Project 3 (4 Weeks, Teams of 3)

Smart Lab on the Web : *Simulating the Internet of Things through the Web of Things*

Access → Find → Share → Compose

General Note

While this document presents a reference scenario for guidance, you are encouraged to design and implement your own Web of Things scenario. Any alternative concept, as long as it adheres to the same principles of interoperability, openness and composition through Web interfaces, is acceptable.

Projects that go beyond the suggested example (e.g., with more devices, advanced automation rules, security layers, data analytics, or interaction with external APIs) will be considered for a bonus grade.

Context

This project focuses on the application-level aspects of the Web of Things, deliberately omitting the network and hardware layers. No sensors or embedded devices are needed since all interactions are simulated through lightweight web services.

The aim is to explore how everyday things can be represented, exposed, and managed on the Web as if they were real connected objects. Through this simplified, software-only approach, the project highlights the essential principles of interoperability, openness, and composability that define the Web of Things, while remaining easy to test and extend without physical hardware.

Objective

The project aims to design and implement a Web of Things environment where virtual devices ("things") can interact through web-based interfaces. Each virtual thing exposes RESTful endpoints for its properties, actions, and events. These interfaces enable discovery through a registry, controlled access through security policies, and composition into automated interactions among multiple things.

The final outcome is a functional demonstration of interoperability, real-time interaction, and secure coordination between distributed web-based objects.

Reference Scenario

The Computer Science department aims to develop a simple and functional demonstrator consisting of three virtual things : a *Thermostat*, a *Lamp*, and a *Motion Sensor*. Each thing operates as an independent web service that exposes its properties, actions, and events through RESTful APIs.

A central *gateway service* acts as the registry and point of coordination. It allows new things to register, provides discovery endpoints to list and access them, and relays real-time events to connected clients. The gateway also manages authentication through access tokens and enforces secure interactions between users and things.

A web-based dashboard provides an integrated interface where users can :

— view the list of registered things and their descriptions,

— inspect the current state (properties) of each thing,

— modify settings such as temperature, brightness, or power,

— trigger actions like turning on the lamp or setting a target temperature,

— and visualize live updates as things change state or emit events.

The system must also implement a set of automation rules that combine multiple virtual devices and demonstrate composition across services :

— **Rule 1 (Motion-based lighting) :** When motion is detected, the lamp switches on for one minute and then automatically turns off.

— **Rule 2 (Temperature comfort) :** If the temperature drops below 19°C and motion is detected, the thermostat activates its heating mode until the target temperature is reached, and the lamp turns on to indicate comfort mode.

Rule 3 (Energy saving) : If no motion is detected for five minutes, the lamp turns off while the thermostat switches to eco mode instead of turning off completely. In this mode, the target temperature is automatically lowered to 17°C to reduce energy consumption. When motion is detected again, the thermostat raises the target temperature and resumes heating until 19°C is reached.

— **Rule 4 (Manual override) :** Any manual action triggered by the user from the dashboard (such as switching the lamp or changing the temperature) takes precedence over automated rules.

System Design and Architecture

The system is composed of three independent web services representing the virtual things : a thermostat, a lamp, and a motion sensor, together with a central *gateway service*. Each thing acts as an autonomous web resource, accessible via standard HTTP requests and exposing its state and operations through RESTful APIs.

The gateway acts as a directory and coordination point. It allows new things to register, maintains a list of active devices with their descriptions, and provides discovery endpoints for clients. It also manages access control through security tokens and executes automation rules that coordinate multiple things.

A web-based dashboard serves as the main user interface. It retrieves the list of registered things, displays their current state, and allows users to trigger actions such as changing a property value or invoking an operation. Whenever a thing's state changes, for example when the lamp turns on or the thermostat updates its temperature, the gateway broadcasts the update to all connected clients through a live event channel, keeping the dashboard synchronized in real time.

All exchanges rely on lightweight web formats such as JSON, and access to sensitive operations is protected by authentication tokens included in requests. This architecture illustrates how the Web of Things enables interoperability, dynamic discovery, secure access, and event-driven interaction between distributed virtual devices in a purely web-based environment.

Deliverables, Installation, and Execution

The final submission must include all source code, documentation, and materials required to reproduce the project. All components should be organized in a clear and documented codebase, ideally hosted on a public or private Git repository (GitHub, GitLab, etc.), with concise installation and execution instructions.

The project must include :

— A documented **codebase** implementing the full system, including the gateway, virtual thing services, and dashboard, producing clear and well-structured outputs with explicit error messages.

— A short **report** (4–5 pages) describing the architecture, design choices, discovery process, security mechanisms, and rule execution logic.

— A concise **demo video** (4-5 minutes) showing discovery, read/write operations, one action, real-time updates, and at least one automation rule.

— An **installation guide** included in the main `README.md`, explaining how to install and run the system locally. It should mention the required environment, either Node.js or Java, and describe how to install dependencies, start all services, test the APIs, and reproduce the demo scenario with events and automation rules.

Submission deadline

November 14, 2025