H2PCSim Web Tool Report

Supervisor: Dr. Hossam Gaber

Date: July 16th, 2024

How to Download and Run project:

Frontend/User Interface:

* Install Node.js and NPM, you can check using “node -v” and “npm -v” in the terminal to verify if it is installed.
* Clone the GitHub repo link into your directory or download the project zip from the file attached in this folder.
* Navigate to the project directory in the terminal using “cd” to the “EnergyUI” directory.
* Install dependencies using “npm install”
* Once dependencies are installed, run the command “npm run dev” to launch the web app locally.

Backend/Server:

* Similar to the above steps, make sure Python and pip are installed on your system.
* Once installed, clone or download the “energy-backend” folder into your directory
* Navigate into the “energy-backend” directory with the terminal.
* To install required dependencies, run this command “pip install -r requirements.txt”
* After all dependencies are installed, run “python src/app.py” to run the server.

Database:

* Have MySQL and MySQL workbench installed, the default installations for both should be enough.
* Once downloaded, go to the MySQL import wizard, and upload the database schema from this project zip into it. It will generate the designed database. Configure your root and password settings accordingly and change it in the backend “app.py” file when done.

Project Structure:

Database:

* Managed using MySQL, database information can be found in Omar’s project repository. In this project zip, there will also be a DB Schema file, which can be inserted into MySQL workbench, and be configured for this project.

Server/Backend:

* Managed using Flask / Python and using SQLAlchemy as an ORM. This backend consists of 2 main files currently, model.py and app.py.
  + Model.py:

Used only to create and describe the object definitions of each SQL table using SQLAlchemy. Makes it easier to work with and is more inline with Python’s object oriented style.

* App.py:

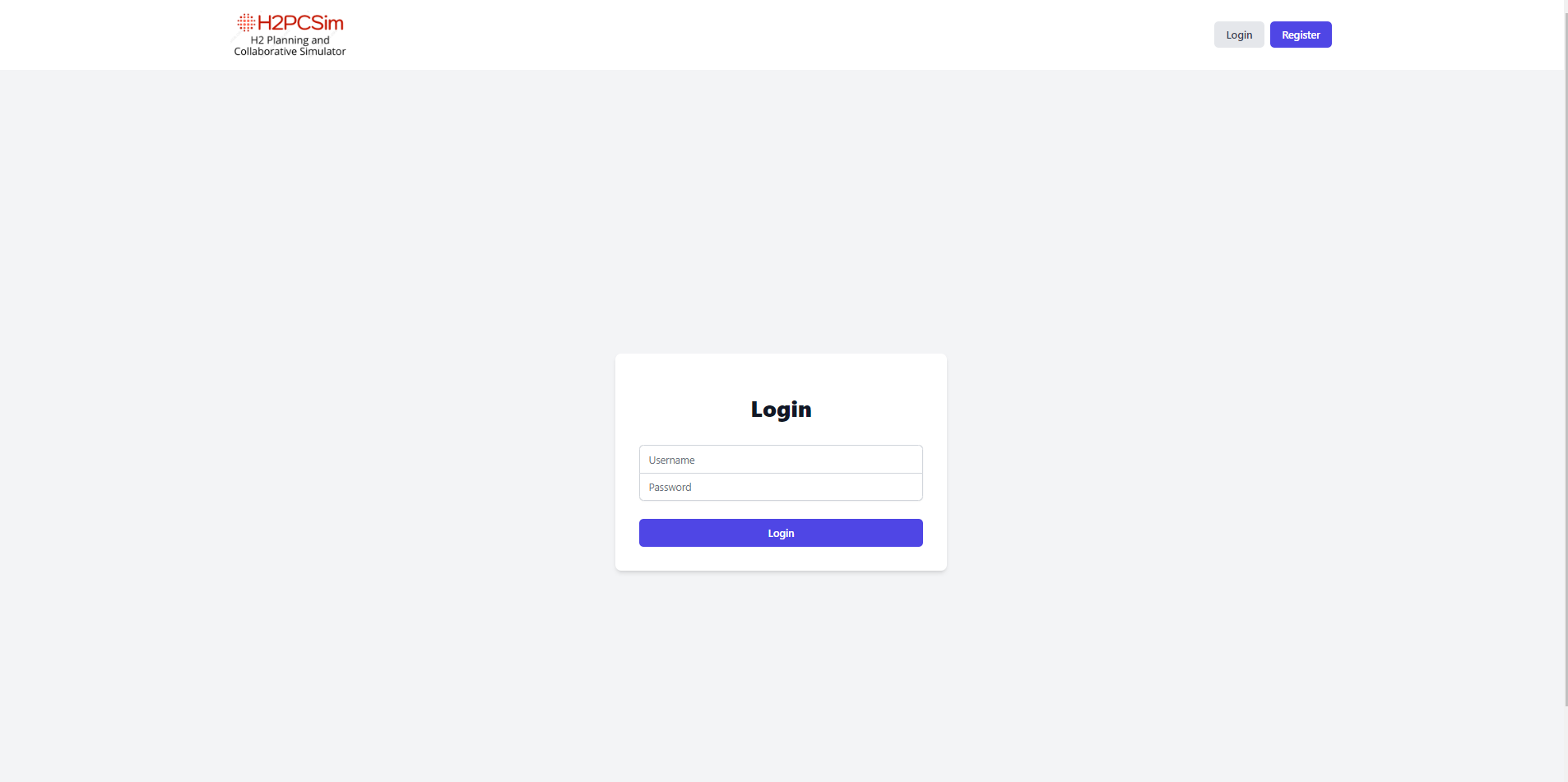
Used to create and handle the API requests made to the server. When running the server, **the entry point would be by running App.py. You would also change the MySQL user/password data here.** Various API routes are defined in this file, such as registration routes, login routes, CSV data uploads, etc. The main API route is the “save\_model” route, which handles data uploads from the PSM section of the User Interface, and saves it to the database.

User Interface/Frontend:

* Developed using React / Javascript components. The main sections will be defined below:

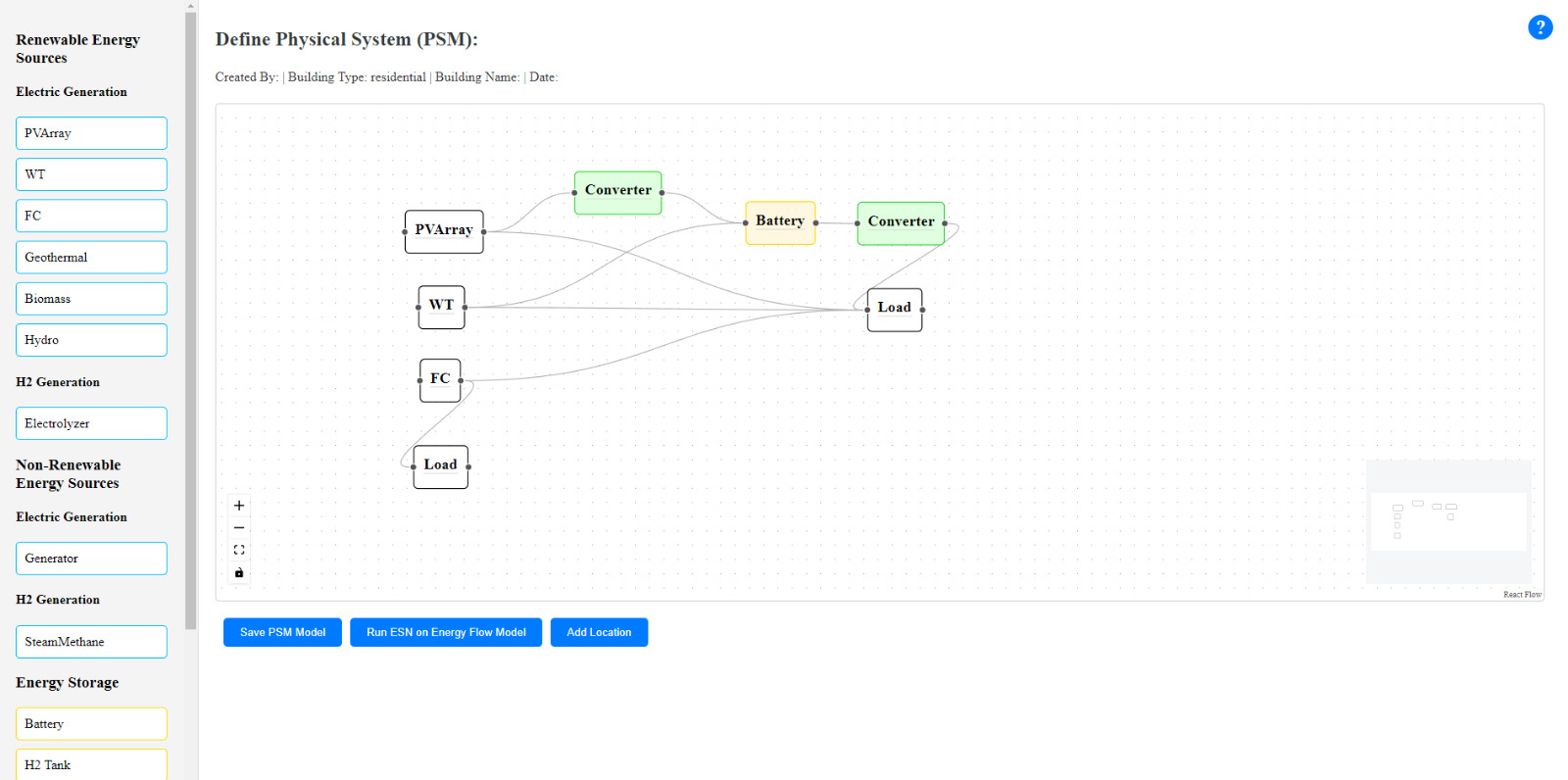
Login/Registration:

* Standard login and registration, usernames and passwords are stored directly into the database upon registration, **and are not salted and hashed.** When a user successfully logs in, the user ID is returned to the front end and is stored in the browser’s session storage. This can be used for requesting user specific information if necessary, and is used to link Energy flows defined in the PSM to the user who created them.



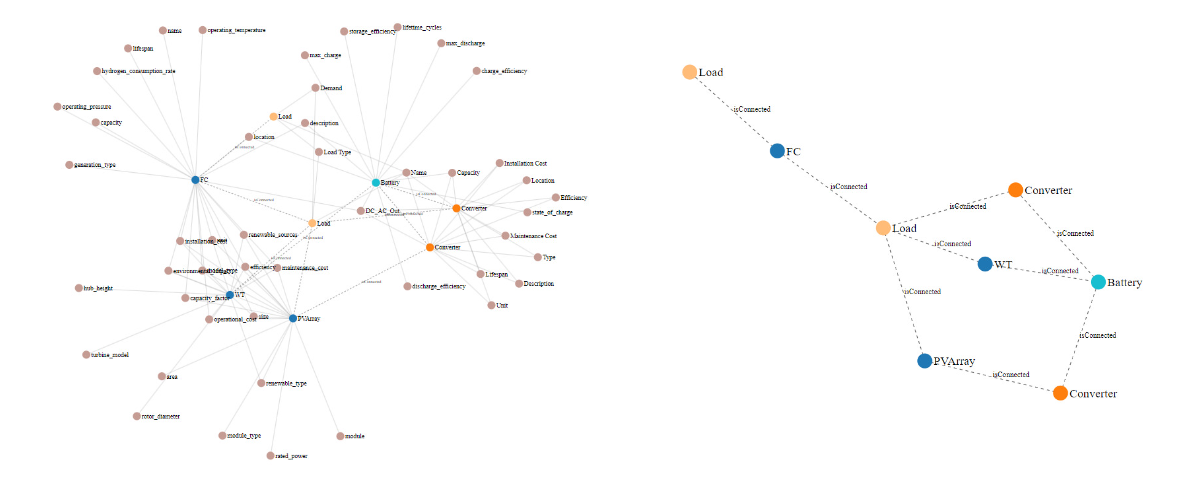
PSM:

* Consists of the PSM component, where all the logic and data related to the drag and drop modules using React-Flow are found. Pop-up forms are also located within this module such as the location pop up and the site pop up when the page is first opened.. When a user saves or moves to the ESN section using the buttons, the backend API is called to save all data entered. All node parameters are defined to match the meta-class parameter outlines.



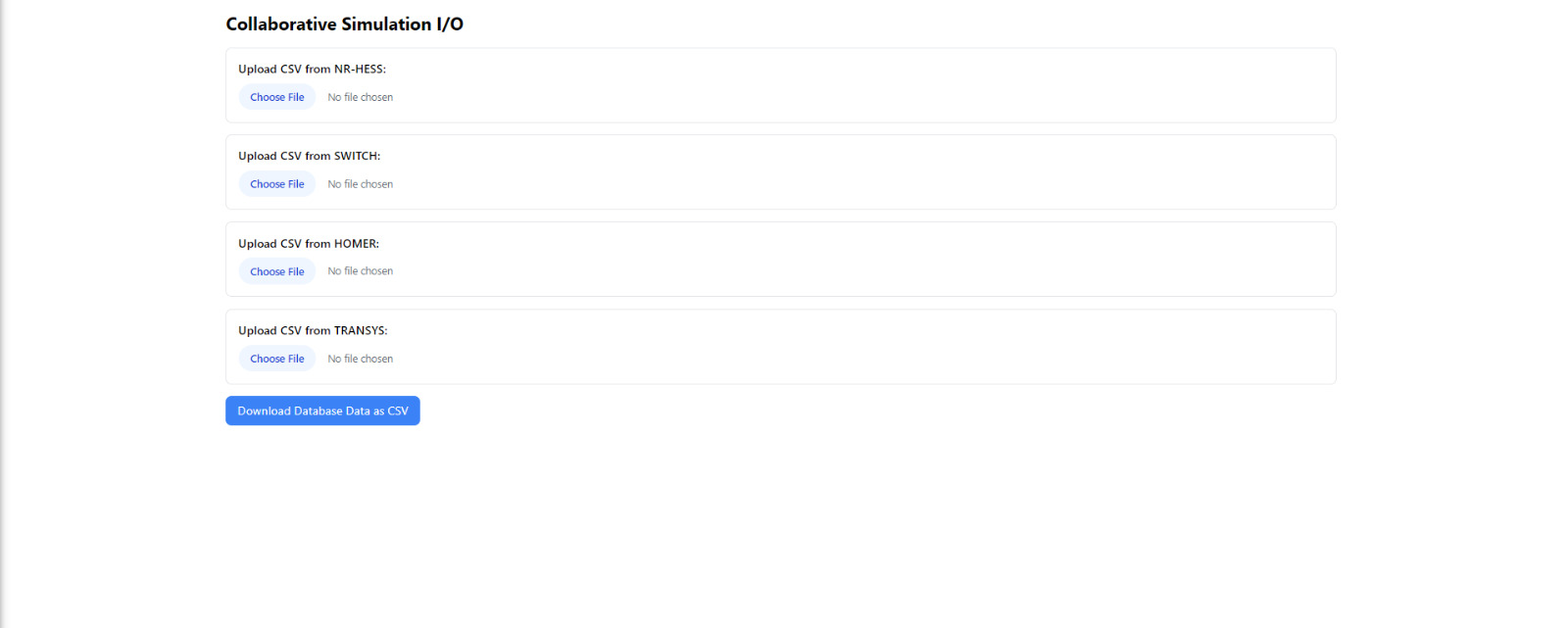
ESN:

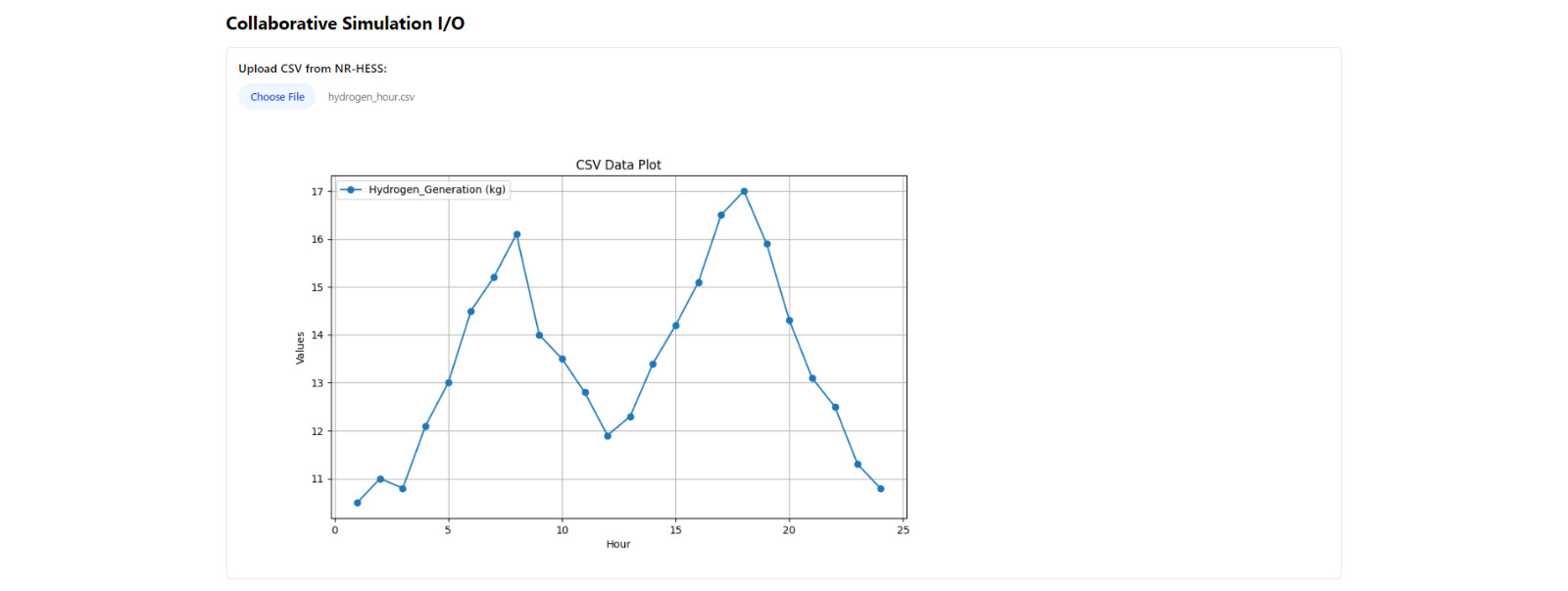
* Uses D3.js to visualise the created Energy Flow diagram defined in the PSM section. Contains a detailed and non-detailed view, where the detailed view contains all parameter nodes, value nodes and component nodes defined in the PSM. The non-detailed view represents only the component nodes and their relations.



Collaboration:

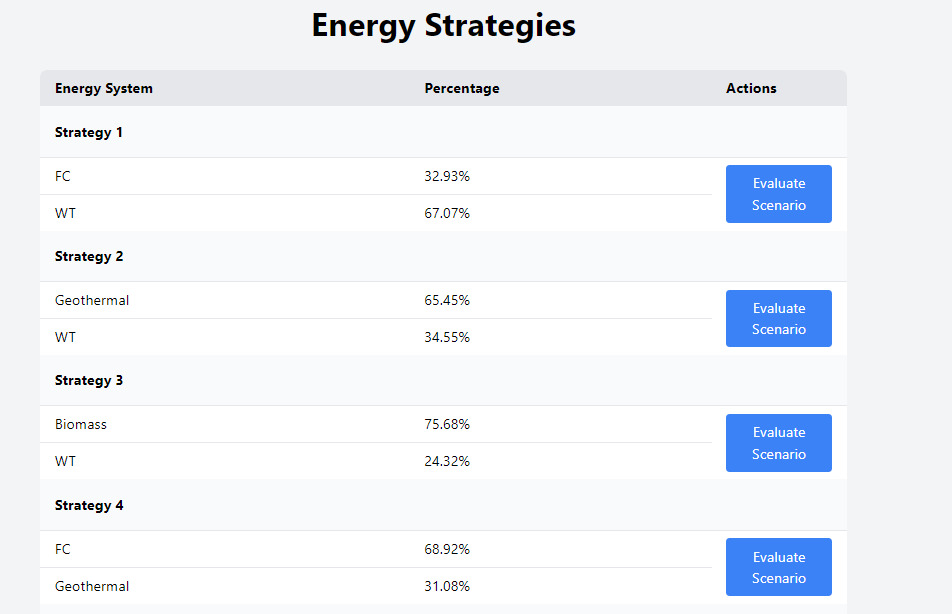
* Uses CSV file uploads and downloads to communicate between the different tools. When the download data button is clicked, it will download a CSV file with all the data stored in the database. This CSV file data **has not** been adjusted to fetch only data relevant to the user, such as fetching data only by user ID or site ID. When the upload CSV file data button is clicked, the website will request a CSV file to be uploaded, which will then pass that CSV file data back to the server where it is processed using matplotlib as a graph, and sent back to the user interface to be displayed to the user.





Scenarios:

* Using the defined data from the Energy Flow model made in the PSM, all the relevant energy sources and storage components are stored in an array. Then all possible combinations of these components are generated to create possible scenarios.



Functional Models and Features:  
  
Login/Registration Functional Model:

| Main Function | Sub-Function | Detailed Function |  |
| --- | --- | --- | --- |
| F1. User Interface and Server - Login Page | F1.1 Receive login data and validate | F1.1.1 Recieves user inputs and validates or rejects user authentication. | Retrieves username and password and checks database for matching credentials. If successful, a token is assigned that will link any created models to that user. |
| F2 User Interface and Server - Registration Page | F2.1 Receive Registration data and validate | F2.1.1 Receives user inputs and validates before registering a user. | Retrieves username, password and roles and checks if valid. If valid, will update the users table. |

PSM Functional Model

| Main Function | Sub-Function | Detailed Function |  |
| --- | --- | --- | --- |
| F1. User Interface - Drag and Drop Components | F1.1 Build Energy Flow Diagram using React-Flow | F1.1.1 Define Nodes | Energy Systems  Energy Storage  Load  Converters and Inverters |
|  |  | F1.1.2 Define Relationships | Link between Components:  Ex. Connection between Energy Systems and Converters/Inverters |
| F2. User Interface - Add Meta-Class Data and parameters | F2.1 Add Site Data | F2.1.1 Define Site Data parameters and update database | Receives User input and saves data to be sent to the backend |
|  | F2.2 Add Location Data | F2.2.1 Define Location Data parameters and update database | Receives user input and saves data to be sent to the backend |
| F3. Server and Database - Store PSM data | F3.1 Receive data from UI and update database | F3.1.1 Define database model and insert data into tables | Retrieves data from UI, and updates respective tables and join tables with all energy parameters. |

ESN Functional Model:

| Main Function | Sub-Function | Detailed Function |  |
| --- | --- | --- | --- |
| F1. User Interface - Generate Network Visualization | F1.1 Define Energy Graph using D3.js | F1.1.1 Represent the Component Nodes | Create nodes for each of the components in the Energy Flow Diagram |
|  |  | F1.1.2 Represent the Parameter Nodes | Create nodes for each of the component nodes parameters |
|  |  | F1.1.3 Represent the value nodes | Create nodes for each of the values linked to each parameter node if applicable. |
| F2. User Interface - Toggle Detailed and Non-Detailed Views | F2.1 Toggle Detailed View | F2.1.1 Represents all the Components, Parameter and value nodes for the network | Displays all nodes along with the relations between each node. |
|  | F2.2 Toggle Non-Detailed View | F2.2.1 Represents only the component nodes and the relationships between those nodes | Displays component nodes and relationships only as defined in the PSM. |

Strategies Functional Model:

| Main Function | Sub-Function | Detailed Function |  |
| --- | --- | --- | --- |
| F1. Server - Display All Strategies | F1.1 Get all Energy Systems | F1.1.1 Retrieves only the energy sources and energy storage items | Pulls this data from the Server based on the created Energy Flow Diagram. |
|  | F1.2 Generate all Combinations | F1.2.1 Generates all combinations of the energy systems | Calculates all combinations of energy systems and displays it in the User Interface. |

Collaboration Functional Model:

| Main Function | Sub-Function | Detailed Function |  |
| --- | --- | --- | --- |
| F1. User Interface and Server - Upload Files from various tools (ex. NR-HESS) and download input data from database | F1.1 Receives CSV files as input | F1.1.1 Upload the output files from the tools, such as NR-HESS, and send them back to the server | Once CSV data is received, the server generates a chart with the data and sends it to the User Interface |
|  | F2.1 Send input to data as a CSV file to User | F2.1.1 Users can download all database parameters as CSV files | Data is retrieved from the database by the server and sent to the User Interface as a downloadable CSV, which can be used as input data into the other tools. |