Home Automation System Web UI

**Web Application Dev with Java**

**EN.605.782.81.FA12**

**Project Preliminary Proposal**

Kyle Broadway

The Home Automation System (H.A.S.) Web UI provides a web interface that enables a user to monitor and control a H.A.S. java backend that communicates with various sensors and actuators over a wireless ZigBee network.

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Details | Initials |
| 0.1 | 9/29/2012 | Creation | KWB |

# Table of Contents

1. Background…………………………………………………………………………………………………………....2
2. Introduction……………………………………………………………………………………………………………2
3. Work Required………………………………………………………………….…………………………….....….3
4. Languages Used…………………………………………………………….…………………………………..…..3
5. Page Flow Diagram…………………………………………………………………………………………..…….4
6. Sample Page…………………………………………………………………………………………………..………5

# Background

The Home Automation System (HAS) was created in the spring semester of 2012 as a class project of the Embedded Systems Development Laboratory (EN.525.743(31)) course at Johns Hopkins University. HAS is composed of a Java backend and numerous remote sensor and control modules. The modules are positioned in strategic locations throughout the house to monitor or control various household settings such as ambient light, temperature, or humidity. Each module is independent and unaware of any other module in the overall network. All modules are tied to the Java backend through a wireless ZigBee network, centralized at the backend. The backend is responsible for gathering all measurements recorded by the sensors and interpreting them in a way that enables the backend or a user to make informed decisions regarding exposed controls. For example, an ambient light module may record a high measurement of ambient light, indicating that blinds are open in a room. If the temperature is too high in that room, the backend shall interpret the data and come to a conclusion to close the blinds in an effort to lower the room temperature.

The backend operates by dynamically creating a map of all discovered sensors and controls, mapped by the unique ID reported by the module. With knowledge of the available sensors and controls, a user interface may be exposed, allowing direct control and monitoring of HAS as a whole. A web interface for HAS is ideal, because of the cross-platform accessibility that a web interface can provide.

# Introduction

HAS is currently only a collection of controls and sensors, networked together with a Java backend. The current user interface for HAS is very utilitarian, since it is still very much in the prototype phase. Creating a web interface for HAS can now be started, since the backend is in a workable state. The HAS web interface will provide a password protected SSL login for multiple user access. The ability to allow multiple users access to the system introduces the capability of permissions on various controls. The web interface will also provide the user with the ability to save controls to a pre-configured state, and return to this state at any time the user wishes to. For example, a user may set up light and thermostat settings for overnight, and save the profile for later execution. Nearly all pages in the web interface must be dynamically generated, because the backend will dynamically discover any new modules that are added to the system after power-on. Any new modules will be added to the backend’s internal map of modules, and exposed through the web interface on the appropriate page.

# Work Required

The required work to be completed for this project includes:

* SSL login implementation for at least two users
* Page creation of the following pages:
  + Controls: exposes miscellaneous controls discovered
  + Thermostat: includes temperature and ambient light measurements along with a thermostat control
  + Status: includes miscellaneous sensors discovered
  + Room view: A grouping of sensors and controls per room
    - Room view includes tabbed dynamic pages that show a single room at a time.
  + Profile Setup: allows a user to pre-configure modules to a particular state and save the profile, which can be executed at a later time.
* Dynamic page generation that represents module information and controls discovered by the backend.
* The ability to save a pre-configured state of all sensors and store it, associated with a particular user.
* Validation on all control input

# Languages Used

Languages used to implement the web interface are:

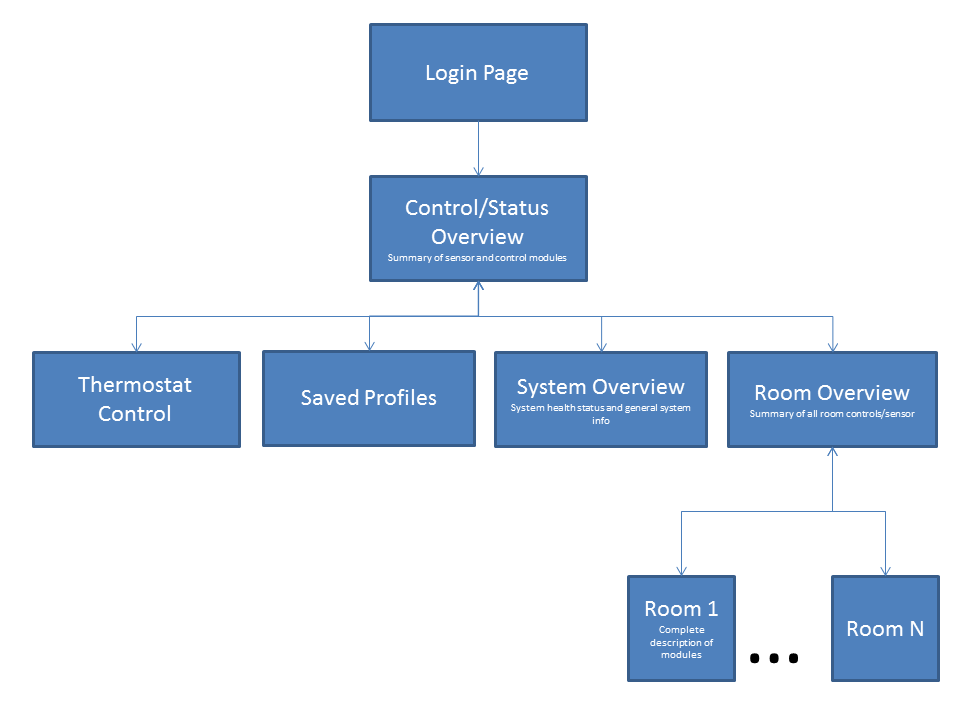
Server-side:

* Java, Java Servlets

Client-side:

* AJAX, Javascript, JSP, HTML

# Page Flow Diagram



# Sample Page

