**Requirement Specification Document**

**Project Name:** IoT Smart Home Dashboard  
**Author:** Venkatesh Santhu  
**Date:** 06/09/2025

**Table of Contents**

1. Introduction (Problem Statement)
2. Background
3. User Stories
4. Functional Requirements
5. Non-Functional Requirements
6. Deliverable

**1. Introduction**

**1.1 Problem Statement**

Users currently manage each device separately, which is time consuming and error prone. The current method of managing smart home devices is fragmented and often requires using multiple apps or manual control.

The IoT Smart Home Dashboard project aims to develop a unified console-based dashboard that allows users to control and monitor smart home devices using a single application built with Java Full Stack technologies.

By allowing remote ON/OFF control, the system also helps reduce unnecessary power consumption when users are away from home.

**2. Background:**

As a smart homeowner, the user wants to control all of my smart devices from a single app to manage their home's technology efficiently.

The application simulates an IoT dashboard thatprovides device control, real-time status monitoring, and backend support for storing device and user data using DynamoDB.

**3. User Stories**

**3.1 Customer Registration**

As a new user, I want to register by providing my name, email, and password so that I can create an account.

**3.2 Customer Login**

As a registered user, I want to log in using my email and password to access my dashboard.

**3.3 Control TV**

As a user, I want to control the TV in a specific room so that I can turn it on/off remotely.

**3.4 Control AC**

As a user, I want to control the AC in a specific room.

**3.5 Control Fan**

As a user, I want to control the Fan in a specific room.

**3.6 Control Robo Vac & Mop**

As a user, I want to control my Robo Vac & Mop in any room.

**3.7 View Gadgets**

As a user, I want to view the list of my connected devices and their statuses.

**3.8 Change Gadget Status (On/Off)**

As a user, I want to change the status of any connected device between On and Off.

**3. Functional Requirements**

**3.1 Customer Management**

* The system should allow new customers to register.
* The system allows customers to log in using email and password.

**3.2 Device Control**

* The system shall allow customers to control the following devices:
  + TV (Samsung, Sony)
  + AC (LG, Voltas, Blue Star)
  + Fan (Atomberg, Crompton)
  + Robo Vac & Mop (only one model)
* The system shall support device control per room:
  + Hallway, BedRoom1, BedRoom2, BedRoom3

**3.3 Device Status**

* Customers can view the current On/Off status of each device.
* Customers can change device status from Off to On and vice versa.

**3.4 Data Management**

* Customer metadata and device information will be stored in DynamoDB.
* Sensor data will be saved as time-series data.

**3.5 Dashboard Interface**

* A console-based UI will provide options for login, control, and viewing devices.

**3.6 Networking**

* Socket programming will simulate device communication (sending/receiving commands).

**3.7 Device Topology**

* A graph data structure will represent device connectivity and relationships.

**3.8 DevOps**

* The application will be Dockerized.
* Jenkins pipeline will automate build, test, and deployment to VM.

**4. Non-Functional Requirements**

**4.1 Performance**

* Device commands should execute in under 1 second.
* Dashboard should load in under 2 seconds.

**4.2 Scalability**

* System supports up to 100 devices per user.
* High-write throughput support for DynamoDB.

**4.3 Security**

* Passwords must be stored using secure hashing.
* Only logged-in users can access control features.

**4.4 Availability**

* The system should have 99.9% uptime.
* One device's failure must not affect others.

**4.5 Usability**

* Console menu must be intuitive and easy to navigate.
* Every user’s action will return a clear success or failure message.

**4.6 Maintainability**

* Code will follow Java coding standards.
* Modularity and separation of concerns will be applied.

**4.7 Testability**

* JUnit or Cucumber (TBD) for unit and integration testing.

**4.8 Portability**

* The application runs on Java 17+.
* Docker support allows cross-platform deployment.

**4.9 Reliability**

* Handles socket communication failures gracefully.
* Device data retries if DynamoDB is temporarily down.