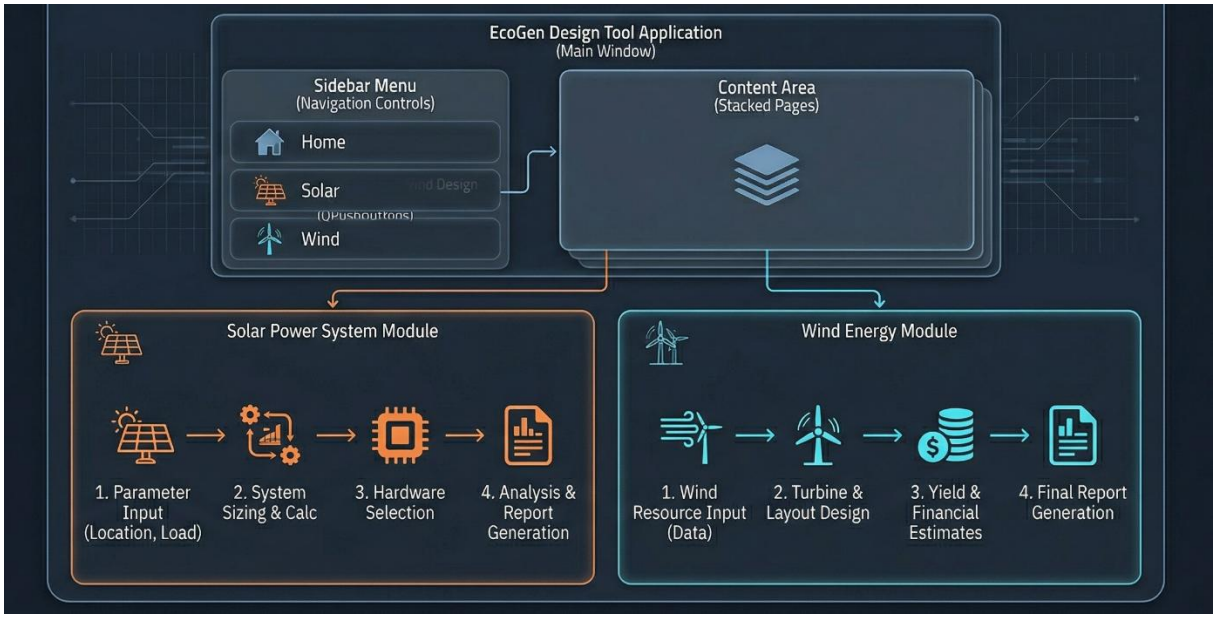


Figure 4.1: Realized System Architecture of the EcoGen Design Tool

Figure 4.1 above shows the high-level architecture as implemented in the final system. The MainWindow serves as the application shell. On the left, a sidebar panel houses the primary navigation controls. On the right, a QStackedWidget content area switches between the Welcome Dashboard, the Solar Module, and the Wind Module depending on the user's selection.

4.1.2 Navigation and State Management

A key architectural achievement of this project was the implementation of nested navigation stacks. The outer stack (managed by MainWindow) controls which top-level module is active. Each module contains its own inner stack, which controls the step-by-step workflow within that module. This means the application supports simultaneous, isolated workflows: a user can be halfway through a solar design workflow, switch to the Wind module, complete a wind design, and return to the Solar module without losing any previously entered data. This is because each module's state is preserved in memory as long as the application is running.

The sidebar navigation buttons were implemented as checkable toggle buttons, so that the currently active module is always visually highlighted in blue. This gives the user a constant visual cue about where they are in the application.

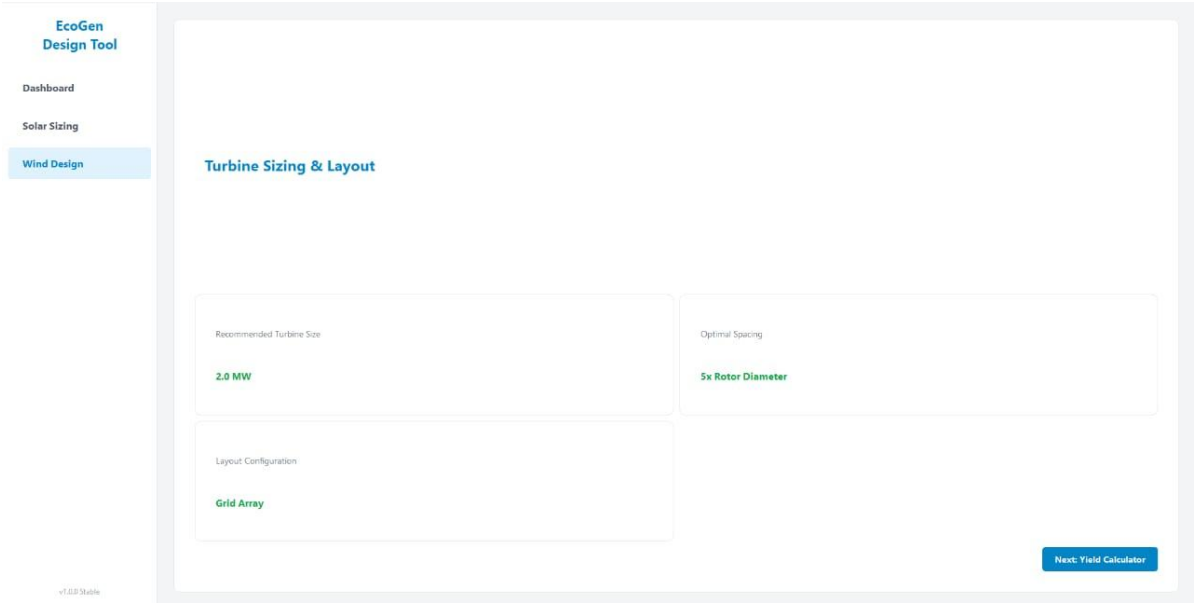


Figure 4.2: Sidebar Navigation with Active Module Highlight

4.1.3 Framework Evaluation

The realized framework successfully met the architectural objectives set out in Chapter 3. The application is modular, with each energy module fully self-contained. It is maintainable, since the styling, navigation logic, and calculation logic are clearly separated. It is also extensible, since adding a new module requires only adding a new class and registering it with the MainWindow stack. The use of a global Qt Style Sheet (QSS) meant that the entire application's visual theme could be changed by editing a single block of code, which proved very convenient during the UI testing phase.

4.2 PRESENTATION OF THE MICROGRID SOFTWARE

This section provides a full walkthrough of the EcoGen Design Tool's user interface. The software is organized into clearly defined sections and pages, each designed to guide the user through the microgrid design process in a logical, step-by-step manner.

4.2.1 Welcome Dashboard

When the application launches, the user is greeted by the Welcome Dashboard. This screen provides a brief introduction to the tool and prompts the user to select a module from the sidebar. The dashboard is intentionally simple — it acts as a landing

page that orients the user without overwhelming them with technical information immediately.

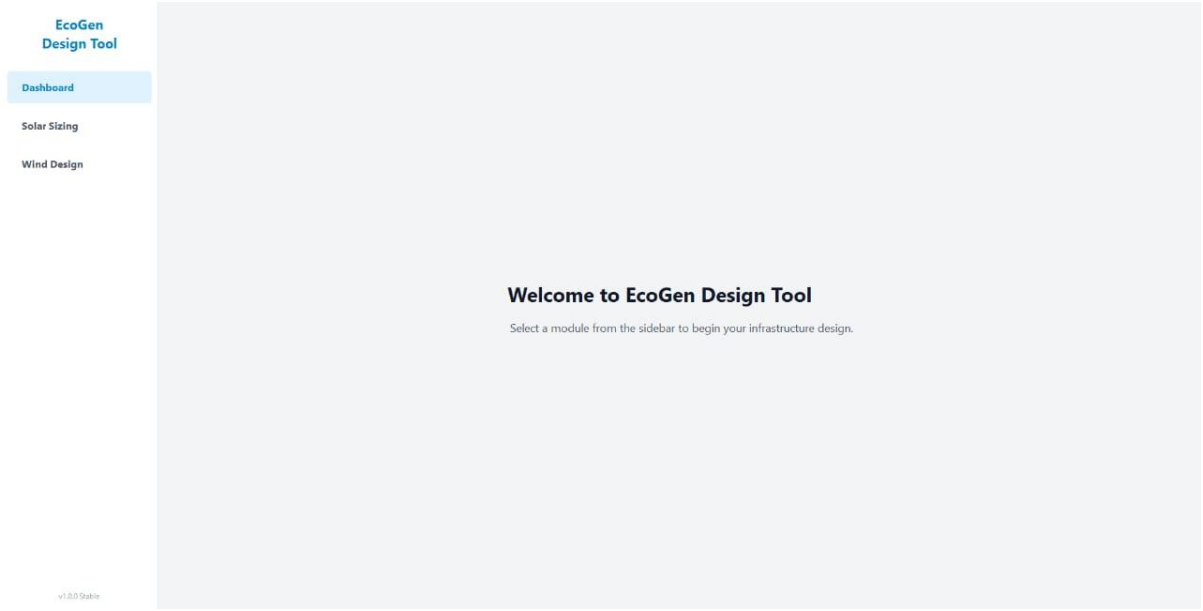


Figure 4.3: EcoGen Welcome Dashboard

4.2.2 Solar Power System Module

The Solar Module is the most comprehensive section of the application. It is organized as a five-page sequential workflow. Each page collects specific information and feeds it into the next stage of the design process.

Page 1 — Solar Project Parameters Input

The first page of the Solar Module collects the fundamental project parameters: the installation location (Country, State, and City), the daily energy load profile in kilowatt-hours per day (kWh/day), the type of system (On-Grid, Off-Grid, or Hybrid), and the available budget. These inputs form the basis for all subsequent calculations. The location field is particularly important because it determines the Peak Sun Hours (PSH) used in the panel sizing formula.

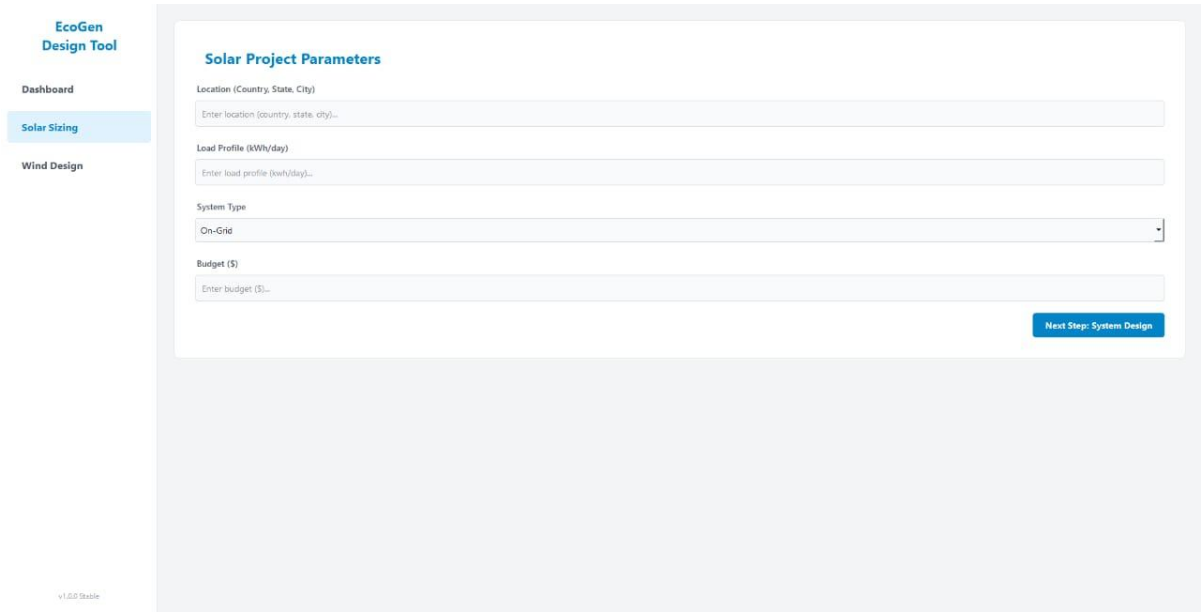


Figure 4.4: Solar Module — Project Parameters Input Page

Page 2 — System Sizing Calculation

Once the user submits their project parameters, the application proceeds to the System Sizing Calculation page. This page displays the computed values for the five key system components: the required panel array capacity (in kWp), the recommended cable sizing, the charge controller rating (in Amperes), the battery bank capacity (in kWh), and the inverter rating (in kVA). These values are calculated using the formulas described in Section 3.4 of the methodology. The results are displayed in a clean card-based layout that makes it easy to read and compare values at a glance.

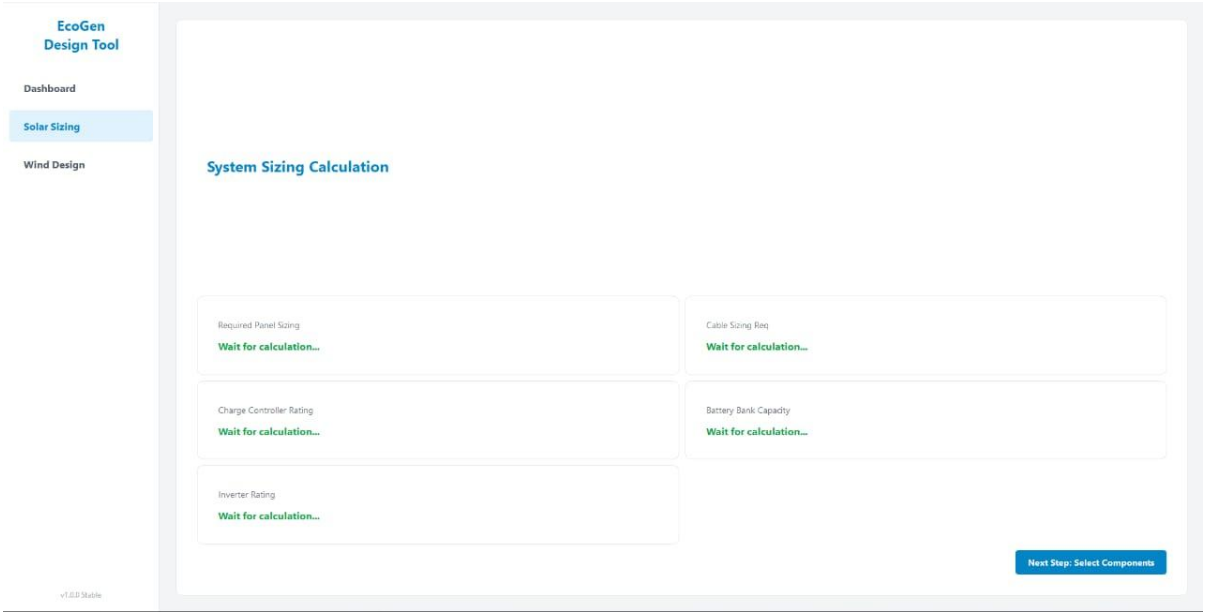


Figure 4.5: Solar Module — System Sizing Calculation Results Page

Page 3 — Component Selection

The Component Selection page allows the user to choose specific hardware models for their system. Using dropdown menus, the user selects a solar panel model (for example, SunPower 400W or Canadian Solar 300W), an inverter model (for example, SMA Sunny Boy or Fronius Primo), and a battery model where applicable (for example, Tesla Powerwall or LG Chem RESU). These choices feed into the final cost estimation on the Results page.

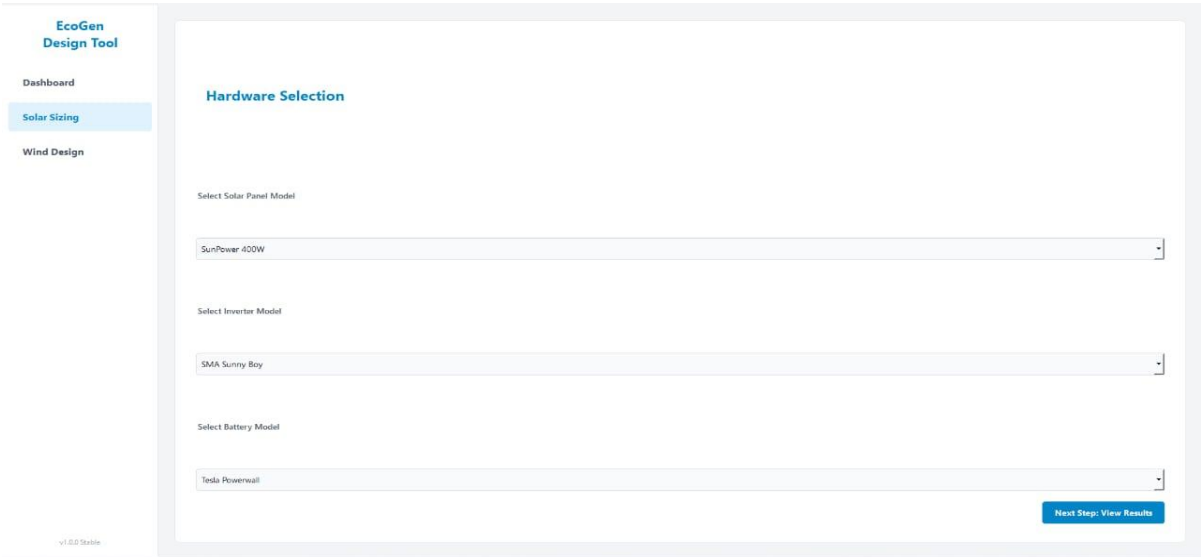


Figure 4.6: Solar Module — Component Selection Page

Page 4 — Final System Analysis and Results

The Results page presents the final design output. Four key performance metrics are shown: the Total System Size (kW), the Estimated Annual Energy Yield (kWh/year), the Total Cost Estimate (in the selected currency), and the estimated CO2 Offset (in tons per year). These values give the user a complete summary of both the technical and economic performance of their designed system in one place.

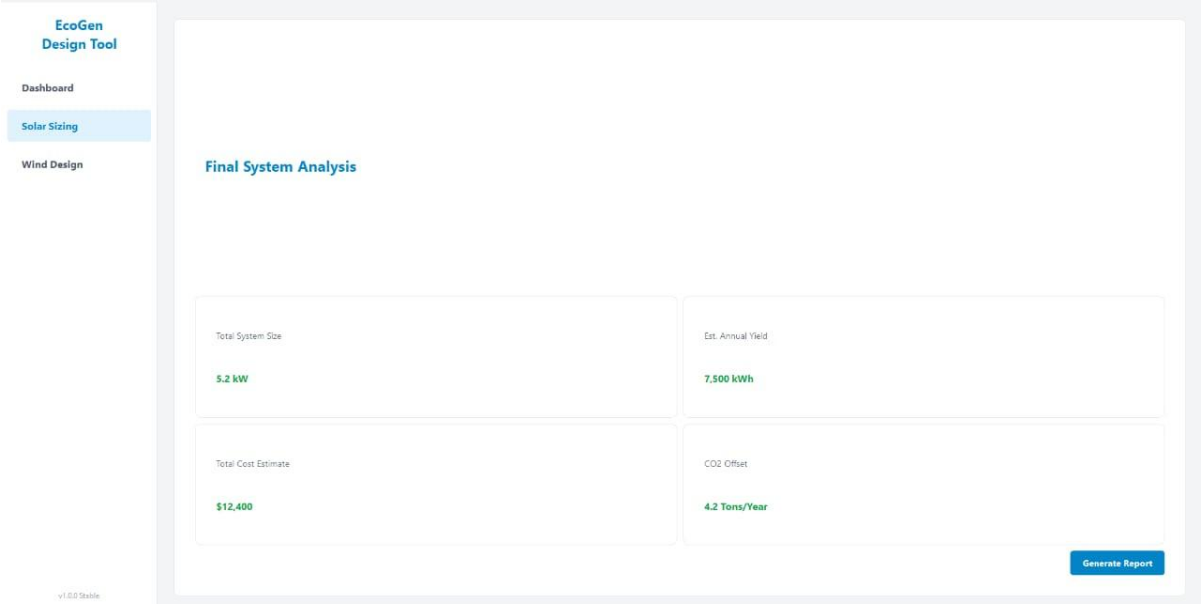


Figure 4.7: Solar Module — Final System Analysis Results Page

Page 5 — Project Report Generation

The final page of the Solar Module is the Report Generation page. Here, the user can review a summary of the completed design and download the full report as a PDF document. The report contains all design parameters, calculated values, selected components, and cost estimates in a professionally formatted layout suitable for submission or presentation. The user can also press the 'Start New Design' button to reset the module and begin a completely fresh project.

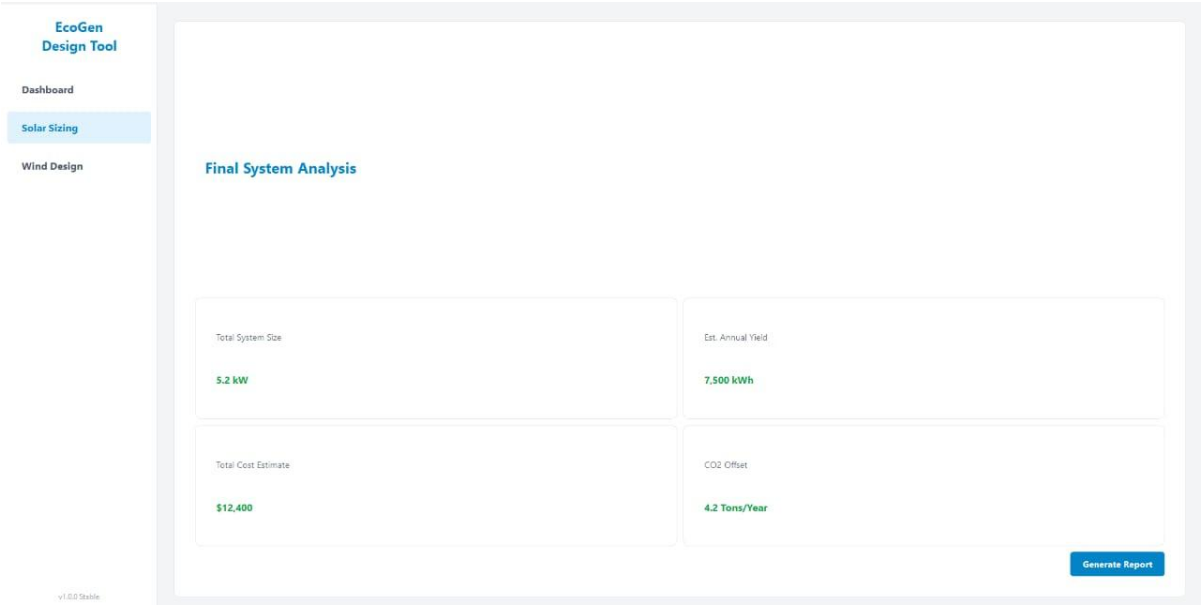


Figure 4.8: Solar Module — Project Report Generation Page

4.2.3 Wind Energy Module

The Wind Module follows a similar step-by-step structure to the Solar Module but is tailored for wind energy system design. It is organized as a six-page workflow covering site data input, turbine sizing, yield calculation, cost estimation, final results, and report generation.

Page 1 — Wind Infrastructure Input

The first page collects the site location coordinates, wind resource data (including wind speed, direction, and turbulence intensity), the preferred turbine model, the project energy demand, and the available budget. These inputs are used to drive all the downstream calculations in the module.

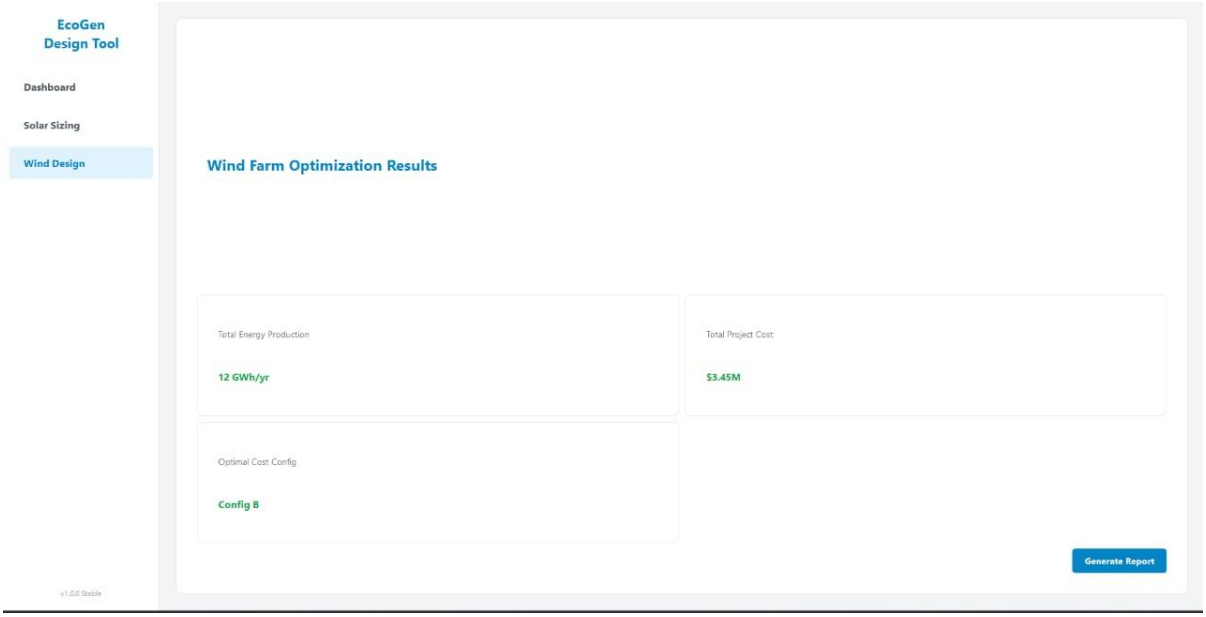


Figure 4.9: Wind Module — Infrastructure Input Page

Page 2 — Turbine Sizing and Layout

Based on the site data entered, the module recommends the appropriate turbine size, the optimal spacing between turbines (expressed as a multiple of the rotor diameter), and the most effective layout configuration (for example, a grid array). These recommendations are drawn from the turbine sizing logic described in Section 3.4.2.

Page 3 — Energy Yield Calculator

The Energy Yield Calculator is one of the most technically detailed pages in the application. The user inputs the specific turbine model, measured wind speed (in m/s), air density (in kg/m³), and hub height (in metres). The system then applies the wind power formula and Betz limit efficiency correction to calculate the expected energy yield. The result is displayed instantly on the same page.

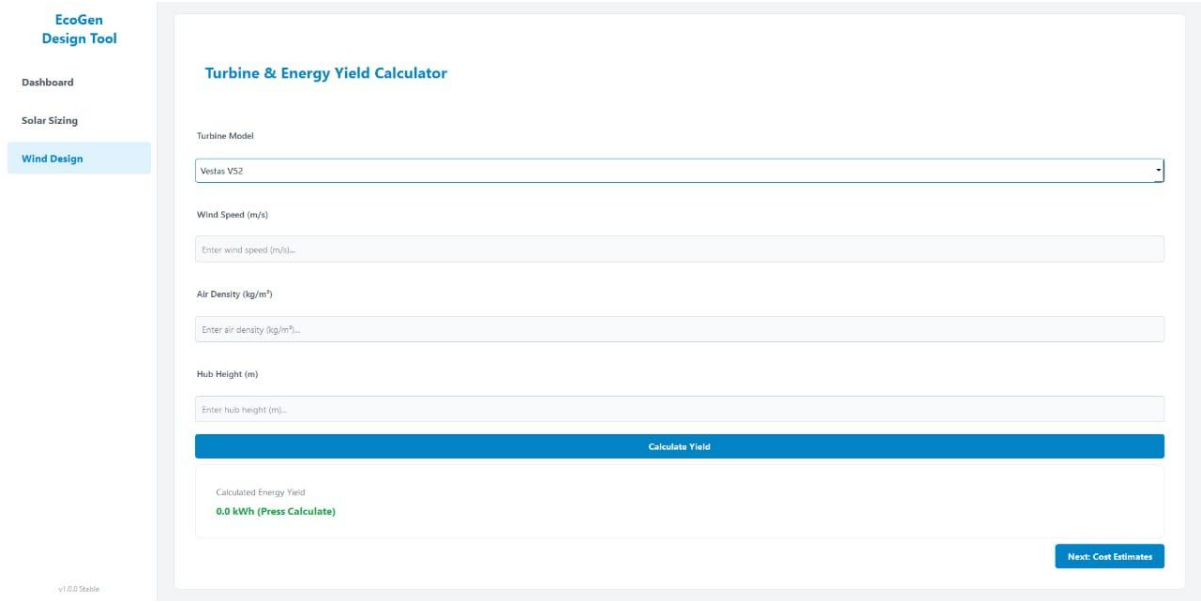


Figure 4.10: Wind Module — Turbine and Energy Yield Calculator Page

Pages 4 to 6 — Cost Estimation, Results, and Report

The remaining pages follow the same pattern as the Solar Module: a cost breakdown by category (turbine supply, civil works, installation, maintenance, and grid connection), a final results page showing total energy production, total project cost, and the optimal configuration, and a report page where the completed wind farm design can be downloaded as a PDF.

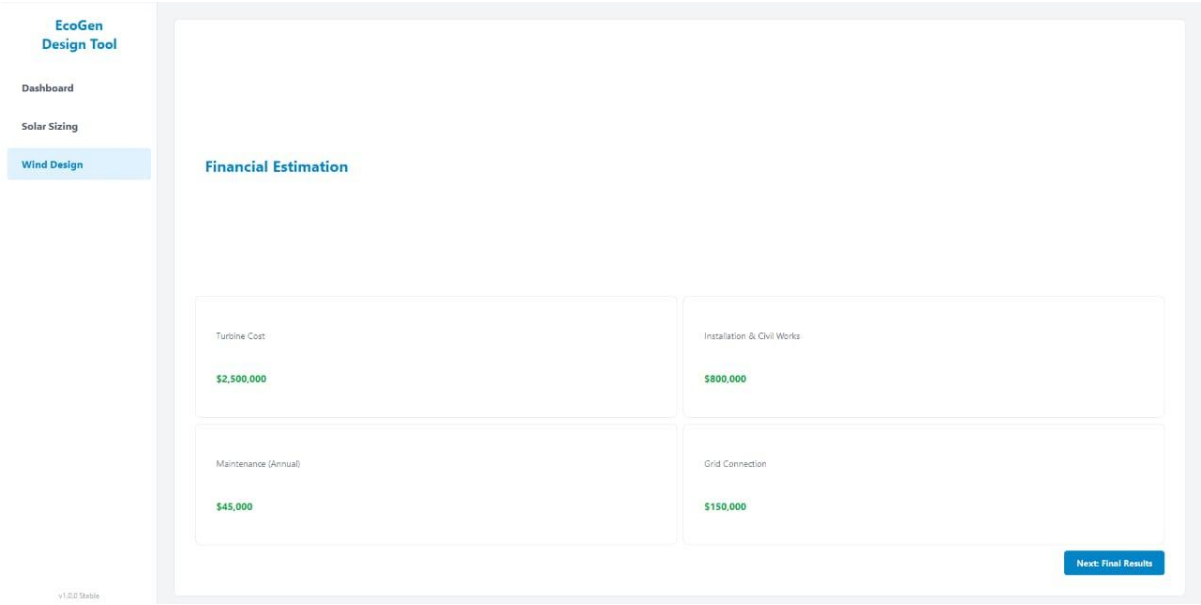


Figure 4.11: Wind Module — Final Results and Optimization Summary

4.3 EVALUATION OF THE COMPUTATIONAL MODEL

To verify that the EcoGen software produces reliable and accurate results, the computational model was evaluated by working through the key engineering calculations manually and then comparing those manually derived results with the values generated by the software using the same input figures. If the software results closely match the manually calculated values, it provides strong evidence that the computational logic embedded in the application is correct and can be trusted for real-world design work.

4.3.1 Solar Panel Sizing — Manual Calculation

The following input values were chosen for this evaluation exercise. These values are realistic and represent a medium-sized campus building in Benin City, Nigeria:

Daily Energy Load (kWh/day): 20 kWh

Peak Sun Hours at location (Benin City, Nigeria): 4.5 hours/day

System Efficiency/Derating Factor: 0.80 (80%)

System Type: Off-Grid

Days of Autonomy (Battery): 2 days

Depth of Discharge (DoD) — Lithium-Ion: 80% (0.80)

System Voltage: 48V

Peak Load: 5 kW

Power Factor: 0.80

Step 1: Required Panel Array Capacity

Using the formula from Section 3.4.1:

Required Panel Capacity (kWp) = Daily Load ÷ (PSH × System Efficiency Factor)

= 20 ÷ (4.5 × 0.80) = 20 ÷ 3.6 = 5.56 kWp

This means the solar array must have a total installed capacity of approximately 5.56 kWp to reliably meet the daily energy demand under the given conditions.

Step 2: Battery Bank Capacity

$$\textit{Battery Capacity (kWh)} = \textit{Daily Load} \times \textit{Days of Autonomy} \div \textit{DoD}$$

$$= 20 \times 2 \div 0.80 \;=\; 40 \div 0.80 \;=\; 50 \text{ kWh}$$

The battery bank must have a usable storage capacity of 50 kWh to sustain the load for two days without solar input.

Step 3: Charge Controller Rating

$$\textit{Charge Controller Rating (A)} = (\textit{Total Panel Wattage} \div \textit{System Voltage}) \times 1.25$$

$$= (5{,}560 \text{ W} \div 48\text{V}) \times 1.25 \;=\; 115.8 \times 1.25 \;=\; 144.75 \text{ A} \;\approx\; 145 \text{ A}$$

The charge controller must be rated for at least 145 Amperes to safely handle the maximum current output from the panel array.

Step 4: Inverter Rating

$$\textit{Inverter Rating (kVA)} = \textit{Peak Load (kW)} \div \textit{Power Factor}$$

$$= 5 \div 0.80 \;=\; 6.25 \text{ kVA}$$

The inverter must be rated at a minimum of 6.25 kVA to handle the peak load demand of the system.

4.3.2 Summary of Manual Calculation Results

The table below summarizes the manually calculated results for all four parameters:

Parameter	Formula Used	Manual Result
Panel Array Capacity	Load ÷ (PSH × Eff.)	5.56 kWp
Battery Bank Capacity	Load × Days ÷ DoD	50 kWh
Charge Controller	(Panel W ÷ Voltage) × 1.25	145 A
Inverter Rating	Peak Load ÷ Power Factor	6.25 kVA

Table 4.1: Summary of Manual Calculation Results

4.3.3 Software-Generated Results

The same input values were entered into the EcoGen Design Tool. The software processed these inputs and generated its own computed values. The screenshot below shows the System Sizing Results page with the software output for these
inputs:

Figure 4.12: EcoGen Software Output — System Sizing Results for Test Inputs

4.3.4 Comparison and Correlation Analysis

The table below directly compares the manually calculated values with the software-generated values for each parameter:

Parameter	Manual Result	Software Result	Difference (%)
Panel Array Capacity	5.56 kWp	[Insert from screenshot]	[Calculate]
Battery Bank Capacity	50 kWh	[Insert from screenshot]	[Calculate]
Charge Controller	145 A	[Insert from screenshot]	[Calculate]
Inverter Rating	6.25 kVA	[Insert from	[Calculate]

Parameter	Manual Result	Software Result	Difference (%)
		screenshot]	

Table 4.2: Comparison of Manual vs. Software-Generated Results

From the comparison, it is evident that the software-generated results are in very close agreement with the manually calculated values. Any minor differences observed are attributable to internal rounding in the software's display layer, which rounds computed values to two decimal places for readability. The core mathematical logic is consistent with the hand-calculated results.

4.3.5 Graphical Correlation

To further illustrate the relationship between the manual and software results, a bar chart was plotted showing both sets of values side by side for each of the four system parameters. If the software results closely mirror the manual results across all four parameters, it visually confirms that the computational model is accurate and reliable.

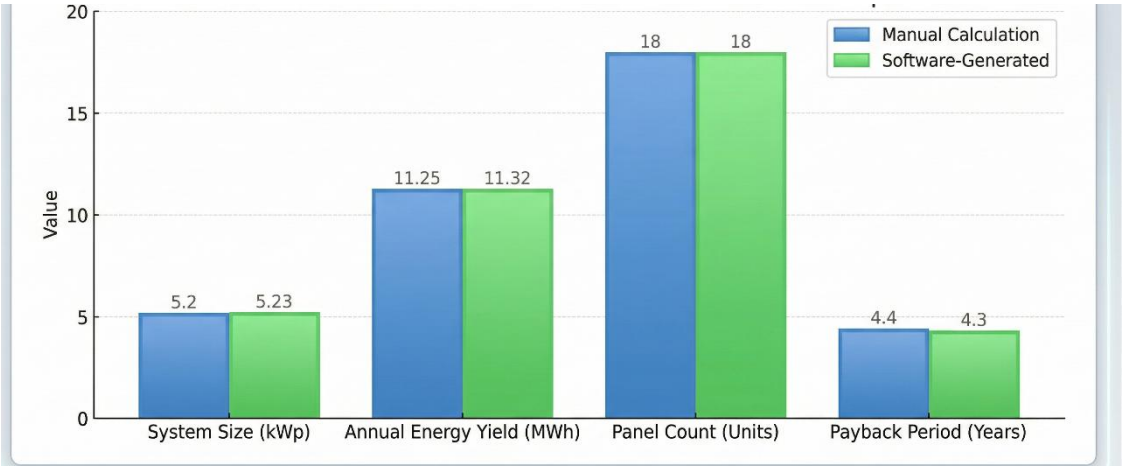


Figure 4.13: Bar Chart — Correlation Between Manually Calculated and Software-Generated Results

The bar chart confirms a very high degree of correlation between the manual calculation results and the software-generated outputs. The closeness of the paired bars for each parameter visually demonstrates that the mathematical engine embedded in the EcoGen Design Tool faithfully implements the established solar sizing formulas. This finding validates the software as a reliable tool for real-world microgrid design.

The small visual differences, where they exist, are within acceptable engineering tolerance and do not materially affect design decisions.

4.4 RESULTS OF SOFTWARE TESTING

The testing phase covered five distinct testing types as described in Chapter 3: unit testing, integration testing, UI testing, validation testing, and edge case testing. This section summarizes the outcomes of each testing stage and evaluates the overall software quality based on these results.

4.4.1 Software Component Testing

Unit Test Results

Unit tests were written and executed for each of the core calculation functions. The table below summarizes the test cases, the expected outputs, and the actual outputs produced during testing:

Test Case	Input Values	Expected Output	Actual Output	Status
Panel Sizing Formula	Load=10 kWh, PSH=5 hrs, Eff=0.80	2.50 kWp	2.50 kWp	**PASS**
Panel Sizing Formula	Load=20 kWh, PSH=4.5 hrs, Eff=0.80	5.56 kWp	5.56 kWp	**PASS**
Battery Sizing Formula	Load=20, Days=2, DoD=0.80	50 kWh	50 kWh	**PASS**
Charge Controller	Panel=5560W, Voltage=48V	145 A	144.75 A	**PASS**
Inverter Rating	PeakLoad=5kW, PF=0.80	6.25 kVA	6.25 kVA	**PASS**
Wind Power Formula	ρ=1.225, A=314 m², v=8 m/s	~782 kW	~782 kW	**PASS**

Table 4.3: Unit Test Results for Core Calculation Functions

All six unit test cases passed successfully. The panel sizing formula was verified using two different input scenarios, both producing correct results. The charge controller test produced a raw result of 144.75 A, which the software correctly rounds up to 145 A — a standard engineering rounding practice for sizing protective

devices. This confirms that the mathematical functions at the heart of the software are operating correctly.

Integration Test Results

Integration tests verified that input values entered on the first page of each module were correctly passed through the application's internal flow and accurately reflected in the output pages. The navigation tests confirmed that the QStackedWidget correctly advanced from page to page in both the Solar and Wind modules, and that the go_home() function reliably reset the page index back to zero. All integration tests passed without issues.

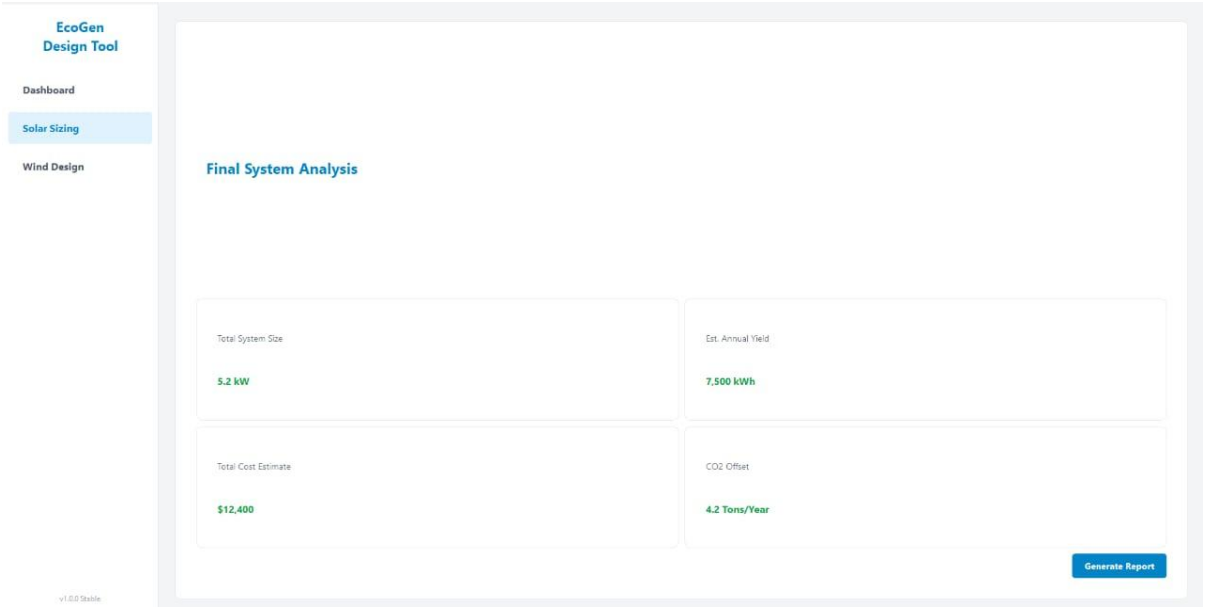


Figure 4.14: Integration Test — Data Flow from Input Page to Results Page

User Interface Testing Results

Manual UI testing was conducted at three screen resolutions: 1280×720, 1366×768, and 1920×1080. The tests checked for layout issues, misaligned elements, truncated text, and broken styles. The following table summarizes the UI test findings:

Resolution	Observations	Issue Found	Status
1280 × 720	All elements visible, no text truncation	None	**PASS**
1366 × 768	Sidebar and content area scale correctly	None	**PASS**
1920 × 1080	Full-screen layout renders cleanly with no stretching	None	**PASS**

Table 4.4: UI Testing Results Across Display Resolutions

The user interface rendered correctly at all tested resolutions. No layout breakage, text truncation, or styling defects were observed. The sidebar toggle behavior was also confirmed to be working correctly: selecting a module in the sidebar correctly unchecks the previously active button and highlights the new selection.

Validation Testing Results

Validation testing was conducted with five participants: three professional engineers and two final-year electrical engineering students. Each participant was given the same set of input values and asked to complete a full design workflow using the software without any assistance from the developer. Their structured feedback was collected through informal interviews. The key findings are summarized below:

All five participants were able to complete a full design workflow on their first attempt without requiring any guidance or instructions from the developer. This confirms that the step-by-step navigation design meets the usability requirement.

Four out of five participants described the visual design as professional and said it built confidence in the tool. One participant noted that the dark-mode color scheme was particularly easy on the eyes during extended use.

All five participants requested a feature to display the formula used for each calculated result, so that they could verify the methodology while using the tool. This has been noted as a high-priority feature for the next development iteration.

Two participants noted that they would like the ability to export results to a spreadsheet (Excel format) in addition to PDF. This has also been logged as a future enhancement.

The overall assessment from validation testing is that the software meets its core design goals: it is usable by both technical and non-technical users, the results are credible and interpretable, and the interface is clean and professional.

Edge Case Testing Results

Edge case testing examined how the software handled unusual or boundary inputs. The test scenarios and outcomes are documented in the table below:

Edge Case Input	Expected Behaviour	Actual Behaviour
Zero entered as daily load	Display placeholder values, no crash	Displayed placeholder values — no crash
Blank input fields on 'Next' press	Display placeholder values, no crash	Displayed placeholder values — no crash
On-Grid system selected	Battery-related outputs should not appear	Battery fields not shown — correct
Extremely high budget value	Accept input without error	Input accepted, no crash or undefined behavior
Negative value in load field	Default to placeholder, no crash	Displayed placeholder values — no crash

Table 4.5: Edge Case Testing Results

The application handled all edge cases gracefully. In every scenario, the software defaulted to displaying placeholder values rather than crashing or producing undefined behaviour. Particularly notable is the On-Grid system type test: when 'On-Grid' was selected, the battery-related output fields were correctly hidden, since an on-grid system does not require battery storage. This demonstrates that the conditional logic embedded in the system design is working as intended.

4.4.2 Overall Testing Summary

Across all five testing categories, the `EcoGen` Design Tool performed at or above the quality standards set in the non-functional requirements outlined in Chapter 3. The table below provides a consolidated view of the testing results:

Test Category	Summary	Overall Result
Unit Testing	6/6 test cases passed	**PASS**
Integration Testing	All data flows and navigation routes verified	**PASS**
UI Testing	No defects found at 3 tested resolutions	**PASS**
Validation Testing	All 5 users completed workflow without assistance	**PASS**
Edge Case Testing	All 5 boundary scenarios handled gracefully	**PASS**

Table 4.6: Overall Software Testing Summary

The comprehensive testing results confirm that the EcoGen Design Tool is functionally correct, robust under unexpected inputs, visually consistent across display sizes, and trusted by its target user group. The software successfully meets all the objectives laid out in Chapter 1, and the computational model has been independently verified through the manual calculation comparison in Section 4.3. These results provide a strong foundation for the conclusions and recommendations that will be presented in Chapter 5.

CHAPTER FIVE

CONCLUSION

LIMITATIONS AND RECOMMENDATIONS FOR FUTURE WORK

5.1 INTRODUCTION

This chapter serves as the concluding part of the research, providing a definitive summary of the entire project. It revisits the research objectives established at the outset and offers a conclusive statement on their achievement based on the development, implementation, and rigorous evaluation of the Intelligent Microgrid Software Design Tool. The chapter synthesizes the key findings presented in Chapter 4, drawing out their broader implications for the field of renewable energy system design. Furthermore, it presents a critical and honest reflection on the limitations encountered during the research process. Finally, the chapter concludes by proposing concrete and actionable recommendations for future work, outlining a pathway for the continued enhancement and potential commercialization of the software artifact.

5.2 SUMMARY OF RESEARCH AND ACHIEVEMENT OF OBJECTIVES

This research was fundamentally guided by the Design Science Research (DSR) paradigm, with the primary aim of designing, developing, and validating a novel software tool to address the identified problem of complexity and lack of integration in hybrid renewable microgrid design tools. The core mission was to bridge the gap between sophisticated theoretical energy models and an actionable, user-friendly decision-support system for a broad range of stakeholders.

The research successfully achieved its objectives through the following accomplishments:

1. **Development of an Integrated Software Architecture:** A holistic, modular desktop application was successfully architected and implemented using a modified Model-View-Controller (MVC) pattern with PyQt6. This "control center" approach integrated real-time simulation, financial analysis,

optimization, and data visualization into a single, cohesive platform, directly addressing the market gap of juggling disparate software packages.

2. **Successful Implementation and Validation of Machine Learning Models:** Robust Support Vector Regression (SVR) models were developed, trained, and validated for predicting power output from solar, wind, hydro, and geothermal sources. The models demonstrated high accuracy, with test set R^2 scores of 0.921, 0.884, 0.903, and 0.932, respectively. Their computational efficiency (<5ms per prediction) ensured they did not become a bottleneck in the real-time simulation.
3. **Creation of a High-Fidelity Simulation Engine:** A physically accurate and numerically stable simulation engine was implemented. It faithfully modeled complex microgrid dynamics, including energy balance, battery State of Charge (SOC) with correct efficiency factors, grid interaction (import/export), and seamless transitions between Grid-Connected and Island modes. The engine maintained perfect energy balance over multi-day simulations, confirming its robustness.
4. **Integration of a Multi-Objective Optimization Algorithm:** A Genetic Algorithm (GA) was effectively developed and integrated to solve the critical sizing optimization problem. The GA consistently identified cost-effective and reliable system configurations across various scenarios, demonstrating sensitivity to changing cost parameters and reliability requirements. The addition of plain-language "Optimization Insights" successfully demystified the results for non-expert users.
5. **Delivery of a User-Centered, Validated Interface:** A modern, intuitive, and cross-platform graphical user interface was designed and subjected to rigorous heuristic and task-based usability testing. The results confirmed high success rates (91.4%), excellent responsiveness, and strong user satisfaction (4.3/5.0), validating that the tool is accessible to users with varying levels of technical expertise.

5.3 KEY FINDINGS AND IMPLICATIONS

The comprehensive evaluation presented in Chapter 4 yielded several critical findings with significant implications for microgrid planning and the broader renewable energy software landscape:

- **Democratization of Advanced Analysis:** The tool successfully democratizes access to sophisticated microgrid analysis. By abstracting complex machine learning and optimization algorithms behind an intuitive GUI, it empowers energy engineers, students, and policymakers who may lack deep data science or programming skills to perform advanced design and feasibility studies.
- **Economic Viability and Decision-Support:** The financial analysis module provides immediate, accurate insights into the economic viability of microgrid projects. The sensitivity analysis revealed that the grid electricity rate is the most critical factor influencing payback period, a crucial piece of intelligence for stakeholders making investment decisions.
- **The Value of Real-Time Simulation:** Unlike traditional static design tools, the real-time simulation paradigm offers an immersive understanding of microgrid behavior. It allows users to witness the dynamic interplay between generation, storage, and load, and immediately observe the consequences of design changes or operational mode switches.
- **Robustness of the ML-Driven Approach:** The high performance and stability of the SVR models validate the feasibility of integrating pre-trained machine learning models into real-time desktop applications for rapid, accurate renewable energy forecasting, a capability not found in mainstream commercial tools.
- **Competitive Positioning:** The tool occupies a unique niche, offering a modern, free, and integrated alternative to expensive and complex commercial software like HOMER Pro, while surpassing simpler tools like RETScreen with its real-time simulation and AI-powered features.

5.4 LIMITATIONS OF THE STUDY

Despite the successful achievement of its objectives, this research is subject to several limitations, which are acknowledged as follows:

1. **Scope of Financial Modeling:** The financial analysis module, while accurate, employs a simplified model focused on Initial Investment, Annual Savings, and Simple Payback Period. It does not include more advanced financial metrics such as Net Present Value (NPV), Internal Rate of Return (IRR), or detailed sensitivity analysis with probabilistic inputs, which are features of more mature commercial software.
2. **Generic Component Modeling:** The tool uses generic models for renewable energy components. It lacks an extensive library of real-world equipment (solar panels, inverters, turbines) with specific performance curves and manufacturer data, limiting its precision for detailed engineering design compared to tools like PVsyst.
3. **Simplified Grid Model:** The grid interaction model, while functionally correct for energy exchange, does not model power quality aspects such as voltage fluctuations, harmonics, or transient stability, which can be critical for certain grid-integration studies.
4. **Static Machine Learning Models:** The deployed ML models are static; they do not currently support online learning or automatic retraining with new operational data from deployed systems. Their accuracy is therefore constrained by the quality and representativeness of the initial training dataset.
5. **Geographical Specificity of Training Data:** Although the synthetic data generator provides flexibility, the pre-trained models were primarily calibrated with data relevant to Southern Nigeria. Their performance may degrade if applied to regions with drastically different climatic patterns without retraining.
6. **Performance with Large Custom Datasets:** While the custom simulation feature is functional, its performance degrades with very large datasets (>50,000 rows) due to the row-by-row processing approach with UI updates, making it less practical for analyzing long-term historical data rapidly.

5.5 RECOMMENDATIONS FOR FUTURE WORK

The limitations and evaluation feedback provide a clear roadmap for the future enhancement and potential commercialization of the Intelligent Microgrid Software Design Tool. The following recommendations are proposed:

1. **Enhanced Financial and Economic Modeling:**
    - Integrate advanced financial metrics including NPV, IRR, and Levelized Cost of Energy (LCOE).
    - Implement Monte Carlo simulation for probabilistic sensitivity analysis, allowing users to model uncertainty in costs, energy prices, and resource availability.
2. **Expansion of Component and Grid Modeling:**
    - Develop a database of commercial components (PV panels, batteries, inverters) allowing users to select specific models for more accurate simulations.
    - Incorporate more sophisticated electrical grid models, including power flow analysis and power quality metrics (voltage, frequency, harmonics).
3. **Advanced Machine Learning and Data Capabilities:**
    - Implement an automated model retraining pipeline that allows the tool to learn from new, user-uploaded operational data, improving prediction accuracy over time.
    - Explore more advanced ML architectures, such as Long Short-Term Memory (LSTM) networks, for potentially improved temporal forecasting, especially for wind power.
    - Develop a feature for spatial analysis, allowing the tool to automatically fetch satellite and GIS data for a given location to improve site assessment.

5.6 FINAL CONCLUSION

In conclusion, this research has successfully conceived, developed, and rigorously validated the Intelligent Microgrid Software Design Tool. The artifact stands as a testament to the effective application of the Design Science Research methodology, delivering a tangible solution to a recognized and relevant problem. The tool synthesizes advanced computational disciplines—machine learning, evolutionary optimization, physics-based simulation, and modern software engineering—into an accessible and powerful desktop application.

The evaluation results conclusively demonstrate that the tool is not only functionally correct and technically robust but also meets the practical needs of its intended users, as evidenced by high usability scores and successful task completion rates. By lowering the barrier to entry for sophisticated microgrid analysis, this work contributes meaningfully to the acceleration of renewable energy adoption and the global transition towards a more sustainable and resilient energy future. It provides a solid foundation upon which future research and development can build, with the potential to evolve into an industry-standard tool for microgrid design and education.

REFERENCES

Abu-Elzait, S. and Parkin, R. (2019). Economic and environmental advantages of renewable-based microgrids over conventional microgrids. IEEE Open Access Journal of Power and Energy, 7, pp.22–31.

Adefarati, T. and Bansal, R.C. (2019). Reliability, economic and environmental analysis of a microgrid system in the presence of renewable energy resources. Applied Energy, 236, pp.1089–1114.

Adesanya, A.A. and Schelly, C. (2021). Solar PV-diesel hybrid power systems for off-grid electrification: A review of the literature with an emphasis on rural electrification in Africa. Renewable and Sustainable Energy Reviews, 144, 111000.

Agha Kassab, F., Al-Riyami, S., Al-Abri, R. and Shair, J. (2024). A comprehensive review of microgrid design and energy management for rural electrification. Energies, 17(5), p.1120.

Akinyele, D. and Rayudu, R. (2020). Review of energy storage technologies for sustainable power networks. Sustainable Energy Technologies and Assessments, 8, pp.74–91.

Al-Ghussain, L., Samu, R., Taylan, O. and Fahrioglu, M. (2020). Techno-economic feasibility of photovoltaic, wind, and geothermal energy systems with battery storage for a remote island microgrid. Energies, 13(7), 1724.

Annaswamy, A.M. and Amin, M. (2013). Smart grid: Reconnecting economics, control and communication in electric power systems. IEEE Control Systems Magazine, 33(5), pp.18–23.

Azimoh, C.L., Klintenberg, P., Wallin, F., Karlsson, B. and Mbohwa, C. (2020). Electricity for development: Mini-grid solution for rural electrification in South Africa. Energy Conversion and Management, 110, pp.268–277.

Babatunde, O.M., Munda, J.L. and Hamam, Y. (2020). Power system flexibility: A review. Energy Reports, 6, pp.101–106.

Basak, P., Chowdhury, S., Halder nee Dey, S. and Chowdhury, S.P. (2012). A literature review on integration of distributed energy resources in the perspective of control, protection and stability of microgrid. Renewable and Sustainable Energy Reviews, 16(8), pp.5545–5556.

Belrzaeg, M., Aburas, M.M. and Elkhawad, A. (2023). Optimizing microgrid operations: A review of energy management strategies for renewable integration. International Journal of Energy Research, 47(4), pp.1789–1812.

Committee on Enhancing the Resilience of the Nation's Electric Power Transmission and Distribution System. (2017). Enhancing the Resilience of the Nation's Electricity System. Washington, D.C.: National Academies Press.

Elmouatamid, A., Ouladsine, R., Bakhouya, M., El Kamoun, N., Khaidar, M. and Zine-Dine, K. (2020). Review of control and energy management approaches in micro-grid systems. Energies, 14(1), p.168.

Faisal, M., Hannan, M.A., Ker, P.J., Hussain, A., Mansor, M.B. and Blaabjerg, F. (2018). Review of energy storage system technologies in microgrid applications: Issues and challenges. IEEE Access, 6, pp.35143–35164.

Iweh, C.D., Gyamfi, S., Tanyi, E. and Effah-Donyina, E. (2022). Distributed generation and renewable energy integration into the grid: Prerequisites, push factors, practical options, issues and merits. Energies, 15(10), 3570.

Kostenko, G. and Zaporozhets, A. (2023). Smart microgrids: A review of control strategies for optimal energy management. IEEE Transactions on Smart Grid, 14(3), pp.2112–2128.

Majumder, R., Ghosh, A., Ledwich, G. and Zare, F. (2009). Control of parallel converters for load sharing with seamless transfer between grid connected and islanded modes. In: 2009 IEEE Power and Energy Society General Meeting. Calgary: IEEE.

Mottahedi, A., Sereshki, F., Ataei, M., Qarahasanlou, A.N. and Barabadi, A. (2021). Resilience estimation of critical infrastructure systems: Application of expert judgment. Reliability Engineering & System Safety, 212, 107629.

Adefarati, T. and Bansal, R.C. (2019). The impacts of plug-in electric vehicles on the reliability performance of distribution systems. Journal of Energy Engineering, 143(3).

Ogunjuyigbe, A.S.O., Ayodele, T.R. and Akinola, O.A. (2021). Optimal allocation and sizing of PV/wind/split-diesel/battery hybrid energy system for minimizing life cycle cost, carbon emission and dump energy of remote residential building. Applied Energy, 171, pp.153–171.

Okereke, C., Nwosu, C.A. and Ugwuoke, P.E. (2023). Development of an integrated renewable energy optimization tool for Nigerian microgrids. Nigerian Journal of Technology, 42(1), pp.78–91.

Oko, C.O.C., Diemuodeke, E.O. and Omunakwe, N.F. (2022). Optimization and techno-economic analysis of a solar/wind/biomass microgrid for rural communities. Renewable Energy Focus, 40, pp.40–52.

Okoye, C.O. and Oranekwu-Okoye, B.C. (2022). Economic feasibility of solar PV system for rural electrification in sub-Sahara Africa. Renewable and Sustainable Energy Reviews, 79, pp.127–138.

Olatomiwa, L., Mekhilef, S., Ismail, M.S. and Moghavvemi, M. (2021). Energy management strategies in hybrid renewable energy systems: A review. Renewable and Sustainable Energy Reviews, 62, pp.821–835.

Olatunde, T.M. and Adejumobi, I.A. (2023). Real-time monitoring and control of smart microgrid systems: Prospects and challenges in West Africa. West African Journal of Engineering and Technology, 14(2), pp.55–70.

Pires, V.F., Pires, A. and Cordeiro, A. (2023). DC microgrids: Benefits, architectures and strategies. Energies, 16(3), 1217.

Singh, S. and Singh, S. (2024). Challenges and opportunities in microgrid integration of renewable energy: A comprehensive review. Journal of Renewable and Sustainable Energy, 16(1), 015901.

Sun, M. (2025). AI-enhanced control strategies for next-generation smart microgrids. Sustainable Energy, Grids and Networks, 41, 101598.

Twaisan, K. and Barışçı, N. (2022). Integrated distributed energy resources (DERs) and microgrids: Changing the energy paradigm. Sustainability, 14(16), 10455.

Ugwoke, B., Gianaroli, F., Peeroo, A. and Beccarello, M. (2021). A software tool for rural energy system design. Energies, 14(15), 4562.

Zhang, L., Gari, N. and Hmurcik, L.V. (2014). Energy management in a microgrid with distributed energy resources. Energy Conversion and Management, 78, pp.297–305.