**Home Energy Optimization Advisor**

1. **Problem Scope**

**Main Goal**: Help users optimize their energy usage with minimal manual input by providing personalized recommendations.

**Key Outputs**:

* Estimate appliance energy consumption and monthly costs.
* Predict potential savings from optimized energy usage.
* Recommend energy-efficient habits or appliances.

**Input Data**:

* User's total energy bill (or usage data if available).
* List of appliances (can be selected via dropdown).
* Location for electricity cost and climate data.

**Deliverables:**

* A functional web app with a clean UI for data entry.
* ML-based insights and recommendations on energy usage.

**Step 2: Gather and Prepare Data**

Start by building a dataset for training your ML models.

**A. Datasets to Use**

1. **Appliance Energy Consumption**:
   * Use public datasets like:
     + UK Domestic Energy Data.
     + [UCI Appliance Energy Prediction Dataset](https://archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction).
   * Include details like power ratings (kW), average usage patterns (hours/day), and efficiency ratings.
2. **Electricity Pricing**:
   * Retrieve average electricity costs from:
     + Energy Information Administration (EIA).
     + Regional or country-specific sources for other locations.
3. **Weather Data** (optional):
   * Use APIs like OpenWeather to pull climate data for the user’s location.

**B. Example Dataset Format**

Structure the data into a tabular format for ML models:

| **Appliance** | **Power Rating (kW)** | **Avg. Usage (hours/day)** | **Efficiency Rating** | **Typical Monthly Cost ($)** |
| --- | --- | --- | --- | --- |
| Refrigerator | 1.2 | 24 | A | 36 |
| Washing Machine | 0.5 | 3 | B | 7.5 |
| Air Conditioner | 1.8 | 8 | C | 43.2 |

**Step 3: Design the User Workflow**

1. **Input Fields**:
   * Electricity bill (e.g., $120/month).
   * Select appliances from a dropdown or checklist (e.g., fridge, AC, washing machine).
   * Location (auto-detected via IP or manual input).
2. **Backend Processing**:
   * Use a pre-built appliance database to infer missing data (e.g., power rating, efficiency).
   * Estimate energy consumption and cost for each appliance.
   * Run ML models to make predictions and generate recommendations.
3. **Output to the User**:
   * Energy consumption breakdown (pie chart or bar graph).
   * Savings recommendations (e.g., "Switch to energy-efficient fridge to save $25/month").

**Step 4: Build the Backend**

**A. Preprocessing and Feature Engineering**

* Write Python scripts to:
  1. Load appliance data and preprocess it (e.g., normalize power ratings, fill missing values).
  2. Merge user inputs with default data to create a complete dataset.
  3. Calculate derived features, such as:
     + Total energy consumption = Power Rating × Usage × Days Active.

**B. Train ML Models**

1. **Regression Model for Energy Consumption Estimation**:
   * Use a dataset with appliances, usage patterns, and energy consumption to train the model.
   * Example: Predict how much energy each appliance uses given its type and user habits.

**Recommendation System**:

* + Use collaborative filtering or simple rule-based logic to suggest changes based on user data.
  + Example: "If appliance efficiency is rated C or lower, recommend an upgrade."

**Step 5: Build the Frontend**

**Tools:**

* Use **React** or **Vue.js** for the user interface.
* Integrate visualizations with **Chart.js** or **D3.js** for showing energy usage and savings.

**Key Pages:**

1. **Home**:
   * Input form for electricity bill, appliance list, and location.
2. **Results**:
   * Graph showing energy consumption breakdown.
   * Recommendations (e.g., "Reduce AC usage during peak hours").
3. **Savings Tracker** (Optional):
   * Track user’s progress over time.

**Step 6: Deploy the Web App**

1. **Backend Deployment**:
   * Use **Flask** or **FastAPI** to serve your ML models as APIs.
   * Example API endpoints:
     + /predict: Accept user inputs and return energy consumption estimates.
     + /recommend: Provide personalized savings recommendations.
2. **Frontend Hosting**:
   * Use platforms like **Vercel** or **Netlify** for the frontend.
   * Connect to the backend via REST APIs.
3. **Database**:
   * Use **PostgreSQL** or **MongoDB** to store user inputs and appliance data.
4. **Cloud Hosting**:
   * Host the backend and database on AWS, Google Cloud, or Heroku.

**Step 7: Iterate and Improve**

1. Collect user feedback to refine the input process and recommendations.
2. Add more features over time, such as:
   * Integration with smart meters for real-time data.
   * Advanced time-series predictions for seasonal trends.