Welcome to the software design take-home assessment! The purpose of this assessment is to evaluate your ability to design high-quality software architecture.

Instructions:

1. Read the problem statement carefully and make sure you understand the requirements and constraints.

2. Identify the key entities, behaviours, and relationships in the problem and sketch out a high-level design diagram.

3. Identify the design patterns that can be used to implement the required functionality.

4. Implement a working solution in Typescript (Node), Java or ColdFusion, making sure to follow good coding practices such as using meaningful variable and function names, commenting on your code, and handling error conditions.

5. The working solution needs to be extensible and show the ability to turn open/close a garage door, turn on/off a dishwasher and a living room light as well as undo functionality.

6. Test your solution thoroughly to ensure it meets the requirements and produces the expected output.

7. Submit your solution along with any supporting documentation such as a design diagram or explanation of your design decisions.

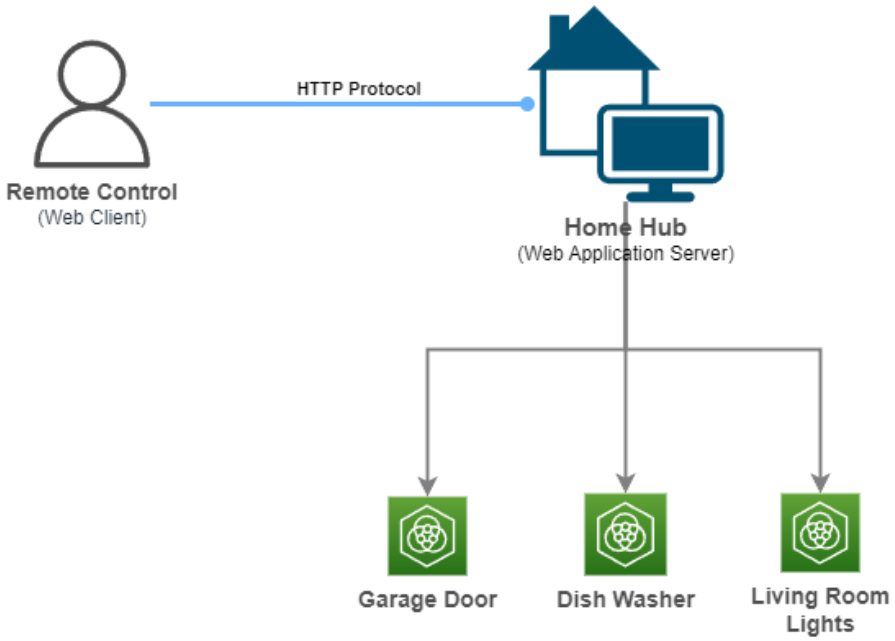
Problem Statement

aXmos is set to bring a Cloud Based Home Automation system built on standard web technologies to market using the SAAS model.

However, it is early days and engineering wants a first proof of concept to define the foundations for the API design that will allow sending commands from a remote controller to the Home Hub.

Design a system that enables client devices to communicate with a central home automation hub that controls various devices around the home.

The system should allow customers to easily control various supported devices. Also, it should be easy to add extra yet unknown devices to the system that the remote should be able to control without reworking the remote controller clients.

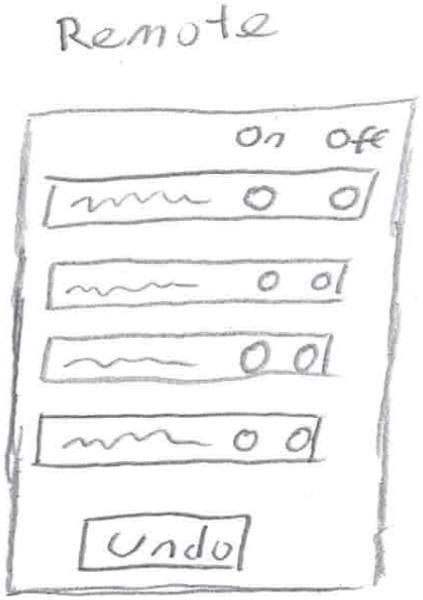


It is unclear what hardware will be used as a remote controller for interacting with the system. Possibly there could be various devices sold that could act as a remote controllers for the system.

What is clear is that all remote controls will communicate with the Home Hub using the standard Web HTTP protocol like any standard web application.

The proof of concept is solely concerned with the API used by a remote to control the Home Hub as such it is sufficient for the Home Hub to simulate turning devices on/off by writing to a console or log file.

To the right is a paper prototype of the UI for a remote controller.



The remote has several slots with two buttons per slot. Customers can assign an arbitrarily supported device to any slot.

Supported devices can be assigned to any slot and turned on or off using one of the two buttons assigned to a slot.

The business is adamant that the system should support the ability to undo previous actions, as such the remote also features a single “undo” button which will undo the previous user action.

Assumptions for the coding part only

- There is no requirement to implement any security mechanism.
- There is no requirement to implement any multi-tenancy.
- Assume this is a single-user system.
- There is no requirement for a database. If you need persistence do whatever is simplest for example in memory or a flat file.
- There is no requirement to implement any UI or Client.
- Simulate the Home Hub turning devices on/off by writing to a log or the console.
- Where there is ambiguity make appropriate assumptions and note these.

Deliverables

- A working API that enables a remote control to turn devices on/off via the Home Hub.
- Link to a GIT Hub Repo with the Working Code and README.
- Diagrams that support the software design and development.
- Any documentation and notes necessary to call the API, run the application and reason about the code should be added to a README file in the root of the project in markdown.