Technical Documentation & Report

# 1. Project Overview

The IoT-Based Smart Home Dashboard project is designed to provide centralized monitoring and control of smart home devices. It enables users to interact with devices such as lights, doors, and thermostats, while also monitoring real-time data like temperature, humidity. The system integrates a backend built with .NET Core, a frontend developed with React, and a database for device and user management. Authentication and authorization are implemented using JWT tokens.

# 2. Key Features

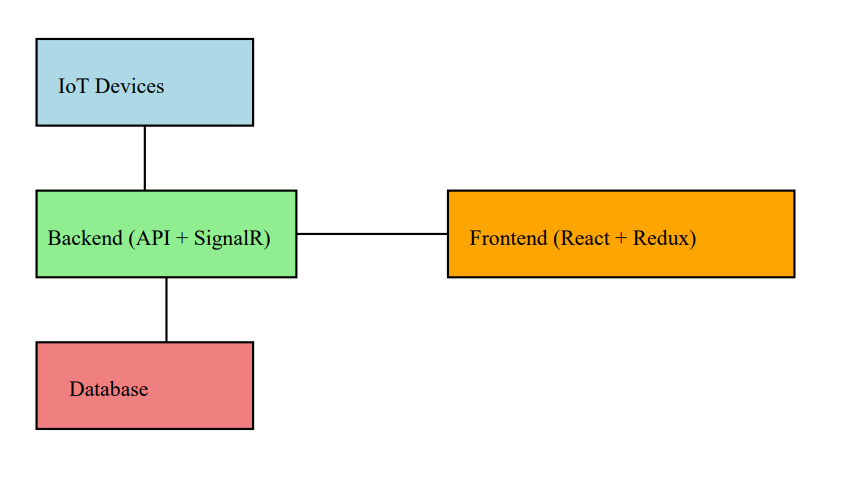
* Real-time monitoring of sensors (temperature, humidity, power usage).
* Device control (lights, thermostat, door locks) from the dashboard.
* Device Manager where we can Perform CRUD operations.
* Alerts and notifications for unusual events (e.g., security breach, power surge).
* Authentication & authorization using JWT.
* Responsive and user-friendly UI built with React and TailwindCSS.

# 3. System Architecture

The system architecture follows a layered structure that integrates frontend, backend, and database with real-time communication(SignalR).

***Flow of the system:***

1. User interacts with the React frontend (dashboard UI).
2. API requests are sent to the .NET Core backend for authentication, device control, and metrics retrieval.
3. Backend communicates with the database to store and fetch device/user data.
4. IoT Simulator or real devices generate sensor data and send updates to backend.
5. Backend pushes real-time updates to frontend via SignalR.
6. Dashboard updates device states, and metrics for the user.



# 4. Setup Instructions

Follow the steps below to set up the project:

* Clone the repository or extract the provided source code zip file.
* Repo Link:
* Backend Setup:

1. Install .NET 7 SDK or later. Or visual studio community installer.
2. Configure the database connection string in appsettings.json.(under DefaultConnection replace with your DB connectivity string).
3. Build the Project first before running migrations.
4. Install these NUGET Libraries:-
   * + Microsoft.EntityFrameworkCore
     + Microsoft.EntityFrameworkCore.Tools
     + Microsoft.EntityFrameworkCore.Design
5. Run migrations and start the backend server.
   * + dotnet ef migrations add Migrationname
     + dotnet ef database update

* Frontend Setup:

1. Navigate to the frontend directory.
2. Run `npm install` to install dependencies.
3. Start the frontend using `npm run dev`.

* Access the system at `http://localhost:5173` (frontend) and backend APIs at `http://localhost:5125`.

# 5. API Documentation

* POST /api/auth/login - Authenticate user and return JWT token.
* POST /api/auth/register - Register a new user.
* GET /api/devices - Retrieve list of devices.
* POST /api/devices – Creates a device.
* PUT /api/devices – Updates a device.
* DELETE /api/devices-Deletes a device.
* POST /api/control/light/{id} - Toggle light status.
* POST /api/control/door/{id} - Lock/unlock door.
* POST /api/control/security/{id} – Turn on and of security camera.
* POST /api/control/thermostat/{id} - Update thermostat settings.
* GET /api/sensors/security – Fetch security camera alerts.

# API Testing Report

API testing was carried out using Postman and swagger. The following tests were performed:

* Login API with valid and invalid credentials.
* Register API with missing/valid data.
* Device fetch API returns correct devices list also test crud operations.
* Device control APIs update device status correctly.
* Metrics API provides real-time data with valid JWT token.
* Unauthorized requests return 401 Unauthorized response.

1. **Deliverables**
   * + **Postman Collection For to test import the json file present in IoT-Smart-Home-Dashboard\docs**
     + **A Video which showcases the overall system flow.check docs folder.**
2. **Conclusion**

The IoT Smart Home Dashboard successfully integrates device management, real-time monitoring, and visualization into a unified platform. Future improvements include mobile app integration, and AI-driven predictive analytics for energy optimization.