Energy Consumption Tracker – Project Documentation

# 1. Introduction & Objective

The Energy Consumption Tracker is a web application that allows users to track, monitor, and visualize energy usage across various devices or household categories. This tool aims to raise awareness and help optimize energy consumption. It provides an intuitive interface to input data, view device-wise usage through dynamic charts, and manage entries efficiently.

# 2. Why I Chose This Topic

With increasing emphasis on sustainable energy use and electricity cost management, developing an energy consumption tracker was a practical and relevant idea. It allowed me to explore both frontend and backend technologies while creating a useful tool. Additionally, the topic aligns with environmental concerns and can be scaled for future applications.

# 3. Technologies Used

The project is implemented using the following technologies:  
- HTML, CSS, JavaScript: For creating the frontend interface  
- Chart.js: For data visualization using dynamic charts  
- Haskell with Scotty: Backend server for API endpoints  
- Aeson: Haskell library for JSON parsing and encoding  
- IORef: Used for temporary in-memory state management  
- Network.Wai & Middleware.CORS: Handles cross-origin resource sharing (CORS)  
- Cabal, GHCup: Haskell build tools used on Windows

# 4. HTML and CSS Explanation

The HTML structure is divided into semantic sections including a header, dashboard, chart containers, forms, and a history table. CSS is used extensively to make the layout responsive using CSS Grid and media queries. Custom CSS variables improve readability and reusability.

# 5. JavaScript Functionality

JavaScript handles data input, DOM manipulation, and communication with the backend using fetch API. Form submissions dynamically update the history table and device-wise pie chart. Chart.js is used to display live consumption metrics.

# 6. Backend (Main.hs) Deep Dive

The backend is written in Haskell using the Scotty web framework. It exposes three main routes:  
- GET /data: Fetch all entries  
- POST /data: Add a new entry  
- POST /clear: Clear all entries  
The data is stored in memory using IORef. JSON is parsed using Aeson, and CORS is handled using Network.Wai.Middleware.Cors.

# 7. Key Code Snippets

Example: Defining the EnergyData type and enabling JSON support

data EnergyData = EnergyData  
 { device :: Text  
 , consumption :: Int  
 } deriving (Show)  
  
instance ToJSON EnergyData where ...  
instance FromJSON EnergyData where ...

This defines a custom type with two fields and makes it compatible with JSON. This structure is used throughout the backend to send and receive device consumption data.

# 8. Challenges Faced & Solutions

- Configuring CORS correctly to avoid frontend request errors  
- Handling form reset and live chart update after each submission  
- Getting Chart.js to work dynamically based on backend responses  
- Setting up Cabal, GHCup, and dependencies on Windows

# 9. Conclusion & Future Scope

The Energy Tracker project successfully integrates frontend and backend to provide a responsive and interactive experience. I learned about Haskell web development, JSON handling, state management using IORef, and DOM + chart manipulation in JS.  
  
Future enhancements could include:  
- Adding persistent database storage  
- User authentication  
- Real-time updates via WebSockets  
- Deployment on a live server