Smart Home Energy & Water Consumption Dashboard — Capstone Report

Title: Smart Home Energy & Water Consumption Dashboard using MERN Stack

Team Members: Prince Kumar, Prateek Raj, Sumit Kumar, Shivam Prakash, Divyanshu Singh, Aditya Verma

Course: Capstone Project-I CSE339

Supervisor: Cherry Khosla

Institution / Department: Lovely Professional University/CSE

INDEX

1. INTRODUCTION

Modern households often lack fine-grained visibility into electricity and water usage. This project builds a web-based dashboard using the MERN stack to monitor, analyze and optimize energy and water consumption for individual appliances. It integrates (real or simulated) IoT sensor data, visualizes consumption trends, issues alerts, and offers AI-powered recommendations to reduce waste and costs. This aligns with sustainability goals and Smart India initiatives.

1.1 Objective of the project

The main objective of this project is to design and develop a smart monitoring system that provides households with clear, real-time insights into their electricity and water consumption. The system aims to empower users by offering actionable information, detailed analytics, and AI-driven suggestions that promote efficient resource usage and cost savings. To achieve this, the project focuses on integrating IoT/sensor data, intuitive visualizations, and intelligent decision-support features.

The key objectives are:

- **Provide real-time monitoring** of appliance-level energy (kWh) and water (litres) consumption, allowing users to track usage instantly and identify abnormal activity.
- **Visualize consumption trends** through daily, weekly, and monthly charts, enabling users to understand patterns, detect peaks, and compare usage across appliances.
- **Generate automated alerts** when consumption exceeds user-defined thresholds, ensuring timely action to prevent overuse or unexpected utility costs.
- **Offer AI-based predictions and personalized recommendations** based on historical data, helping users optimize appliance usage and improve long-term efficiency.
- **Store and manage historical data** in a structured format to support trend analysis, year-over-year comparisons, and data-driven decision-making.

This combination of monitoring, analytics, predictions, and automated alerts forms a comprehensive solution that enhances user awareness, promotes sustainable consumption habits, and contributes to a smarter, more efficient home environment.

1.2 Description of the project

This project focuses on building a smart, web-based monitoring system using the MERN stack to track and analyse 3household energy and water consumption at the appliance level. The system integrates IoT sensor data (or simulated inputs) with a modern, interactive dashboard,

providing users with meaningful insights, alerts, and AI-driven recommendations. By combining real-time monitoring, intelligent analytics, and user-friendly visualization, the project aims to promote efficient resource usage and support sustainable living practices.

The major components of the system include:

- **Sensors / Data Source**
  IoT sensors or simulated APIs continuously stream real-time appliance-level consumption data for electricity (kWh) and water (litres), serving as the primary data input for the system.
- **Backend (Node.js + Express)**
  - Handles data ingestion, cleaning, and normalization.
  - Stores all processed sensor readings, alerts, and user data in **MongoDB**.
  - Exposes secure RESTful APIs for the frontend and prediction modules.
- **Frontend (React Web Dashboard)**
  - Provides users with an intuitive interface for monitoring live consumption.
  - Displays interactive charts, tables, and visual analytics.
  - Includes user profile settings, appliance management, and downloadable reports.
- **AI Module**
  - Implements machine learning or time-series forecasting to predict future consumption.
  - Generates personalized optimization recommendations based on historical patterns and trends.
- **Notification System**
  - Sends alerts via in-app messages, email, or push notifications when abnormal or excessive consumption is detected.
  - Helps users respond quickly to potential wastage or appliance malfunction.

Together, these components form a comprehensive smart home monitoring solution that enhances user awareness, reduces unnecessary resource usage, and supports long-term sustainability goals.

1.3 Scope of the project

The scope of this project is centred around developing a complete smart monitoring solution capable of tracking, analysing, and predicting energy and water consumption in a household environment. The system is designed to be scalable, modular, and user-friendly, ensuring that users can easily understand their consumption patterns and take informed decisions. The project covers all major aspects including data ingestion, analytics, visualization, and notifications.

The defined scope includes:

- **Real-time monitoring** of individual appliance-level electricity (kWh) and water (litre) usage.

- **Interactive dashboard visualizations** for daily, weekly, and monthly trend analysis.
- **Centralized data storage** of all consumption records using MongoDB.
- **AI-driven forecasting and recommendations** for usage optimization.
- **User-configurable threshold alerts** to detect excessive or unusual consumption.
- **Integration with IoT sensors or simulated data sources** for flexible testing.
- **User authentication, profile management, and appliance management** within the frontend interface.

1.3.1 Use Case Model

The Use Case Model represents how different users interact with the Smart Home Energy and Water Monitoring System. It identifies the primary actors, their roles, and the major system functionalities they access. This model helps in understanding user expectations, system boundaries, and key interactions that guide later design and implementation stages.

The system supports multiple user categories, each performing specific actions based on their role. The following use cases summarize the core interactions within the system:

Actors

- **User (Homeowner / Resident)**
  Interacts with the dashboard to monitor appliances, configure alerts, and view analytics.
- **Admin**
  Manages user accounts, oversees system health, and ensures data integrity.
- **IoT Sensor / Data Source**
  Automatically sends consumption readings to the backend in real time.
- **AI Module**
  Generates predictions, trends, and energy-saving recommendations.

Primary Use Cases

- **Login / Authentication**
  Users securely log in to access their personalized dashboard and data.
- **View Real-Time Consumption**
  Monitor current energy and water usage for each connected appliance.
- **Manage Appliances**
  Add, modify, or remove appliances linked to sensors.
- **Configure Threshold Alerts**
  Set consumption limits and receive notifications when thresholds are exceeded.

- **View Historical Trends**
  Explore daily, weekly, and monthly usage analytics.
- **Receive Recommendations**
  Access AI-driven suggestions to optimize consumption and reduce waste.
- **Admin: User & System Management**
  Admin oversees new registrations, monitors system performance, and resolves data-related issues.
- **Sensor Data Transmission**
  Sensors push real-time consumption data to the backend for processing and storage.

1.3.2 Use Case Diagram

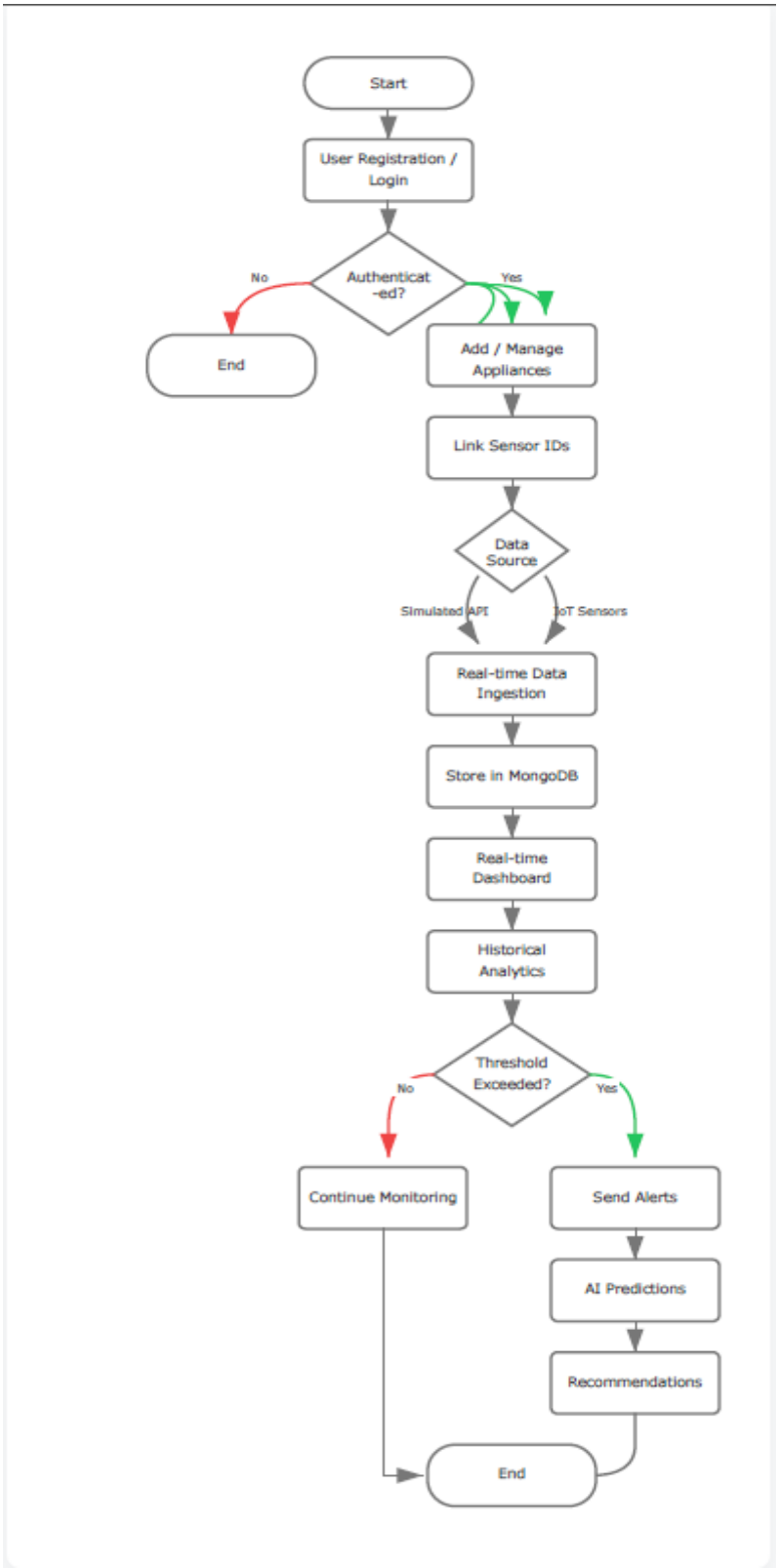

Figure – 1 (Flow Chart)

2. SYSTEM DESCRIPTION

The system follows a structured three-layer architecture designed to efficiently process sensor data, manage application logic, and deliver an intuitive user experience. Each layer plays a specific role in ensuring the reliability, performance, and scalability of the application. The communication flow between components is optimized for real-time data transfer while maintaining high security standards.

The architecture consists of the following layers:

• Presentation Layer (React)

- Implements the user-facing dashboard with charts, analytics, alerts, and reports.
- Provides an interactive interface for monitoring real-time consumption and managing appliances.
- Handles user authentication, settings, and visualization of predictions and historical data.

• Application Layer (Node.js / Express)

- Acts as the central processing unit of the system.
- Manages ingestion of sensor data, applies business logic, formats responses, and communicates with the AI prediction module.
- Exposes secure REST APIs and WebSocket channels to the frontend.

• Data Layer (MongoDB)

- Responsible for storing all consumption records, appliance details, thresholds, alerts, and prediction data.
- Handles time-series data efficiently and supports fast retrieval for analytics and reporting.
- Ensures flexibility through a NoSQL document-based structure.

Communication Flow

To ensure seamless, real-time data processing, the system relies on multiple communication channels:

- **Sensor → Backend (MQTT / HTTP / WebSocket)**
  Sensors send continuous consumption readings using lightweight protocols suitable for IoT environments.
- **Backend → Database (MongoDB)**
  All incoming data is validated, structured, and stored securely in MongoDB collections.

- **Backend → Frontend (REST / WebSocket)**
    - REST APIs deliver processed data, charts, historical logs, and predictions.
    - WebSockets push real-time updates directly to the dashboard.

Security Measures

The system incorporates multiple layers of security to protect user data, prevent unauthorized access, and ensure safe API interactions:

- **JWT-based authentication** for secure login and session handling.
- **HTTPS encryption** to safeguard data transmission between client and server.
- **Input validation and sanitization** to prevent injection attacks.
- **Rate limiting** to block excessive or malicious API calls.

These measures collectively ensure a secure and dependable environment for users.

2.1 Customer/User Profiles

The system is designed to support a diverse set of users with varying needs. Each profile benefits in a unique way from the platform's monitoring, analytics, and prediction features.

• Eco-Conscious Homeowner

- Interested in reducing electricity and water usage.
- Uses the system to identify high-consumption appliances and optimize daily resource consumption.
- Benefits from alerts and AI recommendations that support sustainable living.

• Technophile

- Enthusiastic about smart home technologies and IoT integration.
- Actively monitors real-time energy data and analyzes performance trends.
- Uses advanced features like predictions, alerts, and system customization.

• Property Manager (Future Scope)

- Oversees multiple residential units or rental properties.
- Would use aggregated dashboards to compare and manage consumption across different homes.
- Helps identify units with abnormal usage or maintenance issues.

2.2 Assumptions and Dependencies (If applicable)

The successful functioning of the Smart Home Energy and Water Monitoring System relies on several technical and environmental assumptions. These assumptions define the conditions under which the system is expected to operate, while the dependencies outline the external tools, libraries, and resources required for the platform to run effectively. Understanding these factors ensures a realistic implementation plan and clarifies the boundaries within which the system performs optimally.

Assumptions

The system is developed assuming that the following conditions are met:

- **Availability of Sensors or APIs**
  Reliable IoT sensors or simulated APIs are available to continuously provide real-time appliance-level consumption data.
- **Internet-Enabled Home Environment**
  Users have stable internet connectivity to support real-time data streaming, dashboard updates, and cloud database access.
- **User Devices Support Modern Browsers**
  The frontend requires browsers that support React, WebSockets, and modern JavaScript features.

Dependencies

The project depends on specific technologies and libraries essential for proper system development and deployment:

- **MongoDB (Atlas or Local Instance)**
  Serves as the main database for storing time-series data, user profiles, appliance details, alerts, and predictions.
- **Node.js and Express**
  Required for backend development, API creation, sensor data ingestion, and system logic execution.
- **React.js**
  Used for building the user interface, dashboard components, visual charts, and interactive visualizations.
- **Third-Party Libraries & Tools:**
  - **Recharts / Chart.js** – for rendering interactive graphs and visual analytics.
  - **Mongoose** – for object data modelling (ODM) and simplified interaction with MongoDB.
  - **Socket.io** – for enabling real-time communication between backend and frontend.

2.3 Functional Requirements

Table – 1 (Showing All Functional Requirements)

Smart Home Dashboard: Functional Requirements

ID	REQUIREMENT SPECIFICATION	PRIORITY
FR1	User registration and secure authentication (Sign-up/Login).	HIGH
FR2	Allow users to manage (add, edit, delete) appliances (e.g., refrigerator, AC) and their metadata.	HIGH
FR3	System must be able to ingest timestamped energy (kWh) and water (litres) consumption data from sensor APIs.	HIGH
FR4	Display real-time consumption data for all appliances on a dedicated, interactive dashboard.	HIGH
FR5	Generate and display historical analytics with charts showing usage trends (daily, weekly, monthly summaries).	MEDIUM
FR6	Automatic generation and delivery of alerts/notifications when appliance consumption exceeds pre-set or calculated normal limits.	HIGH
FR7	Provide AI-based predictions of future consumption and customized, actionable conservation recommendations.	MEDIUM
FR8	Allow users to export generated reports (analytics, consumption summaries) to common formats like CSV or PDF.	LOW
FR9	Implement an Admin panel with capabilities for user management and overall system monitoring.	LOW
FR10	Store historical consumption data in MongoDB and allow comparison of usage across different time periods and appliances.	MEDIUM

2.4 Non-Functional Requirements

Table – 2 (Showing All Non – Functional Requirements)

Smart Home Dashboard: Non-Functional Requirements

ID	REQUIREMENT SPECIFICATION	PRIORITY
NFR1	Performance: The main dashboard view must load and refresh consumption data within 2 seconds.	HIGH
NFR2	Scalability: The system must be capable of supporting a growing user base and rapidly increasing data ingestion volumes (e.g., using MongoDB time-series optimization).	MEDIUM
NFR3	Availability: The service must aim for 99% uptime, excluding planned maintenance windows.	MEDIUM
NFR4	Security: All communication must be over HTTPS, user passwords must be hashed (Bcrypt), and session management must use JWT authentication.	HIGH
NFR5	Usability: The React-based user interface must be fully responsive and optimized for viewing on mobile, tablet, and desktop devices.	HIGH
NFR6	Maintainability: The MERN codebase must follow a modular architecture with clear documentation and consistent coding standards.	MEDIUM
NFR7	Integrity: The system must perform data validation, unit conversion, and noise filtering on ingested sensor data before storage and processing.	HIGH
NFR8	Compatibility: The system must be compatible with modern web browsers (Chrome, Firefox, Edge, Safari).	LOW
NFR9	Reliability: The alert engine must reliably process threshold checks and send notifications without delay.	HIGH
NFR10	Technology Stack: The application must be built exclusively using the MERN stack (MongoDB, Express.js, React.js, Node.js) as defined in the project scope.	HIGH

3. DESIGN

3.1 System design

The system is designed with a focus on scalability, maintainability, and efficient real-time data handling. The architecture ensures that various components—such as the backend, frontend, and database—work together seamlessly while supporting future enhancements. The design also emphasizes modular development and optimized data flow to manage continuous sensor readings and analytical operations.

The system incorporates the following core design principles:

• **Modular Monolith Architecture**

- The application is structured as a modular monolith, enabling clear separation of concerns while keeping deployment simple.
- Each module—authentication, data ingestion, analytics, alerting, and device management—functions independently but operates within a unified codebase.
- Facilitates easier debugging, faster development, and the flexibility to evolve into microservices later if needed.

- **Optional WebSockets for Real-Time Updates**
  - WebSockets are integrated as an optional extension to enable live updates of consumption metrics on the dashboard.
  - Allows instant visualization of sensor readings without requiring page reloads or repeated API requests.
  - Enhances user experience for those who rely on continuous monitoring (e.g., technophiles or advanced users).
- **MongoDB Time-Series Optimization**
  - MongoDB's time-series collections are leveraged to efficiently store high-frequency sensor data.
  - Optimized for fast querying, aggregation, and long-term data retention.
  - Reduces storage overhead and improves read/write performance for real-time and historical analytics.

3.1.1 E-R diagram

This ER diagram models the data structure for a Smart Home Energy & Water Dashboard, centering on the Appliance as the main asset.

Core Flow: A User manages multiple Appliances, each connected to a specific IoT Sensor.

Data Ingestion: The Sensor records continuous Consumption Readings (time-series data for energy or water).

Analytics: The system processes these readings to trigger Alerts (if thresholds are breached) and generate Recommendations (AI suggestions), both of which are delivered back to the User.

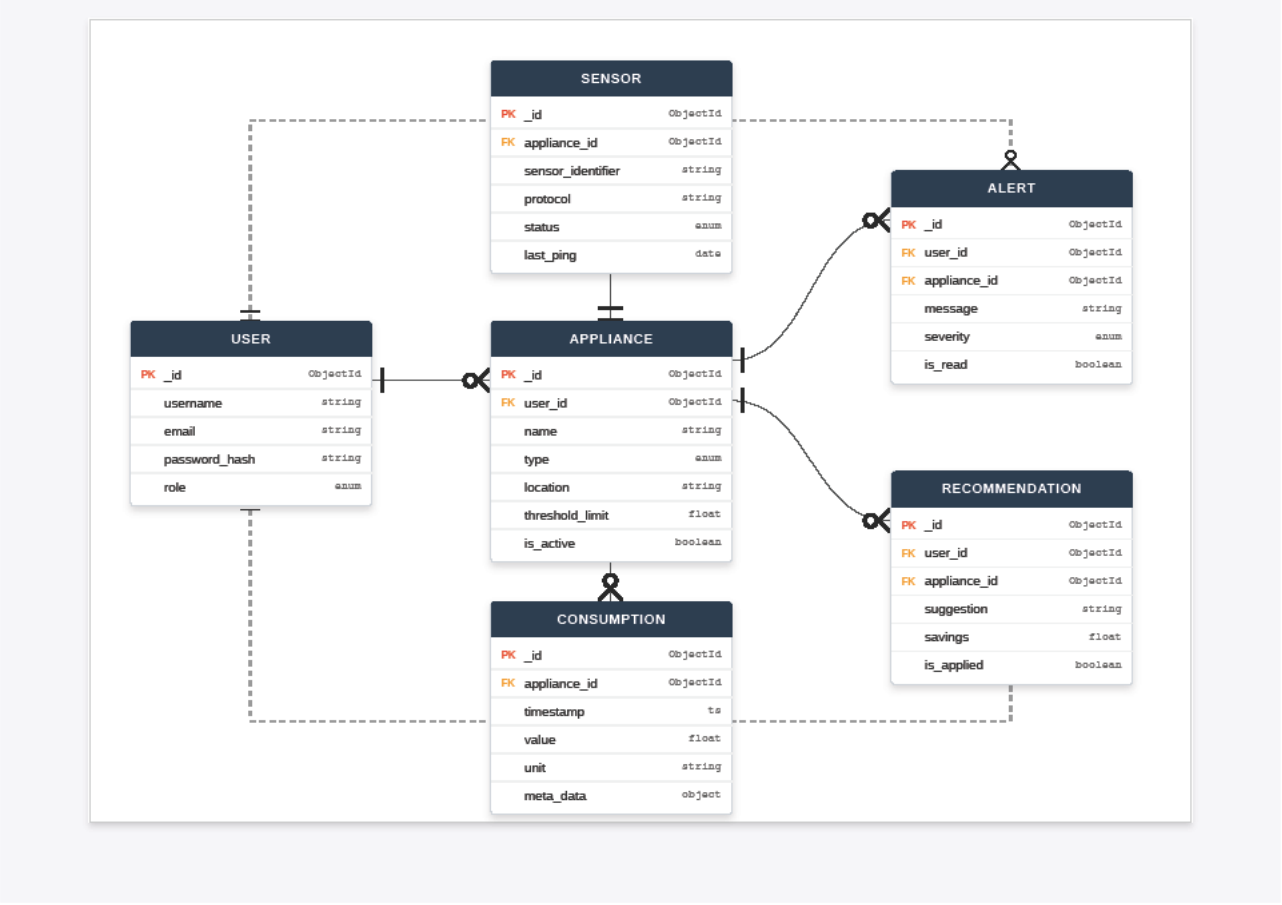

Figure – 2(Showing Relationship Between All The Entities)

3.1.2 Data Flow Diagrams (DFDs)

The Data Flow Diagrams (DFDs) illustrate the logical movement of data within the Smart Home Energy and Water Monitoring System. These diagrams help visualize how information flows between sensors, the backend, the database, and the frontend, showing how data is

processed at each stage. The DFDs provide a clear understanding of system functionality, user interactions, and internal operations.

The system is represented using three levels of DFDs:

- Context Level (Level 0)
- Level 1 (Expanded system processes)
- Level 2 (Detailed breakdown of internal processing steps)

• DFD – Level 0 (Context Diagram)

At the highest level, the system is viewed as a single process that interacts with three external entities: the User, Admin, and IoT Sensors. The diagram represents the overall data exchange between the system and these actors.

Key components:

- **External Entities:** User, Admin, IoT Sensor
- **Main System:** Smart Home Monitoring System
- **Data Flows:**
    - Sensor data flowing into the system
    - Processed data and dashboards flowing to the User
    - System management operations flowing to/from the Admin

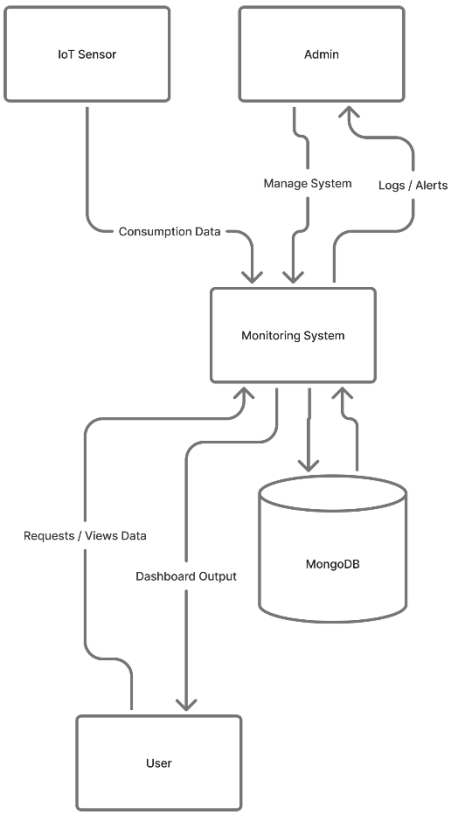

Figure – 3 (System Context Diagram)

• **DFD – Level 1**

Level 1 expands the system into its major internal processes. It details how sensor data is received, processed, stored, and visualized.

Primary processes include:

- **1.0 Data Ingestion**
  Receives incoming sensor readings through MQTT/HTTP/WebSocket.
- **2.0 Data Validation & Normalization**
  Converts raw sensor payloads into standardized formats and checks for errors.
- **3.0 Data Storage (MongoDB)**
  Stores validated consumption data, appliance info, user profiles, alerts, and predictions.

- **4.0 Analytics & Visualization Engine**
  Performs aggregations, builds charts, and provides usage insights.
- **5.0 Alerts & Notification Engine**
  Detects threshold breaches and sends alerts to users.
- **6.0 AI Prediction Module**
  Generates forecasts and recommendations.
- **7.0 User Dashboard (React)**
  Displays charts, tables, real-time graphs, and recommendations.

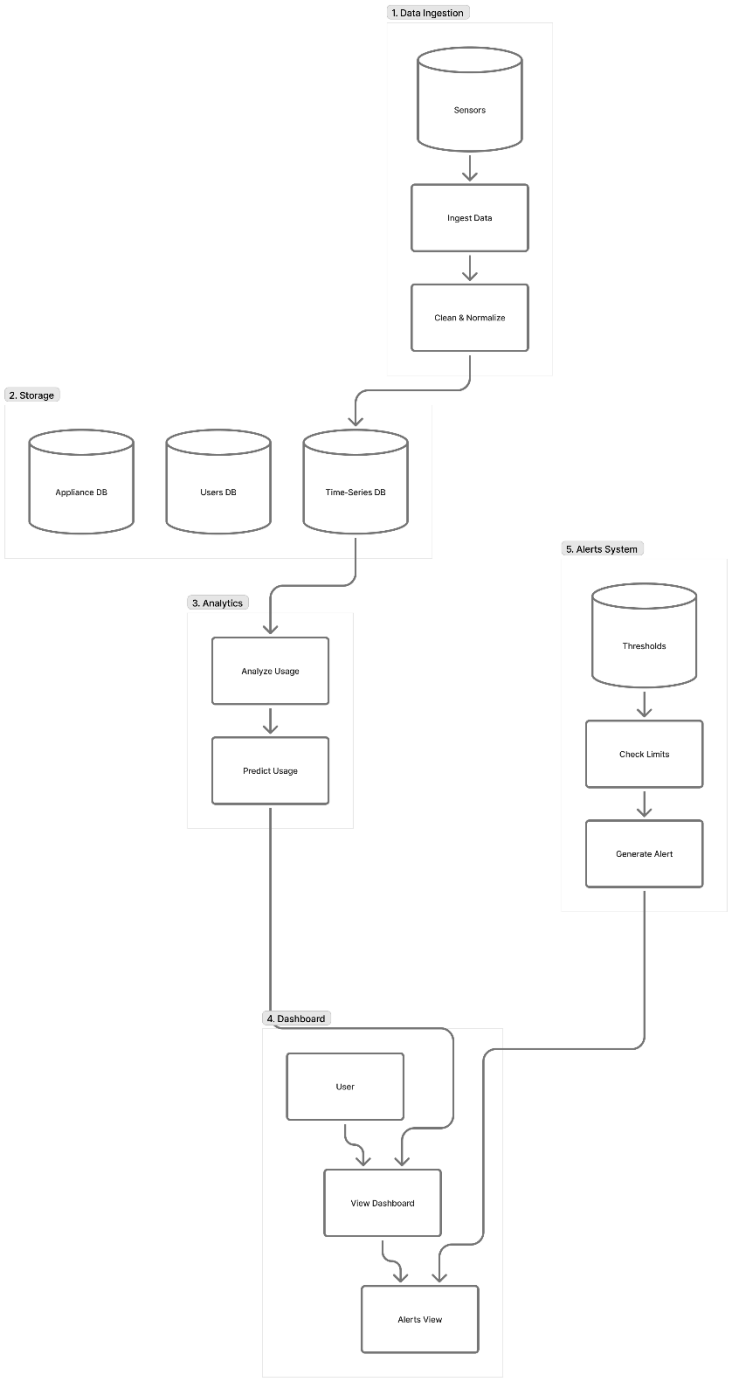

Figure – 4 (System Architecture Overview)

• **DFD – Level 2**

Level 2 provides a deeper breakdown of the internal operations within core processes such as Data Ingestion, Validation, Analytics, and Alerts.

Detailed processes include:

1.0 Data Ingestion

- **1.1 Validation of incoming packets**
- **1.2 Unit conversion (Watts → kWh, ML → litres)**
- **1.3 Noise filtering and anomaly detection**
- **1.4 Packaging data for storage**

2.0 Data Storage

- Efficient time-series insertion
- Indexing for fast query performance
- Historical data archiving

3.0 Analytics Module

- Aggregation of daily/weekly/monthly data
- Appliance-wise comparisons
- Trend detection
- Usage heatmaps and graph generation

4.0 AI Prediction Engine

- Time-series forecasting
- Peak prediction
- Personalized efficiency recommendations

5.0 Alerts Engine

- Threshold comparison
- Alert generation
- Notification dispatch (email, in-app, push)

6.0 User Interface Rendering

- Dashboard loading
- Chart rendering (Recharts/Chart.js)
- Real-time updates via WebSocket

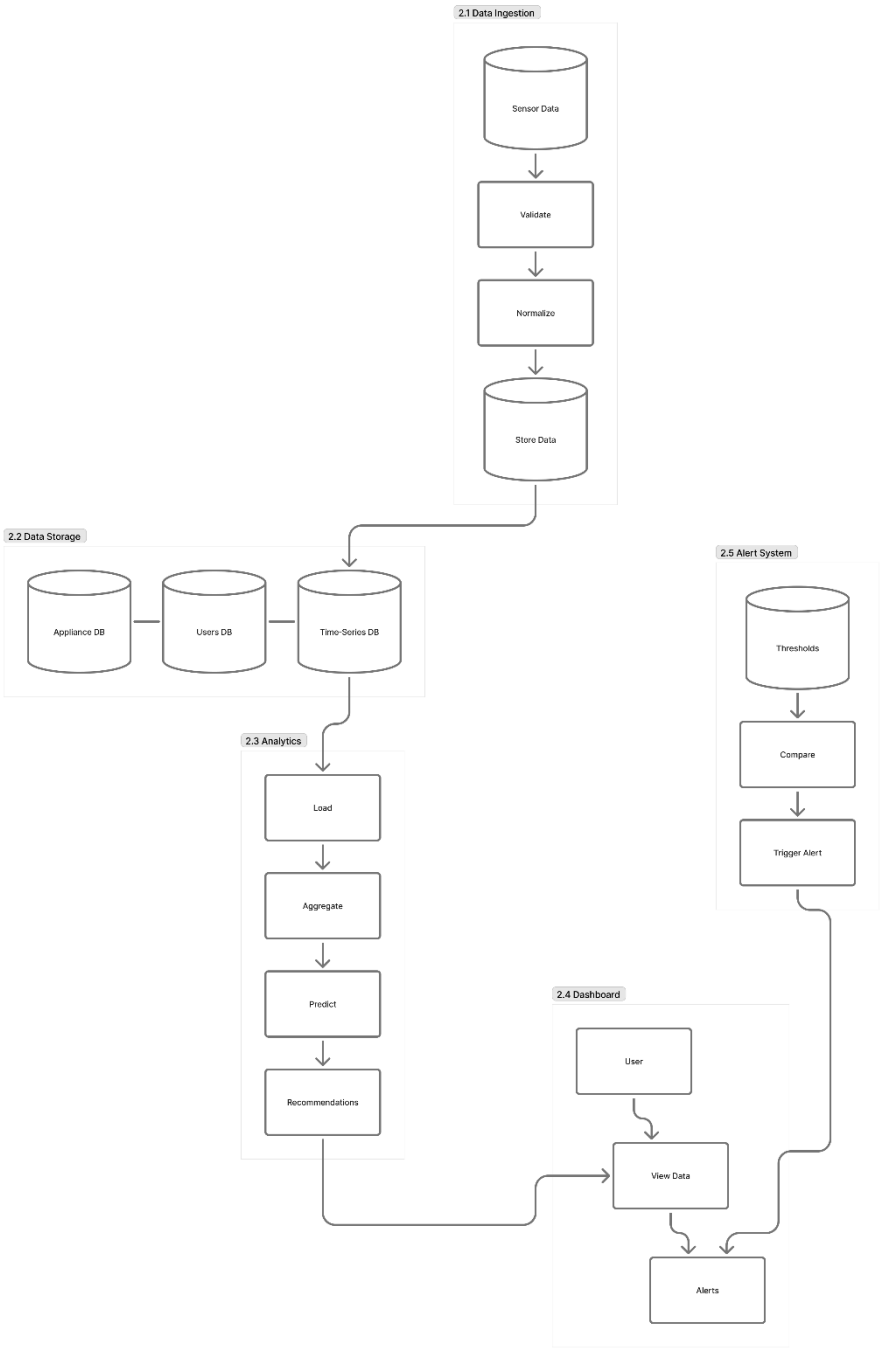

Figure – 5(Detailed Process Flow)

3.2 Database Design

Table 3. Users Collection

Stores user account information, authentication details, and personal settings.

Collection: users

```js
{
  _id: ObjectId,
  name: String,
  email: String,          // unique
  password_hash: String,
  role: String,           // 'user' | 'admin'
  created_at: Date,
  settings: {
    timezone: String,
    notifications: { email: Boolean, push: Boolean }
  }
}
```

Table 4. Appliances Collection

Contains details of household appliances linked to each user, including type, model, and location.

Collection: appliances

```js
{
  _id: ObjectId,
  user_id: ObjectId,      // FK -> users._id
  name: String,
  type: String,           // fridge, ac, washing_machine, cooler
  model: String,
  location: String,       // kitchen, bedroom
  sensor_id: ObjectId,    // FK -> sensors._id
  nominal_power_w: Number,
  created_at: Date,
  tags: [String]
}
```

Table 5. Sensors Collection

Holds metadata for IoT sensors such as unique ID, status, and connectivity information.

Collection: sensors

```js
{
  _id: ObjectId,
  sensor_uid: String,    // unique sensor identifier
  manufacturer: String,
  last_seen: Date,
  status: String,        // active/inactive
  meta: { ip: String, mac: String }
}
```

Table 6. Admin Logs Collection

Tracks administrative actions performed in the system for monitoring and auditing.

Collection: admin_logs

```js
{
  _id: ObjectId,
  admin_id: ObjectId,
  action: String,
  target: String,
  created_at: Date,
  details: Object
}
```

Table 7. Recommendations Collection

Stores AI-generated suggestions to help users reduce energy or water usage.

Collection: recommendations

```js
{
  _id: ObjectId,
  user_id: ObjectId,
  appliance_id: ObjectId, // optional
  generated_at: Date,
  type: String,  // 'timing'|'load'|'maintenance'
  text: String,
  score: Number
}
```

Table 8. Alerts Collection

Maintains records of threshold-based alerts triggered for energy or water consumption.

Collection: alerts

```js
{
  _id: ObjectId,
  user_id: ObjectId,
  appliance_id: ObjectId,
  threshold_type: String, // 'energy'|'water'
  threshold_value: Number,
  triggered_at: Date,
  status: String,         // 'sent'|'acknowledged'
  details: String
}
```

4. SCHEDULING

November 2025 — SRS & Design

- Finalize requirements
- Prepare SRS
- Create ER diagrams and DFDs
- Define system architecture

December 2025 — Environment & UI

- Set up development environment
- Configure project repository and CI
- Create UI wireframes and basic layout

January 2026 — Data Model & APIs

- Design database schemas
- Implement backend APIs
- Set up data ingestion and authentication
- Seed database with sample data

February 2026 — Frontend MVP

- Build dashboard screens
- Implement charts and basic UI components
- Connect frontend with backend APIs

March 2026 — Real-Time & Alerts

Early March:

- Implement WebSockets for real-time updates
- Add sensor data simulator

Late March:

- Build alert engine
- Add alert configuration and notifications

April 2026 — Predictions & Testing

Early April:

- Implement prediction models
- Add consumption forecasting

Late April:

- Conduct testing
- Perform security fixes

May 2026 — Documentation & Final Demo

Early May:

- Complete documentation and reports

Late May:

- Prepare final presentation
- Deliver demo and submit project

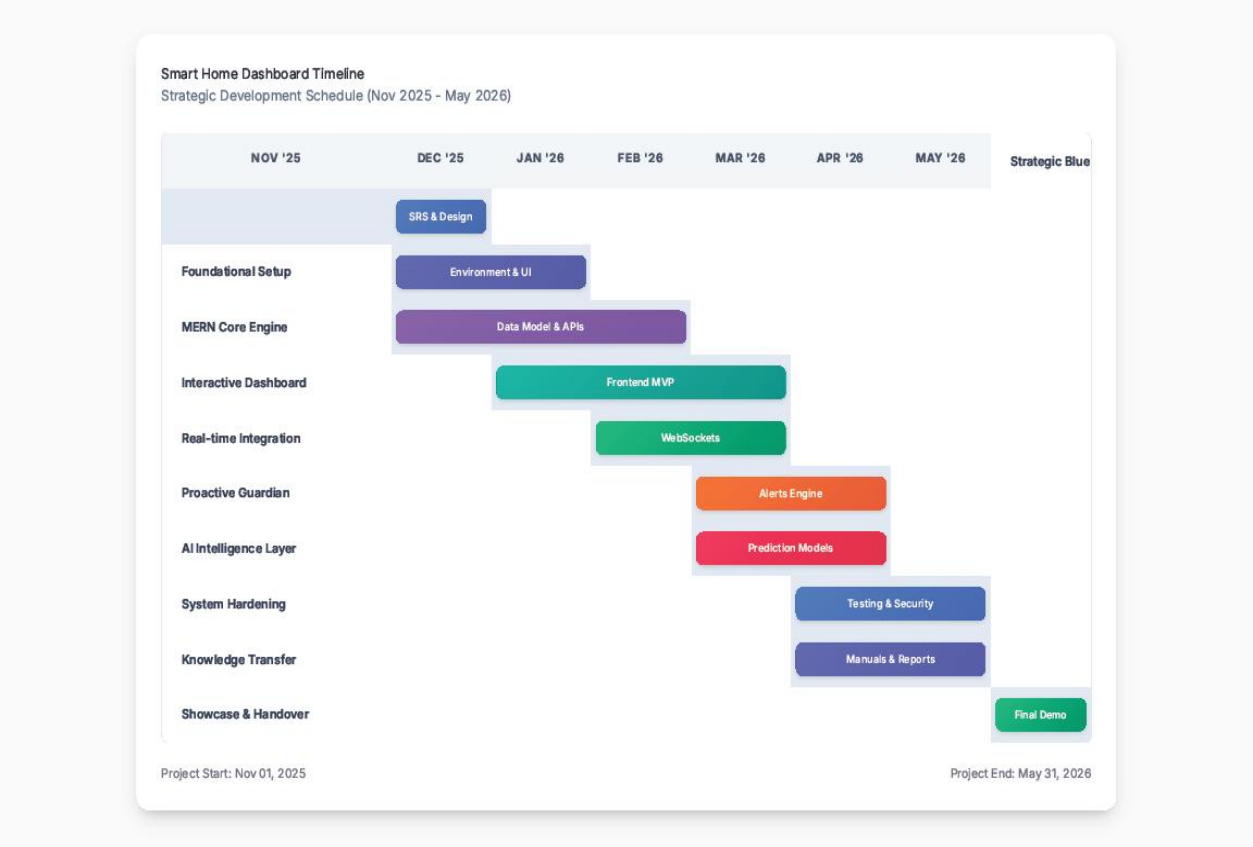

Figure – 6 (Gantt Chart)

REFERENCES

- International Energy Agency. (2023). Energy efficiency 2023 report.
- Government of India. (2022). Smart India Mission: Smart infrastructure and smart homes.
- Cisco Systems. (2022). IoT and smart home technologies report.
- Institute of Electrical and Electronics Engineers. (2021). IoT-based appliance-level energy monitoring. IEEE Publications.
- IBM Research. (2021). Smart utility monitoring using IoT systems.
- MongoDB Inc. (2023). Time-series data and IoT integration: MongoDB documentation.
- Node.js Foundation. (2023). Node.js documentation: Backend and REST API development.
- Meta Platforms. (2023). React documentation: Building interactive dashboards.
- OWASP Foundation. (2023). OWASP web security testing guide.
- TensorFlow. (2023). Time-series forecasting guide: TensorFlow documentation.